

REMark[®]

March 1992

The Official Zenith Data Systems Computer Users' Magazine



Writing Articles for REMark Magazine
Page 13



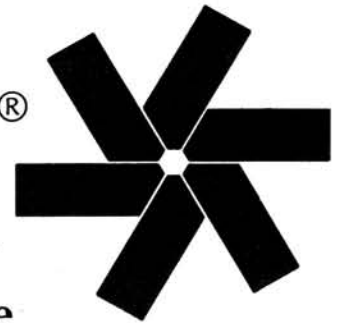
Share the Knowledge!

Have you done something interesting with your computer lately? Found a piece of software or hardware you don't know how you got along without? Designed a new product, be it software or hardware, for your system? By submitting this information in the form of a major article, you can share with others the knowledge of a particular subject. REMark magazine is currently looking for authors (novice or professional) to write articles. Even if you have never written before, give it a try! As a REMark author, you will receive up to \$400 for each article accepted and published. (For more information on current policies, call Lori Lerch at 616-982-3794.) So, Let's get to it and ...

Share the Knowledge!

REMark[®]

March 1992



The Official Zenith Data Systems Users Magazine

Getting Started With ...

Borland's Turbo Vision

John Trybus 5

Desqview 386:

Stealth Comes to the PC

Sanford Shapiro 9

How to Write Articles for

REMark Magazine

Using the Solunar Tables[®]

Hope Fulscribe (Alias, Richard J. O'Connor) 13

DOS 5.0: Is It Really Improved?

Mark Haverstock 16

Computers in the Age of Travel

Daveed Shachar 19

Introduction to C++

Thirteenth Installment

Lynwood H. Wilson 21

DR DOS 6.0

Terry W. Wilk 25

Sorting, Searching and KWIC Indexes

Part 2

John Day 32

PCs Assist Home Health Nurses

in Patient Care

David Dalton 40

Troubleshooting Modem Problems

Robert C. Brenner 41

By Your Command – Chapter 2

Richard J. O'Connor 43

Making the Connection

William Wills 47

Advertising

	Page No.
FBE Research Co., Inc.	18
WS Electronics	12

Resources

Software Price List	2
Renewal Form	4

Managing Editor
Jim Buszkiewicz
(616) 982-3837

Production Coordinator
Lori Lerch
(616) 982-3794

COM1 Bulletin Board
(616) 982-3956
(Modem Only)

Contributing Editor
William M. Adney

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

To Locate your Nearest:

Dealer 1-800-523-9393
Service Center 1-800-777-4630

	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. Check ZUG Product List for availability of bound volumes of past issues. Requests for magazines mailed to foreign countries should specify mailing method and include appropriate, additional cost.

Send Payment to: Zenith Users' Group
P.O. Box 217
Benton Harbor, MI 49023-0217
(616) 982-3463

Although it is a policy to check material placed in REMark for accuracy, ZUG 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 Zenith Data Systems Computer Centers.

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

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

Copyright (c) 1992, Zenith Users' Group

Software Engineer
Pat Swayne
(616) 982-3463

Secretary
Lisa Cobb
(616) 982-3463

Contributing Editor
Robert C. Brenner

Software

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM		DESCRIPTION	PRICE
		H8 - H/Z-89/90			
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
AUTOFIL (Z80 ONLY)	885-1110	HDOS		DBMS	30.00
BHBASIC SUPPORT PKG	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		CHKBK 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 PKG	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 I	885-1029-[37]	HDOS		GAMES	18.00
HARD SECT SUPPORT PKG	885-1121	HDOS		UTILITY	30.00
HDOS PROG. HELPER	885-8017	HDOS		UTILITY	16.00
HOME FINANCE	885-1070	HDOS		BUSINESS	18.00
HUG DISK DUP UTILITY	885-1217-[37]	CPM		UTILITY	20.00
HUG SOFTWARE CATALOG	885-4500	VARIOUS		PROD TO 1982	9.75
HUGMAN & MOVIE ANIM	885-1124	HDOS		ENTERTAINMENT	20.00
INFO SYS AND TEL. & MAIL SYS	885-1108-[37]	HDOS		DBMS	30.00
LOGBOOK	885-1107-[37]	HDOS		AMATEUR RADIO	30.00
MAGBASE	885-1249-[37]	CPM		MAGAZINE DB	25.00
MISCELLANEOUS UTILITIES	885-1089-[37]	HDOS		UTILITY	20.00
MORSE CODE TRANSCEIVER	885-8016	HDOS		AMATEUR RADIO	20.00
MORSE CODE TRANSCEIVER	885-8031-[37]	CPM		AMATEUR RADIO	20.00
PAGE EDITOR	885-1079-[37]	HDOS		UTILITY	25.00
PROGRAMS FOR PRINTERS	885-1082	HDOS		UTILITY	20.00
REMARK VOL 1 ISSUES 1-13	885-4001	N/A		1978 TO DEC '80	20.00
RUNOFF	885-1025	HDOS		TEXT PROC	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 PKG	885-1127-[37]	HDOS		UTILITY	20.00
STUDENT'S STATISTICS PKG	885-8021	HDOS		EDUCATION	20.00
SUBMIT (Z80 ONLY)	885-8006	HDOS		UTILITY	20.00
TERM & HTOC	885-1207-[37]	CPM		COMMUN & UTIL	20.00
TINY BASIC COMPILER	885-1132-[37]	HDOS		LANGUAGE	25.00
TINY PASCAL	885-1086-[37]	HDOS		LANGUAGE	25.00
UDUMP	885-8004	HDOS		UTILITY	30.00
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		CHKBK 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 CAT UPDT #1	885-4501	VARIOUS		PROD 1983 TO 1985	9.75
KEYMAP CPM-80	885-1230-[37]	CPM		UTILITY	20.00
MBASIC PAYROLL	885-1218-[37]	CPM		BUSINESS	60.00
NAVPROGSEVEN	885-1219-[37]	CPM		FLIGHT UTILITY	20.00
SEA BATTLE	885-1211-[37]	CPM		GAME	20.00
UTILITIES BY PS	885-1226-[37]	CPM		UTILITY	20.00
UTILITIES	885-1237-[37]	CPM		UTILITY	20.00
X-REFERENCE UTIL FOR MBASIC	885-1231-[37]	CPM		UTILITY	20.00
ZTERM	885-3003-[37]	CPM		COMMUNICATIONS	20.00

Price List

This Software Price List contains all products available for sale. For a detailed abstract of these products, refer to the Software Catalog, Software Catalog Update #1, or previous issues of REMark.

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM		DESCRIPTION	PRICE
H/Z-100 (Not PC) Only					
CARDCAT	885-3021-37	MSDOS	BUSINESS		20.00
CHEAPCALC	885-3006-37	MSDOS	UTILITY		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
DUNGN & DRAGONS (ZBASIC)	885-3009-37	MSDOS	GAME		20.00
ETCHDUMP	885-3005-37	MSDOS	UTILITY		20.00
EZPLOT II	885-3049-37	MSDOS	PRINTER PLOT UTIL		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	UTILITY		20.00
HELPSCREEN	885-3039-37	MSDOS	UTILITY		20.00
HUG BKGRD 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
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	ATRONOMY UTILITY		20.00
SMALL-C COMPILER	885-3026-37	MSDOS	LANGUAGE		30.00
SPELL5	885-3035-37	MSDOS	SPELLING CHECKER		20.00
SPREADSHEET CONTEST PKG	885-3018-37	MSDOS	VARIOUS SPRDST		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
BACKGRD 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
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 CAT UPD #1	885-4501	MSDOS	PROD 1983 - 1985		9.75
HUGMCP	885-3033	MSDOS	COMMUNICATION		40.00
ICT 8080 - 8088 TRANSLATOR	885-3024	MSDOS	UTILITY		20.00
MAGBASE	885-3050	VARIOUS	MAG DATABASE		25.00
MATT	885-8045	MSDOS	MATRIX UTILITY		20.00
MISCELLANEOUS UTILITIES	885-3025	MSDOS	UTILITIES		20.00
PS' PC & Z100 UTILITIES	885-3052	MSDOS	UTILITIES		20.00
REMARK VOL 8 ISSUES 84-95	885-4008	N/A	1987		25.00
REMARK VOL 9 ISSUES 96-107	885-4009	N/A	1988		25.00
REMARK VOL 10 ISSUES 108-119	885-4010	N/A	1989		25.00
REMARK VOL 11 ISSUES 120-131	885-4011	N/A	1990		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					
CARDCAT	885-6006	MSDOS	CAT SYSTEM		20.00
CHEAPCALC	885-6004	MSDOS	SPREADSHEET		20.00
CLAVIER	885-6016	MSDOS	ENTERTAINMENT		20.00
CP/EMULATOR II & ZEMULATOR	885-6002	MSDOS	CPM & Z100 EMUL		20.00
DUNGEONS & DRAGONS	885-6007	MSDOS	GAME		20.00
EZPLOT II	885-6013	MSDOS	PRINTER PLOT UTIL		25.00
GRADE	885-8037	MSDOS	GRADE BOOK		20.00
HAM HELP	885-6010	MSDOS	AMATEUR RADIO		20.00
KEYMAP	885-6001	MSDOS	UTILITY		20.00
LAPTOP UTILITIES	885-6014	MSDOS	UTILITIES		20.00
PS' 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 UTIL		20.00
TCSPELL	885-8044	MSDOS	SPELLING CHECKER		20.00
ULTRA RTTY	885-6012	MSDOS	AMATEUR RADIO		20.00
YAUD (YET ANOTHER UTIL DSK)	885-6015	MSDOS	UTILITIES		20.00

Attention!!

Zenith Data Systems Owners

When ordering ZUG software, be sure to specify what disk format you would like us to put it on. If you have an H-8, H/Z-89, or H/Z-90, you have the choice of using hard- or soft-sectored disks depending on your drive type. Order soft-sectored by adding a -37 to the end of the part number (i.e., 885-8009-37). Leaving off the -37 specifies a hard-sectored disk (i.e., 885-8009). If you own an H/Z-100 (not PC) series computer, you will always use the -37 at the end of the part number. For PC users, you have the choice of 5-1/4" (-37), 3.5" (-80), or 2" (-90) disks. Just add this number to the end of the ZUG part number (i.e., 885-3009-37, 885-3007-80, 885-3007-90).

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, MasterCard, and American Express phone orders, telephone the Zenith Users' Group directly at (616) 982-3463. Have the part number(s), description(s), and quantity ready for quick processing. By mail, send your order to: Zenith Users' Group, P.O. Box 217, Benton Harbor, MI 49023-0217. VISA, MasterCard and American Express require minimum \$10.00 order. No C.O.D.s accepted.

Questions regarding your subscription? Call Lisa Cobb at (616) 982-3463.

REMark Magazine Subscription & ZDS-COM1 Bulletin Board Information

Your subscription entitles you to receive REMark, our monthly magazine containing articles specific to Zenith Data Systems computer and generally to other PC Compatible computers. All articles in REMark are submitted by readers like you. We welcome YOUR articles, and will pay you for any we accept!

A REMark subscription also allows you full access to the ZDS-COM1 bulletin board system (COM1, for short). There are many, many megabytes of free and shareware software available for downloading to registered COM1 users. Full access also lets you order products from the "Bargain Centre" section of COM1. The money you can save in the Keyboard Shopping Club will pay for decades of REMark subscriptions.

Last, but definitely not least, your subscription puts you in touch with thousands of other Zenith Data Systems computer users, from whom invaluable information can be exchanged.

REMark subscriptions, currently \$22.95, can be obtained in one of three ways. First, by ordering one on the COM1 bulletin board (see the Keyboard Shopping Club section); second, by phone with VISA, MasterCard, or American Express; and third, through the US Mail using a credit card, money order or check made payable to: Zenith Data Systems. Our address is:

Zenith Data Systems Users' Group
P.O. Box 217
Benton Harbor, MI 49023-0217
(616) 982-3463

Once you receive your ID number, registration on the COM1 BBS is NOT automatic. It requires that you log on, enter your first name and last name EXACTLY as they appear on your REMark mailing label, and then enter your ID number as your password. The FIRST time you access the board, you must elect to start a NEW ACCOUNT and answer the various questions. Once you've done this, our automated scanner will compare the system's database against the subscription database. If you made no mistakes, you will be verified and given full access within 24 hours.

Once you've been authorized as a full member, several important things happen. First, you're given full downloading privileges of up to one megabyte per day. Secondly, you'll have full access to the message boards. And finally, you'll be able to take full advantage of the Bargain Centre product savings.



Detach this form, enclose your check, money order or credit card information (no cash please).

REMark Subscription / Renewal Form

New Member: Yes No Credit Card # _____

ID Number: _____ Exp. Date _____

Address Change?

Name: _____	U.S. Bulk Mail	<input type="checkbox"/> 19.95	<input type="checkbox"/> 22.95
-------------	----------------	--------------------------------	--------------------------------

Address: _____	U.S. First Class	<input type="checkbox"/> 32.95	<input type="checkbox"/> 37.95
----------------	------------------	--------------------------------	--------------------------------

City, State, Zip: _____	APO/FPO	<input type="checkbox"/> 32.95	<input type="checkbox"/> 37.95
	Surface Overseas		

Daytime Phone #: () _____	Air Printed Overseas	<input type="checkbox"/> 52.95	<input type="checkbox"/> 57.95
----------------------------	----------------------	--------------------------------	--------------------------------

Getting Started With...



John Trybus
Box 560
Lexington Park, MD 20653

Borland's Turbo Vision

Orphan at the SHOOP Window

Did you rush out to buy Borland's Turbo Pascal 5.0 to get all the great advantages in debugging, single-stepping, watch-points, on-the-fly value changes, the IDE and on and on?

Did you wonder why when Turbo Pascal 5.5 came out it had those funny, hard-to-figure OOP extensions instead of sticking to new advances in 'regular' programming concepts?

Did you look at the announcements for Turbo Pascal 6.0 and wonder why they didn't have new stuff that you could use to demonstrate your genius-like programming ability, instead of that thing called "Turbo Vision" that looks so inviting and seems so impossible to use? Do you feel like an smudge-faced, orphan programmer pressed up against the Bakery SHOOP window?

Com'n In!

If you see yourself included above, or if you're anxious to get OOP-ing but can't find the time to learn new things, this is a good place to get you started down the right path. In reading this article you won't become an overnight expert, but you will come to understand some of the ideas and methods driving object-oriented programming. You'll also learn much about Turbo Vision and how to use it to advantage.

It's time to come into the OOP shop, where everyone insists on the near future of programming lays.

Borland's Turbo Vision as introduced in their Turbo Pascal release 6.0 is a complex idea designed with a simple view;

making a programmer's life easier. It won't make every aspect of your programming chores easier, though. It won't necessarily make you a better programmer, but your finished products will be more professional-looking. It won't help with those quick-and-dirty one-timer programs (that seem to live forever). It won't help those sophisticated, tightly-coded, memory resident, hot-key routines you do so well, and that seem to work like magic. It won't help you design an interface between incompatible LANs, databases or spreadsheets (unless you're already an expert in those fields).

What Turbo Vision does do is make a gift to you of a highly flexible and capable (and totally reusable and almost no-sweat modifiable) interface between a program user and a program or application.

What's it cost?

The price? You have to learn what is effectively a new language. You'll have to learn some new debugging techniques. You'll have to let Turbo Vision do many of the things you used to do so well, like windowing and event handling. Having mastered Pascal isn't going to give you automatic mastery of Turbo Vision, although it will certainly help, since Turbo Vision is written in Turbo Pascal.

Wrapping the Turbo Vision interface around your program will also increase its size considerably. (As an example, consider the Borland-supplied program in the 6.0 Release of Turbo Pascal, HELLO.PAS; it consists of 103 lines of code, 2,577 bytes of storage space. It compiles to approximately

47K bytes on disk storage and runs in memory at over 82K bytes - just for the code! And all the program does is to pay lip service to how you feel. Something like your colleagues do for you on Monday mornings.)

What do I get?

That increase, however, is more than offset by;

- the capabilities provided in Turbo Vision,
- by the fact that it eliminates much of your concern with designing a 'look and feel' each time you sit down to develop a new application, while still allowing flexibility,
- by letting you avoid a lot of fragile, semi-duplicating utility coding for each program you write (such as for windowing, selecting files from lists, mouse code, etc.)
- by providing a built-in "ToolBox",
- debugging becomes considerably easier.

Where does Turbo Vision work best? In any application, although it may be at its best in developing any application that will be used by others.

Damn the Manuals . . .

Many of us like to think we can install a new language product, hit the "GO" button and muddle through to some reasonable result, that is, a workable program.

In other words, "Damn the manuals, full code ahead!"

That approach will work for some,

sometimes. It'll work with Turbo Vision, too. Look at the programs provided with Borland's Pascal Release 6.0.

You'll find TVEDIT.PAS, for example, a multi-window ASCII editor including pull-down menus, mouse support, context-sensitive "Help", and scrolling windows (which does, however, demonstrate a small bug in the release of EDITORS.PAS. See the sidebar.). The program is a mere 398 lines long, many of which seem somehow familiar, but whose purposes escape casual examination. That's not to even mention the program code at the end which consists of three lines, .INIT, .RUN, and .DONE.

(TVEDIT is a viable, compact, multi-window ASCII file editor you can use as a standalone, or include it in your application. There are other niceties available, such as on-screen time display and others mentioned below, that can also be included in your programs.)

Another is TVDEMO.PAS, which, besides its file viewing capabilities, includes some interesting features, such as; a Calculator, an ASCII conversion table, a Puzzle and a monthly Calendar.

You could start with either of these examples for study. To get an easier start, you'd probably look at the TVGUID01 through TVGUID22 sample programs and, after a while, guess about where to add your own program code. You'll make a lot of mistakes on the way, of course, like continuing to use ReadLn and WriteLn, like putting the main code in the usual place, say between the .RUN and .DONE statements, or leaving in a lot of menubar commands because you can't quite figure out how to remove or change them. Doing it this way will probably remind you of your early days in school and how you went about learning the mysterious ways of the opposite sex.

Finally, though, you will get your program to work.

You won't necessarily understand the capabilities available at hand. And there's all that funny new language about OOP.

Turbo Vision is OOP. Borland calls it " . . . an object-oriented application framework for windowing programs." You know what the Turbo Vision shell looks like because you already use Turbo Pascal, where it was used to produce the Integrated Development Environment (IDE), or Programmer's Platform. It's also found in a similar version in Borland's Turbo C++. (What? A software company using its own product to produce marketable products? Now there's a novel idea.)

Now it's your turn. Begin adding this familiar interface to your own programs and applications.

How will you do that without many hours of intensive study and practice?

New words, new .., new ..,new —

No!, no, no, No!

The manuals start by throwing at you new words, new definitions and new concepts all at the same time. Then they tell you it's just like what you already know, only different, that all you have to do is reprogram your lifelong programming habits.

It is true that sophisticated use of Turbo Vision will require a lot of study and practice. Nonetheless, I'll try here to 'jump start' those parts of your brain that have been lying dormant all these years just waiting for OOP concepts to surface. (And you thought those were just ordinary nightmares.)

First, a few words about a few OOP words. You keep hearing about them. No doubt you keep wondering what they mean, and what they mean to you.

FIELD(s) — in simple terms, this just means data.

METHOD(s) — program code. Specifically, self-contained code that performs fixed action(s) on a FIELD or FIELDS.

ENCAPSULATION — this means putting FIELDS and FIELD manipulation code (METHODs) together in a package and calling it an OBJECT. Effectively, this can also seal the package shut, like your transmission, but includes a handle (shift) that allows you to use its capabilities. (Although Turbo Pascal OOP doesn't require it be 'sealed', that is, unless you use the PRIVATE directive, users of the OBJECT can directly access its FIELDS.) Unlike a transmission, however, an OBJECT will never break because you almost never unseal it (assuming everything you put into the OBJECT was fully tested before release).

OBJECT — after you ENCAPSULATE the FIELDS with the METHODs, you give it a name, as for example, Borland called some of their OBJECTs; TScroller, TWindow, TMenuBar, and TDosStream.

You might easily guess that these OBJECTs would, respectively, include FIELDS and METHODs to:

- Set up horizontal and vertical scrollbars in a window (TScroller).
- Provide a window border, optional title and window number, also allows moving, sizing, zooming and closing. Besides all that, it listens for your mouse (TWindow).
- Display an active menu bar at the bottom of the screen (TMenuBar).
- provide access to DOS without exiting the program (TDosStream).

Good guesses! Now how about TRadioButtons, TListViewer, TDialog?

- Build in some user run-time choices by presenting mutually exclusive buttons (only one can be on at a time) on the screen (up to 65,536 buttons in a cluster — is that enough?) (TRadioButtons).
- Let the user view a list, perhaps for selection, like, say, a DOS DIR listing, or

a text file (TListViewer).

- A descendent of TWindow that opens a dialogue between program and user. The dialogue window can be moved and closed, but not changed in size and is not scrolling (TDialog).

How about the mysterious Borland TApplication? Perhaps it knows what you want to do in your program — just call on TApplication and let it provide a finished product. Well, not quite, of course. TApplication is, in effect, the Turbo Vision framework OBJECT for your application code. It handles all necessary Turbo Vision initializations (through the METHOD TApplication.Init — look it up on page 207 of the Turbo Vision Guide and you'll see reference to InitMemory, InitVideo, InitEvents, InitSysError, InitHistory, and a call to TProgram.Init for other initializations). At program end it handles the opposite chore, closing everything down (through the METHODs TProgram and TApplication.Done).

Note: Descriptions of all the Borland-supplied OBJECTs are in the Turbo Vision Guide, Chapter 13, OBJECT Reference. Specifics of FIELDS and METHODs are included. Other OBJECTs, such as the Editor provided by Borland, are described in the online documentation files.

INHERITANCE — After you've defined an OBJECT as above, you can define a **DESCENDENT OBJECT** — with the obvious purpose of it being a lot like the **ANCESTOR OBJECT** (inheriting all the ancestor's features, that is, the FIELDS and METHODs, hair color, nose, personality, your good looks, etc), but with such different and/or additional features as are needed for this new OBJECT (more functionality, new FIELDS, replacement METHODs, new METHODs, more brains, more money, nurturing skills plus pitching arm, etc.).

In your future programs, you can use the original Borland-defined OBJECT as is, wherever, whenever it's needed. Or, you can use your own developed **DESCENDENT** to override the original. Or even develop a **DESCENDENT** of your original **DESCENDENT** (Don't forget to develop a genealogy chart to keep up with your growing family. See Figure 3.1 in the Turbo Pascal Turbo Vision Guide.)

In any case, you NEVER need to change the original code. In fact, you don't even need the original code, just like you don't need your Great-Grandfather to create — well, you get the idea.

POLYMORPHISM — If you use the Borland OBJECT TWindow straight from the box, it can create windows with a frame, a background, a number (1 through 9, for Alt-n key selection), a name, scrollbars, etc. TWindow includes a METHOD called HandleEvent, which is there to process commands relevant to the window, like moving, sizing, closing, scrolling, etc.

TWindow knows how to do all these things to itself - your code doesn't ever worry about such things. You can then create a DESCENDENT of TWindow, overriding whatever of TWindow you need to change. Perhaps you want a purple background. Besides that, you need to put code in your program to do whatever your program does when a window is open. To accomplish this, you override the TWindow's HandleEvent (and it's called just that, TWindow.HandleEvent) with your own HandleEvent. You still have all the functionality of TWindow's HandleEvent, plus you have your own additional coding. The METHOD is still called HandleEvent, but now it's redefined in your program, your OBJECT, and you call it something like TMyPurpleWindow.HandleEvent. It's a Pascal Procedure with the name you give it, just like any other Pascal Procedure. The two METHODS have the same name, i. e., HandleEvent, but the new one overrides the original one. These two METHODS are POLYMORPHIC. (Different first name, same last name. Parent-child. Mother Michelle.Jones, and daughter Jenny.Jones.)

Actually, you won't see the terms ENCAPSULATION, INHERITANCE and POLYMORPHISM once you get into writing code. These are theory words needed to define an OOP system. Nonetheless, understanding them is necessary to understanding object-oriented programming, and of course, will help you to understand Turbo Vision. You'll never run afoul of a reserved word called ENCAPSULATION, nor ever call a thing named POLYMORPH in your program code. Maybe in your house when you're calling your children. Yes, that's right. Your children, while probably not theoretical, do embody the ideas of ENCAPSULATION (each has its own possessions and ways to deal with them), INHERITANCE, (they're basic human beings, like you, getting their genes, and jeans, from you), and POLYMORPHISM (they do some, or all, of the same things you do, usually differently).

ABSTRACT TYPE - after the gurus of OOP (gurOOPS?) have presented all the new words and ideas and assume you've mastered them all, they'll always throw in some new ones in casual discussion - and never define them. You're supposed to divine that ABSTRACT TYPE is simply another name for OBJECT, with one difference; the OBJECT is developed to have DESCENDENTS (in other words, to have its METHODS overridden, or to be 'POLYMORPHISIZED'). It's used like a blueprint, or a program flow chart, even though an ABSTRACT TYPE can have FIELDS and METHODS just as any other OBJECT.

EXTENSIBILITY - another (theory) word for the idea of INHERITANCE, which implies the ability to add functionality to the original OBJECT without touching the

original code.

MUTE OBJECTS - kids, or your kid sister, when they're at Grandmother's, two states away (and the phone's broken), or, in OOPland, any part of your program that works, but is invisible to the user. This is in contrast to;

VIEWS - any element you see on the screen when your program is functioning. Each item you see is also called an OBJECT. Several VIEWS can be combined to work together, e. g., putting together a frame, scrollbars, background, window name, etc., as TWindow does, results in a collective VIEW known as a GROUP. When your kid brother is off at the store, buying school supplies for you to use, he's a MUTE OBJECT. When he comes home and comes into your room, he's a VIEW. And, when he starts whacking you in the head, he becomes an;

EVENT - which is what your program is all about - it handles EVENTS, meaning anything to which the program must respond, be it modem input, keystrokes, mouse action, head whacks, whatever.

To Work at Last?

Finally, having absorbed the essence and theory of OOP programming, we're ready to unleash our excited cranial neurons to tackle the task at hand, namely, To Write A Program.

Almost.

A few more words, some of which will actually show up in your program code.

Turbo Vision Units - these are the units you'll include in your programs via the USES statement. For most programs, the first five will usually be called on.

APP - elements of the basic framework, such as TApplication and TDesktop.

OBJECTS - some basic OBJECT definitions for Turbo Vision, like TDosStream and TSortedCollection.

MENUS - for menus and status bars OBJECTs, TMenuBar, TStatusLine, etc.

VIEWS - all the OBJECTs needed to write to the screen, like - TFrame, TListViewer, TScrollBar, etc.

DRIVERS - for EVENT handling - InitEvents, DoneEvents, GetKeyEvent, InitSysError, etc. and, less often used

DIALOGS - for dialogue boxes

TEXTVIEW - for displaying text in a scrolling window

MEMORY - memory management

HISTLIST - to keep a running record of certain EVENTS.

METHODs - OOPs. Sorry. You already know what METHODS are. As long as you're here, though, how about a quick discussion of STATIC and DYNAMIC METHODS?

(First, it's necessary to make some assumptions to give this discussion validity. Assume there exists an OBJECT called TBaseBallGame. Included are all the

OBJECTs with their data FIELDS and METHODS needed to play a game, things like TCatchBaseBall and TThrowBaseBall, various Init and Done and HandleEvent METHODS, TDoNext, TStealBase, THitToThirdBaseMan, etc. Let's say the whole set was supplied by Borland, just like TVEDIT and TVMEMO and all those things that really are.)

You would be able to compile and play the game, of course, just as you're able to compile and use TVEDIT and TVMEMO.

If all the METHODS were defined as STATIC METHODS, that's all you could probably do - play baseball. Every time you call on TThrowBaseBall, it's going to throw a baseball. TCatchBaseBall will always catch it, and probably wears a glove to do so. You end up with fixed and very efficient METHODS that respond consistently. You can override the METHODS with your own, but if a STATIC METHOD you call on calls another STATIC METHOD, there's no facility for overriding that called METHOD.

What about football? Or Volley Ball? You probably best go write your own game.

What if you really wanted to play a variation of baseball, with a different size playing field, just two bases, Home and Second. Besides, you want livelier balls, bigger gloves, and heavier bats, and infielders can't throw base runners out, they must personally tag them.

If the authors of TBaseBallGame wanted you to have that flexibility in your gaming without having to give you the code they wrote, they would have defined many of their METHODS as VIRTUAL, also known as DYNAMIC METHODS. Then they'd give you a manual describing each data FIELD and METHOD in TBaseBallGame. They'd also compile all that code into Turbo Pascal Units and ship the Units to you. (Data FIELDS within OBJECTs cannot be overridden. You wouldn't be able to redefine the way a baseball is defined within TBaseBallGame. However, a METHOD can allow specifying the values used in the definition, like diameter, weight, etc.)

Because the METHODS are defined as VIRTUAL, you can override them even when the calls are buried in another OBJECT. For example; in the original game, when a ground ball is hit to the third baseman with no one on base, the proper action to be called in the OBJECT TBaseBallGame.TDoNext is the METHOD TThirdBaseManAction, where the ball is thrown to the first baseman. In your version, you create a DESCENDANT (details later) of TBaseBallGame.TDoNext called TMyAction. It includes a METHOD called TThirdBaseManAction. In that code, you override the METHOD that directs the third baseman's actions. You want the fielder to chase the batter on the way to second base (remember, your game has only the

two bases) and tag him out.

You haven't modified the original code because you don't have it nor need it. Through the mechanisms supplied by Borland, the original game code is executed as appropriate until the METHOD TThirdBaseManAction is called for. That's when your version is called on rather than the original. This is accomplished through the CONSTRUCTOR convention. See below.

Note: If we were talking about a real application called TBaseBallGame, it's obvious the code would be large and complex. The writer would have to provide the information that tells us which METHOD will normally get the EVENT we're interested in processing according to our needs, in this case, a ground ball to third, no one on base. Any other condition can be executed according to the original program unless we chose to make other processing exceptions. The description of TDoNext's FIELDS and METHODS in the "User's Guide" would provide the answer, for there we would find out what to override, specifically, in our code.

This is also where the term BINDING comes in; EARLY BINDING is what happens in STATIC METHODS, that is, the procedure to call is fixed at compile time. LATE BINDING is for VIRTUAL METHODS — you tell the procedure what it's dealing with at execution time. The CONSTRUCTOR Init(- parameters -) is how you do it.

Memory for VIRTUAL METHODS is allocated on the heap at run time, using the NEW() procedure. When the program is through with its task, it must release the space using the DISPOSE() procedure.

CONSTRUCTOR and DESTRUCTOR — sounds like 'build it up and tear it down', doesn't it? That's more or less what actually happens. Since all VIRTUAL METHODS require that memory be allocated at run time for the METHOD (so the code knows what to do next), Turbo Pascal provides the CONSTRUCTOR convention.

CONSTRUCTOR at first just seems to be another name for PROCEDURE, since the declaring code looks something like this;

```
TMyAction = OBJECT (TDoNext)
  CONSTRUCTOR Init (- parameters -);
VIRTUAL;
END;
```

The CONSTRUCTOR call is always required when you're overriding an existing OBJECT. It must have the same parameters as the Init METHOD of the OBJECT being overridden. It must be executed before you do anything else in your OBJECT and METHOD override code. It's how Turbo Vision knows what to allocate on the heap, how much space, etc. In short, it's how Turbo Vision figures out where to go next in response to an EVENT. What you

are CONSTRUCTing is the VIRTUAL METHOD TABLE, or VMT, the Borland-supplied vehicle that tells Turbo Vision what it's dealing with.

INSTANCE, INSTANTIATION — when you called on the CONSTRUCTOR (TMyAction.) Init, resulting in memory on the heap being allocated, you have created an INSTANCE of the METHOD. It now exists for the program and can be processed according to its needs.

This should be enough to keep you thinking about Turbo Vision for some time. If you understand everything in this article, you've no doubt already come up with some questions and doubts along with your new knowledge. Certainly there are other words and ideas belonging to Turbo Vision that need to be processed through your brain. In the next article, you'll learn a few more words and then you'll start coding a program using Turbo Vision. A working program, although of no great usefulness in itself. It should serve to clarify some issues and give a good running start on using Turbo Vision.

Registered Trademarks

All Borland products are trademarks or registered trademarks of Borland International, Inc.

CompuServe is a registered trademark of CompuServe Incorporated.

Products

Turbo Pascal 6.0 \$149.95

Borland International, Inc.
1800 Green Hills Road
P.O. Box 660001
Scotts Valley, CA 95066-0001

Turbo Pascal 6.0
Techniques and Utilities \$39.95

Neil J. Rubenking
Ziff-Davis Press
5903 Christie Avenue
Emeryville, CA 94508

Inside Turbo Pascal \$79/yr

The Cobb Group, Inc.
P.O. Box 35160
Louisville, KY 40232



EDITOR.PAS as released by Borland contains an error; you can't use the numbers 3 through 6. The editor won't accept them. Borland has released an appropriate fix, which can be found on the Borland Language Forum on CompuServe. This is it;

The following patch fixes the problem of not being able to enter numbers 3 through 6 into the editor of the EDITORS.PAS example shipped with Turbo Pascal 6.0.

Open EDITORS.PAS in the Turbo Editor and search for the EXISTING code. Once that is found change the EXISTING code to the CHANGE TO code. Save your changes and recompile EDITORS.PAS.

Existing (Line 421)

```
LODSW
OR      BL, BL
JE      @@2
CMP     BL, DL
```

Change to:

```
LODSW
CMP     BL, DL
```

Existing (Line 427)

```
JE      @@4
@@2:   CMP     BH, DH
JE      @@4
```

Change To:

```
JE      @@4
CMP     BH, DH
JE      @@4
```

Existing (Line 741)

```
var
  Key: Word;
```

Change To:

```
var
  ShiftState: Byte absolute
  $40:$17;
  Key: Word;
```

Existing (Line 746)

```
begin
  Key := _Event.KeyCode;
```

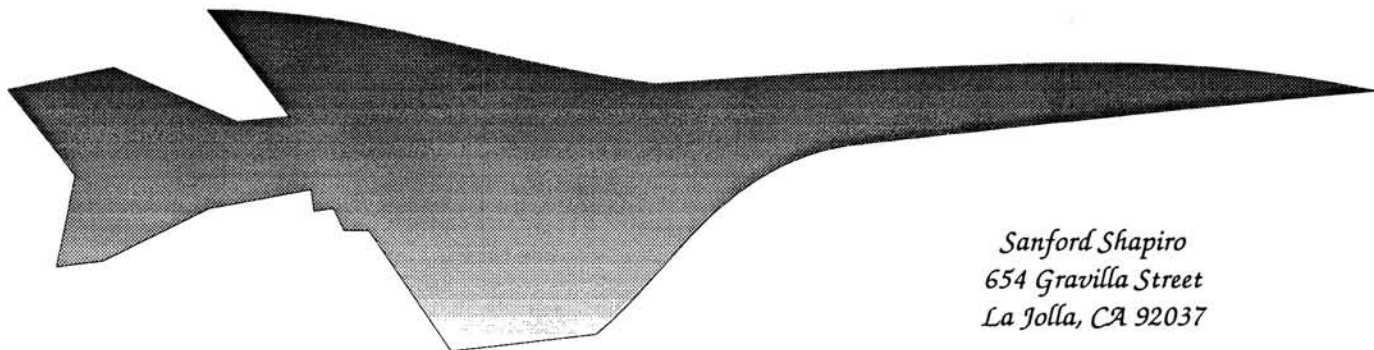
Change To:

```
begin
  if(ShiftState and $03 <> 0)
    and (Event.ScanCode >= $47)
    and (Event.ScanCode <= $51)
  then
    Event.CharCode := #0;
  Key := Event.KeyCode;
```



Desqview-386:

Stealth Comes to the PC



Sanford Shapiro
654 Gravilla Street
La Jolla, CA 92037

Desqview is a wonderful program. Desqview is a pain in the neck. Desqview lets you run several DOS applications simultaneously — each program in its own window. Desqview needs a lot of fine tuning. If you have an 80386 or 80486 computer, you need Desqview — you also need much patience. Desqview is not for the faint hearted.

I hate seeing my computer tied up while downloading files over the modem. I love being able to use Desqview and switch to another window where I can work with my word processor while the modem works in the background. And I love switching to yet another window for a data base program, looking up what I need, and seamlessly transferring the text into my word processor program. Three times — full of frustration — I put Desqview back on the shelf assuming my system was too slow or inadequate. And, each time, I found that I only needed some fine tuning to make my system purr. Some fixes I found buried in the Desqview manuals, but others I learned by calling the Quarterdeck technical support bulletin board system. I call them a lot.

Here are some examples:

Problem: You are downloading a file over your modem in a DV window. You switch to another window and your communication program grinds to a halt abruptly ending your file transfer. **Solution:** Tap the "Alt" key to bring up the "Desqview" menu. (Figure 1). Type "O" to bring up the "Open window." (Figure 2). Then type "CP" to bring up the "Change a Program" window. Type the letters that normally load your communications program, and a menu of options will appear. (Figure 3). Type the "F1" key for "Advanced options." (Figure

4). Using the TAB key, move to: "Can be swapped out" and type "N" for no; then TAB to: "Runs in background" and type "Y" for yes.

Problem: You are running Windows 3.0 in a DV window (yes, you can run Windows applications in Window's Standard mode under DV). You switch to a task in another DV window, and when you switch back to your Windows program your screen is a hodge-podge. It looks like a painting from the Museum of Modern

DESQview	
Open Window	O
Switch Windows	S
Close Window	C

Rearrange	R
Zoom	Z

Mark	M
Transfer	
Scissors	

Help for DESQview	?
Quit DESQview	Q

Figure 1

Art. **Solution:** Go into the "Change a program" menu, type "F1" for "Advanced options," TAB to "Graphics pages" and change the "1" to a "2." Many problems are solved by increasing the number of text pages or graphics pages in a window.

Problem: Data that normally scrolls smoothly across your monitor screen now is so jerky, you get a headache watching it. **Solution:** Exit Desqview and, while in your \DV subdirectory, type "Setup." Type "En-

ter" for the "Advanced Setup Menu," type "P" for the "Performance menu," and change the Foreground:Background ratio from 9:3 to 2:2.

Problem: While working in a DV win-

Open Window	
BASICA	BA
Big DOS	BD
DOS (128K)	D1
DOS Services	DS
EXCEL	EX
GRAMMATIK	GK
Info Select	IN
LEDGER H	LH
LEDGER O	LO
Manifest	MF
Memory Status	MS
MIRROR	MR
MS Windows 3 Real Mode	WR
MS Windows 3 Std. Mode	W3
PCFile	PF
Word Perfect 5.0	WP
WordStar 5.0	WS

Add a Program	AP
Delete a Program	DP
Change a Program	CP

Figure 2

ter, you try to "hot-key" into your favorite "TSR" and nothing happens. **Solution:** Call up the "Change a program" menu, type "F1" for the "Advanced options" menu, TAB to "Keyboard conflict," and change the "0" to "8."

Problem: While typing data, a diamond appears on your screen and your keyboard acts funny. You have hit the "Ctrl" key and have enabled the "Keyboard mouse" (a totally worthless feature). Hitting the "Ctrl" key will send the diamond

Options:

```

Writes text directly to screen..... [Y]
Displays graphics information..... [Y]
Virtualize text/graphics (Y,N,T).... [Y]
Uses serial ports (Y,N,1,2)..... [N]
Requires floppy diskette..... [N]

```

Figure 3

Advanced options:

```

Text Pages: 4 Graphics Pages: 2 Initial Mode:
-----

```

```

Close on exit (Y,N,blank)..... [ ]
Allow Close Window command..... [Y]
Uses math coprocessor..... [Y]
Share CPU when foreground..... [Y]
Can be swapped out (Y,N,blank): [ ]

Uses its own colors..... [Y]
Runs in background (Y,N,blank)... [ ]
Keyboard conflict (0-F)..... [0]
Share EGA when foreground/zoomed.: [Y]
Protection level (0-3)..... [0]

```

Figure 4

expanded memory emulator. QEMM creates a pool of expanded memory that can be converted, on the fly, back into extended memory when called for by programs. You don't have to reset your computer for each program.

Stealth

My dictionary defines STEALTH as: "The act of proceeding slowly, deliberately, and secretly to escape observation." QEMM386 version 6 can, in some computers, hide the ROM code and make its memory addresses usable as High RAM. The ROM code is swapped out, and the memory ad-

FEST. Optimize checks your high RAM and automatically finds the most efficient way to load your TSR programs. Manifest provides memory analysis and reporting functions. It gives hints that allow you to squeeze every last byte of memory from your system. Figure 7 shows the first meg. of memory in my computer, as reported by Manifest, without Stealth. Figure 8 shows the first meg. of memory in my computer when using Stealth. Note that in Figure 8, the ROM code from F000 to FBFF has been hidden, and the page frame area has been adjusted giving a larger area of High RAM from CA00 to EBFF.

Desqview

Desqview maps all available expanded memory into a series of DOS windows — each window looking like a separate computer to DOS. You load a different program into each window — and each program runs simultaneously. The CPU divides itself among each program (called "time slicing"), and each program runs slightly slower.

You can switch instantly from program to program, and data can easily be transferred

away and restore your keyboard. **Solution:** Exit Desqview, call up the "Setup" menu, hit "Enter" for the "Advanced Setup" menu, type "K" for "Keyboard," and then TAB to "Change System Keys? (Y/N)." Type "Y" to bring up the "Define System Keys" menu, TAB to "Keyboard Mouse" and change the "C" to "0." The Keyboard mouse has now been disabled, never to bother you again.

Problem: You install a program like Info Select or Lotus 1-2-3 in a DV window, and suddenly your system turns into a slug. **Solution:** Call your favorite BBS and download the shareware program, "TAME." Installing TAME brings you back up to speed.

Problem: While typing, you suddenly get all Caps and your keyboard acts weird. **Solution:** Add the parameter "IA" to your "DEVICE=QEMM386.SYS" command line. If that doesn't work, try adding "U8" instead.

Problem: You start Desqview and your computer greets you with beeping and then crashes. **Solution:** Add the parameter "OK" (stands for old keyboard) to your DV command line when starting Desqview (dv/ok).

The Change a program menu:

Background

Desqview 386 is a combination of two programs: QEMM386, a memory manager, and Desqview, a windowing program. QEMM was described by William M. Adney in the October, 1991 issue of RE-Mark ("On the Leading Edge"), and I will only summarize the program here.

QEMM can replace the two DOS 5.0 device drivers: "HIMEM.SYS," an extended memory manager, and "EMM386.EXE," an

	Total Memory	Total Available	Largest Available
Common Memory	17408	14002	13986
Conventional Memory	638K	631K	625K
Expanded Memory	1904K	1456K	624K

Figure 5

	Total Memory	Total Available	Largest Available
Common Memory	17408	13994	13978
Conventional Memory	670K	663K	660K
Expanded Memory	1872K	1424K	656K

Figure 6

resses are used for data. When a call is made to ROM, QEMM silently swaps the data out and the ROM code back in.

Figure 5 shows the memory status in my system without Stealth. Figure 6 shows the increase in available memory with Stealth enabled.

With Stealth enabled, each Desqview window in my system has 32K more RAM (656K instead of 624K) available. I am using a Zenith 151 (XT compatible) computer with the Intel Inboard 386/PC (a 16 MHz 80386 CPU), three megabytes of RAM, a Hercules compatible monochrome monitor, and DOS 5.0. The Stealth feature is sensitive to hardware and software, and some Zenith computers may need the Zenith version of MS DOS to use Stealth.

QEMM comes bundled with two excellent programs: OPTIMIZE and MANI-

A program that was suspended will, when called up, resume where it left off.

Fine Tuning Desqview

Fine tuning Desqview can become a way of life. Although many of the "options" and "advanced options" choices are straightforward, some are confusing — particularly the two related options: "Writes text directly to screen," and "Virtualize text/graphics."

Answering "Y" to "Writes text directly to screen" and "Y" or "T" for "Virtualize text/graphics," prevents programs from writing on top of each other on the screen, and allows programs to run in the background. Desqview sets up an area of memory to intercept screen writes. The program thinks it is writing to the screen, but Desqview keeps the data in a separate

from one window to another. If Desqview runs out of available memory, programs will be suspended in operation and swapped to disk.

Memory Area	Size	Description
0000 - 003F	1K	Interrupt Area
0040 - 004F	0.3K	BIOS Data Area
0050 - 006F	0.5K	System Data
0070 - 0492	16K	DOS
0493 - 0511	2K	Program Area
0512 - AFFF	683K	[Available]
Conventional memory ends at 704K		
B000 - BFFF	64K	Hercules Video
C000 - C7FF	32K	High RAM
C800 - C9FF	8K	Disk ROM
CA00 - DFFF	88K	High RAM
E000 - EFFF	64K	Page Frame
F000 - F3FF	16K	System ROM
F400 - F7FF	16K	High RAM
F800 - FCFE	20K	System ROM
FD00 - FDFF	4K	High RAM
FE00 - FFFF	8K	System ROM
HMA	64K	First 64K Extended

Figure 7

Memory Area	Size	Description
0000 - 003F	1K	Interrupt Area
0040 - 004F	0.3K	BIOS Data Area
0050 - 006F	0.5K	System Data
0070 - 04A0	16K	DOS
04A1 - 0583	3.5K	Program Area
0584 - AFFF	681K	[Available]
Conventional memory ends at 704K		
B000 - BFFF	64K	Hercules Video
C000 - C7FF	32K	High RAM
C800 - C9FF	8K	Disk ROM
CA00 - EBFF	136K	High RAM
EC00 - FBFF	64K	Page Frame
FC00 - FCFE	4K	System ROM
FD00 - PDFF	4K	High RAM
FE00 - FFFF	8K	System ROM
HMA	64K	First 64K Extended

Figure 8

area — allowing access to the screen only when the program is in the foreground. Virtualizing text and not graphics keeps your system from slowing.

Keyboard Conflict, another source of much confusion, determines how keyboard input is handled in a window. Each Desqview window has its own 128 character keyboard buffer. Some applications will get their keyboard input differently than others. The following information comes from the Quarterdeck Technical Bulletin #206, "Desqview and the Keyboard":

Keyboard Conflict = 1 helps with certain scripts and transfers if a program is prone to throwing away keystrokes in the type-ahead buffer.

Keyboard Conflict = 2 or **= 3** (both do the same thing) is used when a setting to "1" is not quite enough. It is for programs even more prone to throw away type-ahead.

Keyboard conflict = 4 helps with programs that steal keystrokes while in the background.

Keyboard Conflict = 5 is a combination of "1" and "4."

Keyboard Conflict = 6 or **= 7** is a combination of "2" and "4."

Keyboard Conflict = 8 helps some 3270 emulator programs and helps TSR's pop-up over a window.

Keyboard Conflict = 9 is a combination of "8" and "1."

Keyboard Conflict = A (Hexadecimal for 10) is a combination of "8" and "2."

Keyboard Conflict B = same as A.

Keyboard Conflict C = combination of "8" and "4."

Keyboard Conflict D = combination of "8," "4" and "1."

Keyboard Conflict E = combination of "8," "4" and "2."

Keyboard Conflict F = same as E.

Another option, "Runs in background," can be left blank, and Desqview will decide the choice. However, many programs do not need to run in the background, and setting this option to "N" will speed up your system. But be sure your communications program is allowed to run in the background.

Performance

Setting the number of foreground ticks and background ticks determines how the CPU will divide itself up — "time slice" — among the different tasks.

The default setting of 9:3 is probably the worst setting and causes jerky screen writing. Desqview picked this default setting to ensure compatibility with even the slowest XT computer. A setting of 2:2, or even 1:1 on some machines, will provide smooth performance.

You set the performance ratio with the "Setup" program. Or you can experiment, changing the performance ratio on the fly, using the "Performance menu." Tap the "Alt" key to bring up the "Desqview menu" and type "R" for rearrange. (Figure 9). Then type the "T" key to bring up the "Tune Performance" menu. (Figure 10). Change

Rearrange	
Move	M
Resize	R
Scroll	S
Position	123456789

Freeze	F
Hide	H
Put Aside	P

Change Colors	
Video Options	V
Tune Performance	T

Figure 9

Tune Performance	
DONE	<...+
Revive Mouse	

This Window	
When in foreground:	
Share CPU.....	[Y]
Share EGA.....	Y
Run in background..	[Y]
Allow swapping out..	[Y]

System	
Foreground ticks: [2]	
Background ticks: [2]	
Allow swapping out: [Y]	

Figure 10

the settings for "Foreground ticks" and "Background ticks."

One way to experiment is by opening a "Big DOS" window and running a speed test, like Norton's "Sl." Call up the performance menu, try another setting, and rerun the speed test. When you find the best setting, exit Desqview and run the "Setup" program to permanently fix the setting.

TAME

Well behaved programs are clever enough not to poll the keyboard when running in the background. Programs that don't behave well will bog down your system. I encountered this problem when I installed "Info Select," a free form data base program. Lately I have become more forgetful, and I now write little notes to myself. Loading Info Select in a DV window — in non resident mode — I use the computer instead of scraps of paper for my notes. Info Select keeps my notes and other countless bits of random information organized and available, and I switch to this window whenever I want to note or check something.

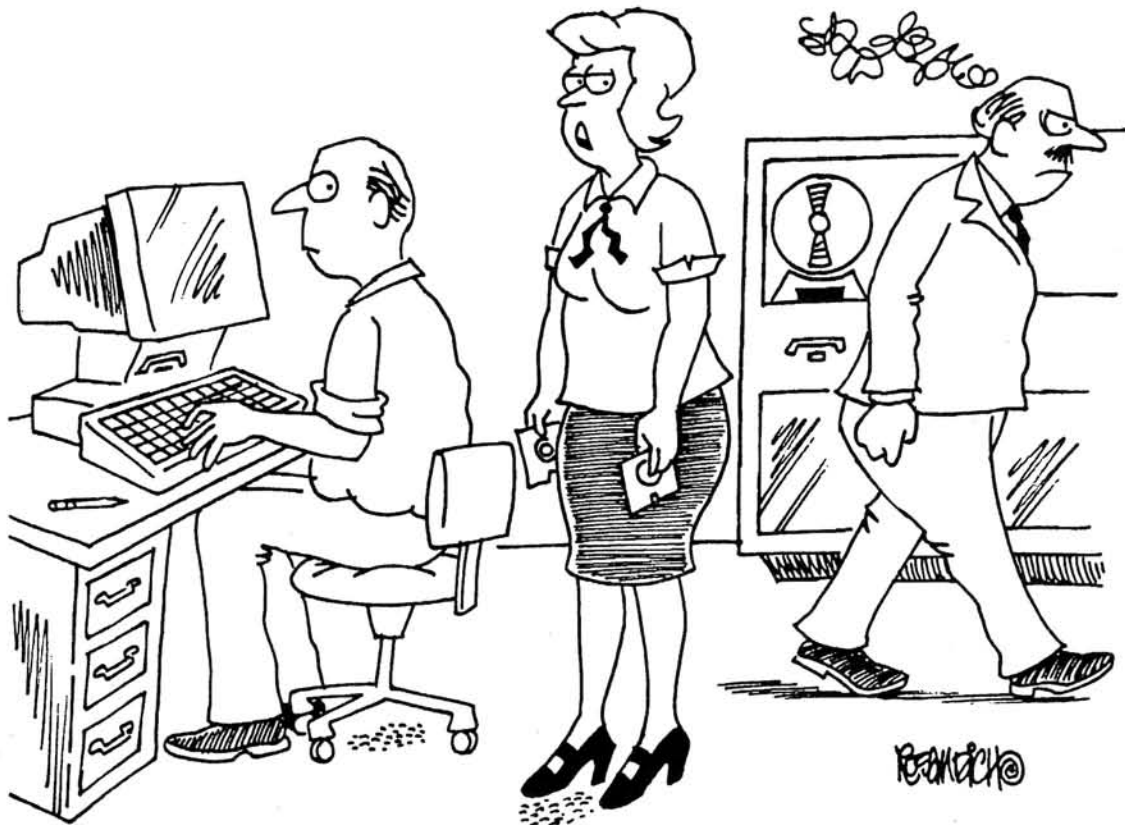
After installing Info Select, I realized that my system had changed from flying to crawling. Tame, a shareware program, solved the problem. Tame tells the CPU to ignore the keyboard polling and get on about its business. Available on most bulletin boards, you can try Tame and, if you like it, you can purchase (license) it for \$20.00.

Support

Quarterdeck provides technical support by telephone: (310) 392-9701. I don't like calling them. It is a long distance call, and I have to call during business hours when the rates are high. And either the line is busy or, if I do get through, I get put on hold. When I finally get to talk to a tech person, they usually haven't heard of my problem before or don't know anything about my type of computer.

Fortunately, Quarterdeck runs an excellent tech support BBS (310) 314-3227.

Continued on Page 46



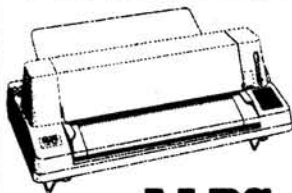
"MR. BREWSTER'S IN ONE OF HIS HIGH-TECHED-OFF MOODS."

W S Electronics

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

1106 State Route 380, Xenia, Ohio 45385

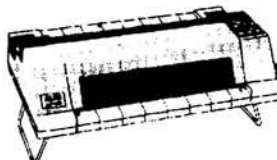
YOU AND YOUR BIG IDEAS.



- * 300 cps draft speed
- * Wide carriage
- * 24 pin printing
- * Front panel controls

ALPS Allegro 500XT™

THE ALPS ASP1600 PRINTER. COSTS VERY LITTLE, JAMS NOT AT ALL.



- Auto tear bar prevents paper waste. With the flick of a button, your fanfold paper advances to the perforation, then automatically returns to top-of-form position.
- Rugged 9-pin head delivers crisp output at 192 cps in draft mode, 38 cps in letter quality.
- Compact 9-lb. body makes for easy portability.
- Printer stand is built-in.
- Prints labels easily.
- Full Epson FX-85 compatibility.

ALPS
AMERICA

Built by popular demand.

SPECIAL

Z-415 1.5 Meg Ram for Z-248	\$100.00
Z-505 1 Meg Ram for Z-386	\$200.00
Z-315 EMS Kit for Z-159	\$ 10.00
Z-417 H. D. Controller Z-248	\$100.00
Z-317 H. D. Controller Z-150	\$ 75.00

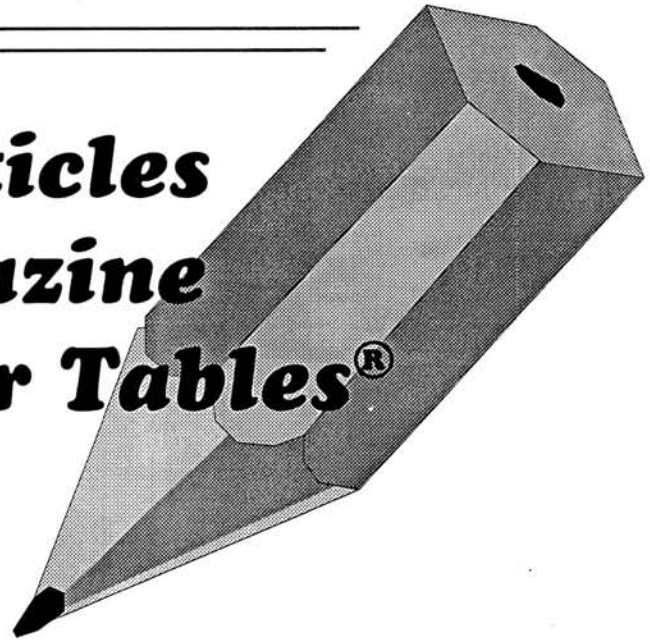
Quantities limited to stock on hand

Attention: Federal Government Offices
We stock ALL ALPS Printer Models and 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.

How to Write Articles for REMark Magazine Using the Solunar Tables®



Hope Fulscribe
(as inspired by Dr. Arthur Author)

Ha-ha, pretty good, what a clever ending. Guess that Dr. Author can speak pretty well, too. Maybe I can get a chance to... oh wait, he's stopped at the head table. Of course, it's polite to have a word with the hosts before the next presentation. Now, he's heading over this way. Let's see... there's a break in the program right now before the next speaker. If I can just get a word in with him, I bet I could...

Oh, Doctor Author! Over here, sir! Could I please have just a few minutes of your time?

Certainly, young lady, and please call me Arthur. May I sit here?

Please do, Doctor, uh, I mean, Arthur. Thank you very much. My name is Hope, and I must say I really enjoyed your presentation on The Application of Digital Logic To Analog Spouses When You Need Even More Expensive Computer Equipment. I certainly intend to try some of your ideas on my husband!

Well, (chuckling), I'm glad to be of some small, shall we say, inspiration? Sometimes it can truly be said that all we need to know is how to act like kindergartners!

Right, sir, I'm sure there's some truth in that! Let me get to the point, if I may. You are quite an experienced writer, as well as an accomplished speaker, and I've always dreamed of being able to write articles about computers. Would you be willing to share some of your secrets on how to write such articles for distinguished computer magazines?

Of course, Hope. Now there may be a few tricks of the trade, but between you and me, I'd hardly classify most of them as secrets. Where shall we begin?

WHO?

OK, first of all, who can become a writer? I mean, what does it take to write

articles like the ones you seem to turn out so easily? I never was very good in English in school, and, frankly, I wonder if I have the right "stuff" to BE an author.

Oh, stuff and nonsense!! Now there's a stereotype. You neither need to be a Shakespeare nor a techno-dweeb to write computer articles for other users to enjoy and learn from! The key is, are you interested in sharing some knowledge you've picked up with other people who might find it just as useful? Here, let me show you some little-known facts I just made up last week... hand me that napkin, would you? Let's see, about a third there, a bit less there, a SLIVER there, and, here you are, take a look at that (Figure 1)! See what I mean? Most people who write articles part-time have no greater English background or innate knack than you or I do. The difference between them and all those who don't write is that they WANT to write. Desire... that's the key here!

WHAT?

Is that all? I mean, those people are just like me, and you say they are able to write simply because they want to? How do they know what to write about? Where do they get all their ideas? I never was very creative, you know. I nearly flunked Art class because every picture I tried to paint ended up as the same sunset scene with the brook in the foreground and a purple mountain in the background.

Look, Hope, creativity is a very powerful tool in writing. But don't overlook the material you already have at your fingertips! Take another look at the applications you use most on your computer. Remember how much fun it is to show someone else how you do something that appears 'magical' to them? Readers are interested in the same stuff you do all the time,

because perhaps you've perfected a few handy shortcuts that have never occurred to them. Here, slide me another napkin, and, OK, here... we... go! Take a look at this (Figure 2); here's what new and occasional computer users are interested in reading about. Describe a few macros you've written... that's a 'how-to' article! Walk them through installation of a new package that you've found you can't live without, and you have a 'Getting Started' piece. Compare the features of two competing packages, explain why you prefer the one you do, and you've just completed a review! (Of course, the neat part about reviews is that sometimes a company notices what you've done and sends you a review copy of the next version!)

WHERE?

Oh boy, reviews, how-to's, Getting Starteds... I can do some of those things right now! But where can you find a magazine interested in printing stuff like that?

My goodness, that's an easy one. Why, REMark magazine, of course! Since you're attending this conference, I assume you own a Zenith Data Systems computer, correct?

Well, yes, of course, but...

...and you DO belong to the Zenith Users' Group, and you DO read REMark every month, don't you?

Of course, Arthur, EVERYone here does. What does THAT have to do with...

Come on now, don't tell me you've never read the little request-for-articles pieces the Editor includes from time to time! Who do you think she's trying to reach?

But... but she COULDN'T mean me! I'm not a famous author, I'm just another user!

And the Zenith Users' Group is made

Major Areas of College Study for REMark Authors

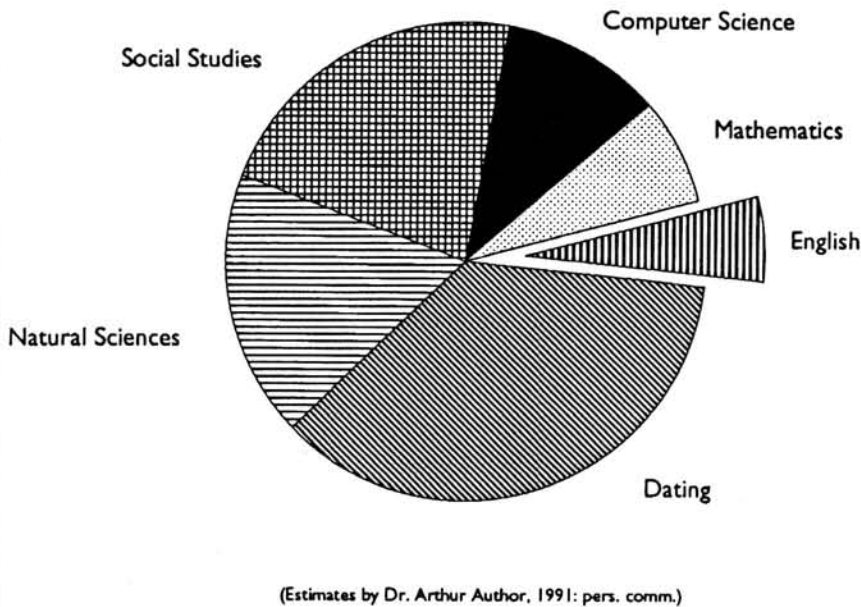


Figure 1

up of thousands of users just like you, who enjoy reading about the kinds of things their fellow members write about. Don't you see? It's a match made in heaven! REMark wants articles, you want to try your hand at writing... what more could you ask for?

WHY?

You're right, of course. I never thought about it that way. So, how does a person get themselves motivated to sit down and write? Why do people like YOU write articles?

Oh, Hope, there are lots of motives, and it really doesn't matter which one works for you, as long as you take the plunge and try it. Doubtless, some are seeking fame, others fortune, still others, a way to collect more software and hardware without risking the buzz saw of the Family Budget Committee. In fact, when guests visit my computer room, the most common question they have is, "Where Do You Get All Those WONDERFUL Toys?" The answer: from writing REMark articles, of course!!

Ah, so you were one of those in the third category, right?

Actually, that's not really how I got motivated to write the first time. I had just finished a useful, straightforward article in REMark, and it struck me that there wasn't anything particularly fancy about the writing. Waving the magazine in the air, I

proclaimed "Even I could write something like that!" My dear spouse overheard me, and responded "Well, why don't you stop scaring the dog with that thing and go DO it?" The rest is history.

HOW?

So starting out isn't that hard, you say. But this probably takes a lot of time, and you should see my schedule! I'm already going full speed from dawn till dusk, and I don't want to take time away from Bill or the kids. How do YOU get your writing done?

Now that's where I DO have a few little secrets worth sharing, I guess. Alright, you've got subject and motive; what you need now is opportunity. When can you write? Early in the morning. After 11 at night when the kids are tucked away, the spouse has nodded off on the couch during the news, the cat is outside, and no one will miss you. On the bus on the way to work. On a plane. Instead of going bowling, for goodness sake. Even during talks like the one I just gave (just kidding, just kidding)! Keep your pen and paper handy, and be prepared when opportunity strikes. 2000 words for a major article isn't nearly as much writing as it sounds, you know. Organize an outline, develop each point, stitch them together so they flow logically, and *voila* you have an article!

After 11 PM? What are you, the NightHawk? How on earth do you stay

awake?

Why, with Doctor Author's Secret High-Energy Snack, you can easily make it until 1 AM, at which point you probably have to let the cat back in anyway, so you might as well knock off for the night! Simply brew a cup of tea, and mix up a bowl containing one part raisins and one part chocolate chips. Sip, snack, type, type, type! Of course, you should keep away from cookies, even chocolate chip cookies, because crumbs and keyboards are mortal enemies! Use cookies as a reward for putting in a specified amount of time, and then eat them far from your precious keyswitches. Why tea? Well, this varies from one writer to the next, of course, but I found that too much coffee after 11 and I was feeling unimaginably proud of pages whose only prose consisted of the legend "THIS SPACE INTENTIONALLY LEFT BLANK." Once you start finding such pages funny, it's time for some shut-eye!

I've noticed that successful authors use a particular style when they write. Would it be a good idea for someone like me to write in a particular style? I mean, if it works for them, it might work for me...

No, no, forget style for now! You don't want to sound like a user's manual, you want to sound like a PERSON. As far as mimicry goes, well I think that, alas, there will never be more than one Jerry Pournelle, and that's alright. If your first attempts sound stiff and stilted, try writing conversationally. If it's easy to read, it will get the point across to an audience of learners faster. (Remember, that REMark readers are ALL learners; they're just at different stages.) If charts help, draw up a chart! If a listing of a program you've written will clarify your message, include the listing. Visual aids and a friendly tone is all the 'style' you need to worry about.

But what if it doesn't work? I mean, say you hit upon the perfect idea, and you get yourself organized, and you SOMEHOW find the time to put it all together, and you like it, and your spouse likes it, and the dog doesn't howl when you read it aloud, and you send it in... and then it gets rejected? How do you handle that?

Rejection... I wish I could say I've had no experience with that, but I have. And so has every other author I've ever heard of. Sometimes what you write or the way you say it just doesn't 'fit' into the Editor's needs at the time. A good Editor will tell you the reason for rejection, and this gives you a simple way to handle it, called the 'Oh Yeah?' approach. Meet the objections head on, determined to obliterate them with your next incredible submission, and try again. And again. And soon instead of rejection letters, you'll be receiving forms that require your Social Security Number so the magazine can PAY you. And the memory of that rejection starts to fade

away as you begin to calculate using your new mental currency (let's see now, a SuperVGA monitor is two articles, the fancy new video card is another, a Bargain Centre Special 386 is only FOUR articles...!)

WHEN?

Oops, I almost forgot; the "when" decision! How can you tell if a given day is a good day to start writing? I mean, does the Muse visit you every day? Some days I start off with the feeling that if I get home with both shoes still on I should celebrate. Do you have any special tricks to help you tell the 'good' days from the 'forget it' days?

Naturally, Hope, and I owe it all to Mr. John Alden Knight, creator of the famous Solunar Tables® many years ago. You see, Mr. Knight postulated that fish and game movements and, thus, feeding activities, would be correlated with movements of the sun and moon, much like ocean tides are generated. After years of study, he developed a table showing major and mi-

nor periods of activity for each time zone in the United States on a monthly basis, and would publish these in sporting magazines to assist fishers and hunters in planning their trips.

Oh, I see. And a period of major activity corresponds to some sort of high astral energy level within, which you then harness for effective writing. Therefore, these major activity periods are a good time for writing!

No, no, you don't see at all. A period of major activity is a good time to go FISHING. Every OTHER time is a good time to go writing!

FINAL WORDS

Um, OK, I guess I see that, Arthur. I'm very grateful for all your help, and I do hope I haven't taken up too much time. Are there any final 'words of wisdom' you could share with me?

Of course, Hope. I once heard a saying attributed to Gertrude Stein which I have since taken to heart: "I don't like

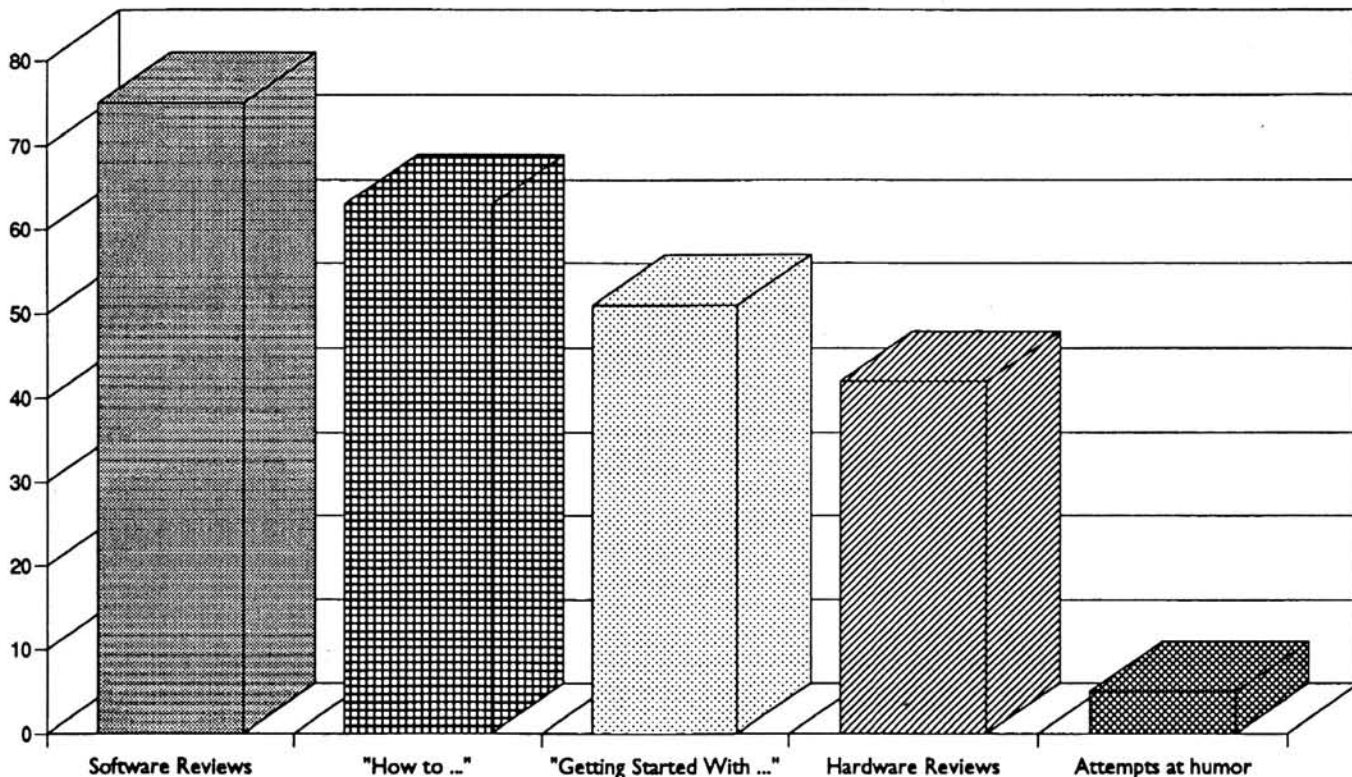
writing; I like HAVING written!" That's exactly how I feel sometimes. Writing can begin to feel like real work in the middle of a project. But once you balance that with the thrill you feel after the final spell-check when you're finally DONE, you actually find yourself going back to the keyboard and trying it again. Sometimes many weeks later, of course, but you eventually do go back!

Say, speaking of writing, you've taken quite a few notes there! Certainly, I can't be (ahem) THAT fascinating. What on earth do you plan to DO with all those notes, Hope?

Oh, ah, well, ah, gee, Doctor Author, I, oh LOOK at the time, will you? I really have to be heading back home now. It was SUCH a pleasure meeting you, and chatting with you, and, well, you'll never know just how helpful you've been to me!

(The "Solunar Tables®" are copyright 1990 by Mrs. Richard Alden Knight)

Subjects of Interest to REMark Readers



(Estimates by Dr. Arthur Author, 1991:pers. comm.)

Figure 2



DOS 5.0

Is it really improved?

Mark Haverstock
6835 Colleen Drive
Youngstown, OH 44512

Choosing the new version of DOS reminded me of shopping at the grocery store. As you walk down the aisle, you see dozens of packages proclaiming in big, bright letters: New! Improved! Free - 20% more. All these packages make great claims, but are they really better than the brand you've been using all along? Should you switch?

Those of you who were disappointed by DOS 4.0 remember well. A can of Raid was a necessary peripheral to exterminate all the bugs in this ill-starred DOS. In my case, I chose to stick with the 3.3 version. Sure, it didn't have all the bells and whistles of the newer version claimed, but it was about as bulletproof as DOS could get at that time. Later, I switched to version 4.01-only because it came installed as a freebie on my last computer.

What about DOS 5.0? Should you wait until an amended 5.01 version comes out before you buy? Sensitive to DOS 4.0's problems, Microsoft took extraordinary steps to make sure 5.0 was as bug-free as humanly possible before its release. Its beta testing program lasted nearly a year and included almost 5000 DOS users. With this kind of conscientious testing, DOS 5.0 is no doubt the most completely tested release to date.

If you still want to hedge your bets, you can try the new DOS and still have the option of returning to your old version. The setup program in DOS 5.0 first copies your old DOS files to a backup directory, creates an uninstall disk, then it installs itself. If there are any unforeseen problems, you can always revert to your old DOS. If everything is A-okay, a cleanup command deletes the old version. After cleanup, DOS will reside neatly in about 2MB of hard disk space.

As with previous versions, you may install it on floppy disk. If you own a laptop without a hard disk, this is your only option. IO.SYS, MSDOS.SYS and COMMAND.COM are necessary for your boot disk. Depending on the size of your floppy drive, you can add the other files you'll need most in day-to-day use (Table 1).

Memory and More

Ten years after the introduction of DOS, Microsoft fixed many of the problems that PC users have griped about. Among these are memory management, a useable text editor, an improved operating shell and online help.

How do you spell relief? In DOS 5.0, it's spelled M-E-M-O-R-Y. Freeing up memory in the lower 640K region while adding more features was Microsoft's major challenge. The kernel was shrunk to its pre-dos 4.0 size and it will load into high memory, just above the 1MB area. This means that almost all new PCs and many older ones can bump about 40K of DOS code out of the lower memory-leaving a program workspace of about 620K for applications. This is great news to windows users, who will notice an immediate boost in performance (Table 2).

If you have a 386- or 486-based machine with extended memory, a device driver named EMM386.SYS makes it possible to load DOS, your device drivers and TSRs into the high memory area (HMA) to give you additional space in the lower 640K of memory. But

getting the maximum amount of memory is the trick. You'll need to juggle the order of the drivers in your CONFIG.SYS file to come up with the best combination. On computers with a 286 processor you won't need to worry about this. You can load DOS high, but you can't do the same with TSRs or device drivers.

DOS 5.0 provides support for large hard disks. You no longer need SHARE or other utilities to access partitions greater than 32 meg. It will support up to 2 gigabytes directly. This is a welcome relief in a world of PCs that routinely include larger hard drives with each model change.

Formatting is now goof-proof. You can unerase accidentally deleted files and unformat disks that were accidentally formatted. This is accomplished by using the quick format feature. If you do a FORMAT/q, the format erases the FAT table and directory, but leaves the data intact. This is

Table 1
Recommended Files for DOS 5.0 on Floppy Disk

360K disk	
BACKUP.EXE	FORMAT.COM
CHKDSK.EXE	MEM.EXE
EDLIN.EXE	RESTORE.EXE
FDISK.EXE	SYS.COM
720K or larger disk	
ATTRIB.EXE	MEM.EXE
BACKUP.EXE	MIRROR.COM
CHKDSK.EXE	QBASIC.EXE
DISKCOPY.COM	RESTORE.EXE
EDIT.COM	SYS.COM
EDIT.HLP	UNDELETE.EXE
FDISK.EXE	UNFORMAT.COM
FORMAT.COM	XCOPY.EXE

Tips for Configuration

The README.TXT files have a considerable amount of information not included in the *MS-DOS User's Guide and Reference* or in the online help. I've selected several that may be helpful to many readers.

Mouse

If you have a Logitech or Microsoft mouse, you may need a new mouse driver to use the MS-DOS shell. Logitech owners should get version 5.01 or later. Microsoft mouse owners should obtain version 6.21 or later. These can be found on many BBS and online systems or can be ordered from their respective companies free or for a nominal charge. Be sure to get these upgrades before you attempt to use the DOS shell.

Zenith Computers

To use the GRAPHICS command with a Zenith computer, set the STACKS command in your CONFIG.SYS file to at least STACKS=9,256.

Phoenix BIOS

If your system has a Phoenix BIOS and doesn't work correctly, try adding the /machine:1 or /machine:8 switch to the DEVICE=HIMEM.SYS command in your CONFIG.SYS file. Example: DEVICE=HIMEM.SYS /machine:1.

HIMEM.SYS

SETUP doesn't install HIMEM.SYS if another extended memory manager is present. If you want to use this utility instead of your present one, you'll need to expand and copy the file to your hard disk.

a big time saver when you're reusing disks. If space permits, FORMAT/q saves the original FAT and directory elsewhere on the disk so you can unformat the disk easily.

MIRROR, a TSR, sets up a delete-

You can now get online assistance with DOS. There are help files available for internal commands such as FORMAT, DIR and TYPE. All you need to do is type the command followed by the /? argument or type HELP followed by the command name.

You'll get a brief, but usually helpful explanation. This is a welcome change for a traditionally user-tolerant DOS. You'll also find that the documentation is more helpful and clearly written.

Although DOS 5.0 won't run everything on every PC, it will do most of its basic functions on everything from a lowly original PC to the newest and fastest top-of-the-line 486. A new command, SETVER, assures compatibility with programs that need an older version of DOS to run properly.

DOS Shell Game

If you're tired of the C> prompt, you'll welcome the new DOS 5 shell screen. It bears a striking resemblance to Windows 3.0 and operates in a similar way. You point and click to do most file maintenance tasks, and it even lets you click on a filename and drag the icon representing the file onto

Table 2
Conventional RAM available for Windows

	DOS 4.01	DOS 5.0
Real Mode	490,640	566,656
Standard Mode	503,280	579,296
386 Enhanced Mode	550,640	617,392
Windows not running	559,888	626,646

tracking file (MIRROR.FIL) that records information used by the UNDELETE tool. This undelete feature can recover a file whose sectors haven't been reused. If no MIRROR.FIL exists, UNDELETE checks out the DOS directory and prompts you to enter the first character of a deleted filename. If MIRROR was active, it grabs the filenames from MIRROR.FIL and does the recovery chores automatically.

One of the best new programs contained in DOS 5.0 is DOSKEY, which lets you scroll through the last ten DOS commands issued and edit text on the command line. Even better, DOSKEY makes it possible for you to store command strings as hot-key macros—a quicker alternative to batch files.

Previous versions of DOS used the program EDLIN, a cumbersome line-oriented editor. Now DOS has a real full-screen editor, EDIT. It works more like a word processor, and commands are issued from an easy-to-follow menu. Editing capabilities include insert, delete, cut, paste and a full range of block commands. You can use the keyboard, mouse or both when working on files (Figure 1).

```
File Edit Search Options Help
CONFIG.SYS
DEVICE=C:\QEMM\QEMM386.SYS RAM
DEVICE=C:\DOS\SETVER.EXE
devicehigh=c:\dos\fastbios.sys
REM DEVICE=C:\DOS\HIMEM.SYS/machine:1
files=38
break=on
stacks=8,8
buffers=28
devicehigh=c:\dos\ansi.sys
DEVICEHIGH=C:\DOS\Smartdrv.sys 1024 512
REM DEVICEHIGH=C:\DOS\EMM386.EXE nocms
device=c:\hhscand.sys /a=288/i=5/d=1/h=4:88:12:16/w=183/t=15
DOS=HIGH,umb
SHELL=C:\DOS\COMMAND.COM C:\DOS\ /p

MS-DOS Editor <F1=Help> Press ALT to activate menus
```

Figure 1
EDIT — An easy to use line editor.

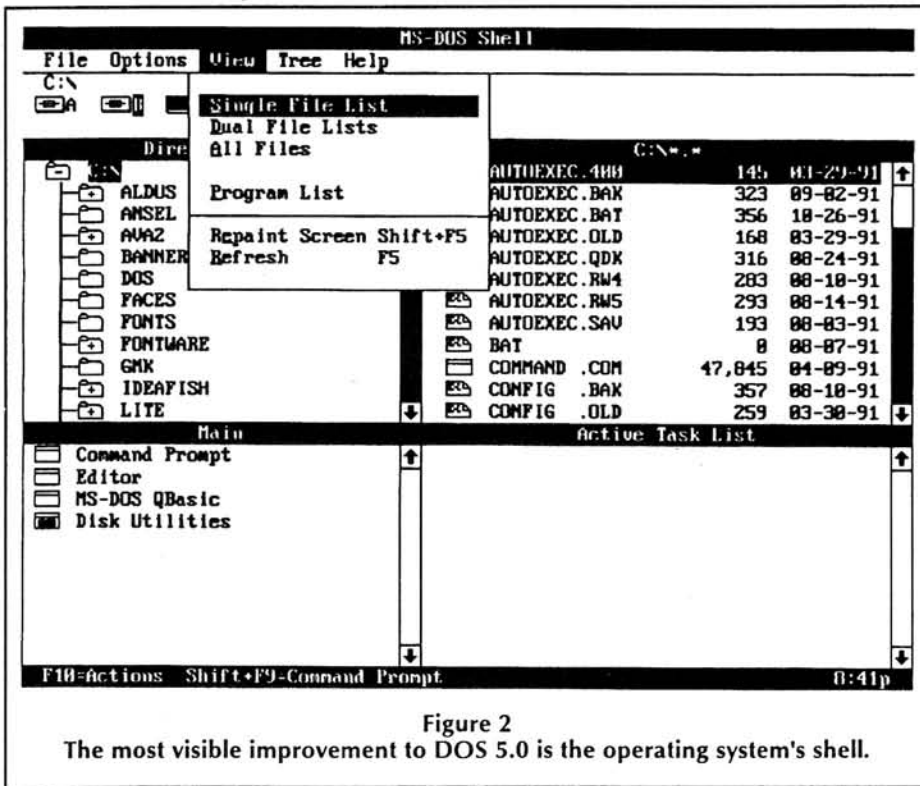


Figure 2
The most visible improvement to DOS 5.0 is the operating system's shell.

FBE

EaZy PC: EZM128 128K Memory Expansion, \$95; EZCOM Serial Port \$85; EZCOMBO Memory Expansion and Serial Port, \$145

SmartWatch: No-slot calendar/clock module. Software included. For all H/Z PC's, \$32

H/Z-148: ZEX-148 1-1/2 Card Expansion Bus, \$79.95; ZP-148 704K Memory PAL, \$19.95

H/Z-151: VCE-150 removes existing video card, allows use of EGA/VGA card, \$49.95; ZP640+ PAL modifies existing memory card to 640/704K using 256K RAM chips, \$19.95; ULTRA-PAL modifies existing RAM card to 640/704K plus 512K EMS/RAM disk, \$39.95; COM3 kit changes existing COM2 to COM3, allows internal COM2 modem, \$29.95

H/Z-100: ZMF100a modifies old motherboard for 768K memory, \$75; ZRAM-205 converts Z-205 card into 768K RAM disk, \$39

H89: H89PIP two port parallel printer interface card, \$50; Printer cable \$24; SLOT4 adds extra expansion slot to right-side bus, \$39.95

Call Or Write For Additional Information

Order by mail, phone or FAX. VISA/MasterCard/AMEX accepted. No charges for UPS Ground or USPS shipping. WA residents add 8.2% tax. Hours: M-F 1-5 PM Pacific. We return all calls left on our answering machine!

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

another subdirectory.

DOS 5's shell is flexible, and offers a good selection of options. The FILE, OPTIONS, VIEW, TREE, and HELP pull down menus cover most of the usual file maintenance commands. A disk utilities group covers other options such as backup, restore, format and undelete.

When used in its default mode, the screen divides into three portions: tree, file list and programs. The task swapper appears in the lower right when activated (Figure 2). Of course, you can always change its appearance with the VIEW menu to suit your needs. For example, you can view two pairs of tree/list windows at one time—handy for comparing the contents of two directories.

The program window lists programs ready for launching. You can use a mouse to double click on one to start it. The process of adding programs is almost identical with that in Windows. Under FILE/NEW, you select a program. Then you type in a title and a pathname and the choice installs on the menu.

The Bad News

Although DOS 5.0 is a vast improvement over its predecessors, it still has some weak links. One of these is the task swapper, found under the options menu of the DOS shell. You can have several applications open at once. In fact, you can have as many as your memory will support. But only one program can be active at a time; the operation of the others is temporarily suspended.

Task swapping is also slow—in some cases up to 25 seconds—and there is no clipboard feature to exchange material between applications.

DOS commands still aren't obvious or easy, which means you'll probably need to look some of them up in the manual or get clues via the online help feature. The DOS shell, though improved since version 4.0, is still inferior to many stand-alone products on the market. Don't retire your copies of Norton Utilities or PC Tools yet.

Another thing they didn't fix in this version was the eight-character plus three-character extension limit on filenames. You'd think that we'd be able to tag our files with some meaningful names—maybe next time.

Installing DOS 5.0

Compared to previous versions, DOS 5.0 installation is fairly painless and fully automated. You just type SETUP and it checks out your existing system configuration, including the hardware, CONFIG.SYS and AUTOEXEC.BAT files.

Next, it offers to backup your hard disk before changing anything. This is a smart precaution and besides, this job is probably long overdue. Unfortunately, the BACKUP utility hasn't improved much, so you may want to use a commercial backup utility such as Fastback or PC Tool's PCBackup instead.

DOS 5.0 makes copies of the important disk tables and the old configuration setup and stores these on a separate Uninstall floppy. Then it copies all the old DOS files into a new subdirectory. If you encounter problems at any time, SETUP lets you exit from the installation.

When the installation is complete, you'll re-boot your computer. Depending on the choices you made during installation, it will start with the DOS 5.0 shell or the DOS prompt.

Is it Time to Switch?

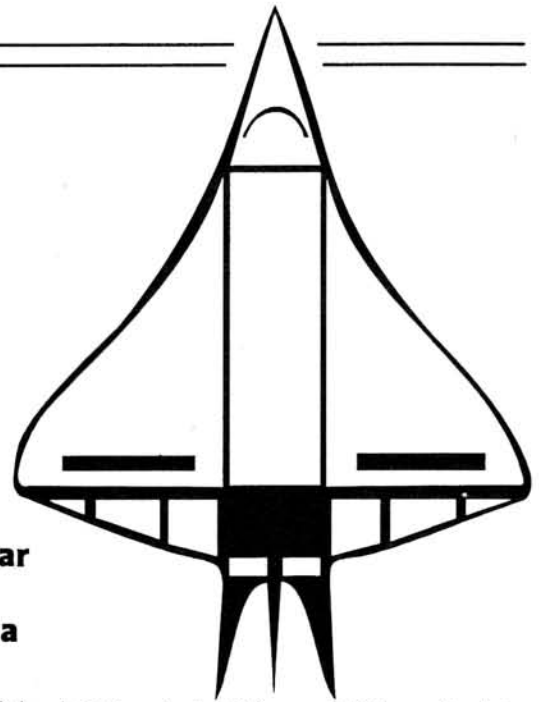
Although it doesn't contain everything we could possibly want (and what piece of software does?), DOS 5.0 has a lot to offer in the way of improved functions and features. The increase it provides in conventional memory alone is worth the price of the upgrade. Add to this a full-screen editor, DOSKEY, UNDELETE and several other goodies and you have a little something for everyone.

If DOS 5.0 sounds good to you, current upgrades carry a suggested retail of \$99. Of course, street prices will be a bit more reasonable—usually between \$60-70 at national chains and mail order outlets. You can even order it directly from MicroSoft, 800-992-3675. ✽

Computers in the Age of Travel

Using Code-Pages and Serial Printers

Daveed Shachar
P.O.B. 835
86107 Dimona
Israel



This article assumes that you are using MS-DOS 3.3 or higher, and that your monitor is an EGA or better.

Code-Paging Made Friendly

Unfortunately, one of the nicest things about code-pages is that Americans normally don't need to use them. They are difficult to learn to use, and the DOS manual, with its cryptic comments, is not much help even to people who thought they understood how to use DOS. Sentences such as, "Use a DEVICE configuration command to tell MS-DOS the hardware code page in a device and to allocate buffers for prepared code pages" do not seem very user-friendly!

On the other hand, Americans finding themselves using computers in non-English speaking countries may need to know enough about code paging to cajole the computer into printing and displaying English. Even many people living in the U.S. may need to set up a Spanish or Latin American code page.

I live in a country where the EGA and VGA cards have Hebrew installed. This means that although I can access ASCII characters up to 127 with no problems, ASCII 128-154 contain the Hebrew alphabet. This means the total loss of almost all non-English foreign letters, such as the ones needed to write German. For those of you who may be unfamiliar with the language, this includes ä ö ü Ä Ö Ü and ß.

The installation program for DOS 4.01 actually asks you what country you will be using your computer in. If you pick Germany, it will install the code page automatically. Unfortunately, it will also assume that you wish to use a German keyboard. This means relearning how to type! Several of the letters are in the wrong place for US QWERTY typists, and the special letters

listed above simply replace some of the symbols on the American keyboard we have all grown to love. Therefore, in order to type German, I had to find a way to obtain German letters with an American keyboard, and have them show up on a monitor which wants to talk Hebrew to me.

First, add the following line to the CONFIG.SYS file on your hard disk:

```
DEVICE = c:\dos\display.sys con= (EGA,437,1)
```

This command tells the computer to set up code page 437 in the EGA, which is the one for American English. After this code page has been set up by CONFIG.SYS, it will be possible to exchange it for any other code page we want. After adding this, or any other command to CONFIG.SYS, you must reboot the computer for it to take effect.

The first problem was the monitor. I decided to solve it by setting up the code-page for German, selecting the code-page for German, running my word processor, and then selecting the code-page for American English when I left the word processor. I did this with the following commands, which I incorporated into a batch file (GERMAN.BAT):

```
@echo off
```

This forces the rest of the lines in the batch file to take effect without displaying. The "@" at the beginning causes the "echo off" command not to display, as well.

```
cd \german
```

This changes to my "GERMAN" subdirectory, where I have stored the German-language version of my word processor. More about that below.

```
\dos\mode con cp prepare-((850) c:\dos\ega.cpi)
```

This prepares code-page 850 using the MODE command. It also informs the code-page to take the letters it needs from the EGA.CPI file.

```
\dos\mode con cp select=850
```

This selects code page (CP for short) 850. Unfortunately, because the ".cpi" file is for

EGA only, it will force a VGA monitor into EGA mode. This means that as long as you're using a code page, you can forget about those nice VGA fonts you had just begun to get used to.

```
ed 81
```

This calls up my word processor, together with the file I gave it when I called the batch file (GERMAN MYFILE).

```
\dos\mode con cp select=437
```

This calls up the American English code page (and puts my monitor back in VGA mode at the same time) as soon as I exit my word processor.

This batch file can be used to call up any program which needs code page 850. All you have to do is substitute the name of that program for "ED" above.

The next problem is the keyboard. It is important to find a method which will allow you to type the second language quickly. I decided to retain the American English keyboard, but to use Ctrl- and Alt-key combinations for the special letters. In order to do this, I had to create a short translation table inside my word processor. Naturally, different wordprocessors handle this differently. Basically, it's a matter of setting up a short list of macros. I decided to use Ctrl-a, Ctrl-A, Ctrl-o, Ctrl-O, Ctrl-u, Alt-U and Ctrl-s in order to obtain the letters listed above.

PC-Write has a file called ED.DEF, which includes all of the permanent macros, i.e. macros which become a permanent part of PC-Write. I added to following commands to that file:

```
001:132  
472:142  
015:148  
486:153  
021:129  
523:154  
019:225  
&A:850
```

In the table above, the number on the left is the numerical representation of the Ctrl-letter combination which calls the spe-

cial German letter. For example, 001 is the same as Ctrl-a. Ctrl-a is the key combination I decided on to call ä. The last line, \$A:850, tells PC-Write that I will be using code-page 850.

Notes: The reason I used Alt-U for Ü is that for some reason, although my keyboard (a Northgate OmniKey/102) recognizes Ctrl-u, it does not recognize Ctrl-U.

The reason I have this macro table in a special ED.DEF file in the \GERMAN subdirectory, rather than in my regular ED.DEF file is that PC-Write recognizes most of the WordStar Ctrl-letter commands. When I redefine some of them for German, I lose the possibility of using them as if they were WordStar commands. For example, Ctrl-a is a WordStar command also used by PC-Write. It moves the cursor left one word. When I am writing in German, Ctrl-a means ä, so I lose the original WordStar function.

Certain task-switching software is not happy with code-pages. I found it impossible to run GERMAN.BAT from within DESQview, and wrote to them asking about this. I received the following reply from the Technical Support Team:

"You should load your code page prior to running DESQview, as the code page affects the computer globally. You won't be able to selectively have separate code pages in different DESQview windows either way."

In other words, you can set up a batch file similar to GERMAN.BAT above, but with "DV" (runs DESQview) replacing "ED %1". You will now be able to use code page 850 within DESQview, but all the other programs you call from within DESQview will use it as well. This will cause no problems if you have an EGA and if you don't need any other code-pages (such as American 437). If you have a VGA, you will find out what DESQview looks like in EGA.

The only thing left to do is to take the German spelling dictionary sold by Quicksoft for \$20, copy it into the GERMAN subdirectory, and rename it WORDS.MAS. Now, PC-Write will not only allow you to type in German, it will even tell you when you have made a spelling mistake and offer possible corrections.

Serial Printers Made Friendly

Most PC owners find using their printers to be fairly straightforward. They plug one end of a funny-looking cable into the printer, and the other end into the parallel port at the back of the computer. They can then proceed to use their printers without any problems, pausing only occasionally in order to add another pile of tractor-feed paper, or to change a ribbon. There are several instances, however, when you will find yourself having to deal with a printer

which connects to the serial port, or a serial printer.

Serial ports are generally used to connect a modem or a mouse to a computer, but certain printers need a serial connection as well. Some instances of these:

A printer such as the Citizen 120D which was connected to an Apple computer, but now you want to move it to an IBM-compatible. The 120D has two possible interface cards, one serial and one parallel, which can be snapped out in seconds. However, only one card comes with the printer when you buy it, and the other one has to be purchased separately. You may feel it is easier to use the printer as a serial printer with the IBM-compatible, rather than shell out the money for a new card. You may also want to be able to continue using it with the Apple.

A laser printer. Certain laser printers connect to computers through their serial ports.

A portable printer. Certain printers are so light that they are perfect for travel. You may not be sure that your favorite software will recognize whatever printer you find at your disposal when you arrive at your destination. You may want to do some printing in your hotel room. The Brother EP-44 weighs only 2.5 kg (a little over six pounds) with batteries, but it can only be connected to a serial port.

A second printer. I use a Citizen MSP-10 9-pin printer for normal work, but when I want something that is really letter quality, I use the Brother, which has a 24 x 18 dot matrix print head.

In my case, I needed to be able to call up my serial printer for a certain task, but then have the computer revert to using the parallel (other) printer, when the task is completed. In order to do this, I needed a way to tell my word processor to use a serial printer temporarily. This also involves having the word processor use a different printer definition file temporarily.

Again, different word processors accomplish this task differently. For PC-Write, I first renamed my PR.DEF (printer definition file) to PR.DFE. This was a temporary measure to keep the printer definition file for the serial printer from writing over the original PR.DEF file (which is for my Citizen parallel printer). I then ran MENUPT, the printer-picker for PC-Write. This set up a new PR.DEF file, which I immediately renamed PR.BRO, to help me remember that it was a PRinter file for the BROther printer. I finally renamed PR.DFE as PR.DEF. I now had to printer definition files, one the default PR.DEF for normal word processing with the Citizen printer, and one named PR.BRO, for quality type on the Brother printer.

My computer has two serial ports, also called COM ports. As my mouse is already attached to COM1, I attached the printer

to COM2. Next, I edited the PR.BRO file. I had to tell it to print to the serial printer. The printer definition files have no way of knowing whether the printer you are using is attached to a parallel or to a serial port, so the default is parallel, which is more common, and you have to add a line if you wish to use a parallel port. This line is Alt-g (hold the Alt key down and press "g", .O (capital o) : COM2. It will look like this, highlighted: **O:COM2**. If I were attaching the printer to the other serial port, or if I only had one serial port, I would have to write: **O:COM1**

The next thing is to tell PC-Write to use the PR.BRO file, but only for the task at hand. To accomplish this, I added the following line to the batch file used to call up PC-Write with the serial printer: `set pr=!pr.bro`. Every time PC-Write goes to print, it looks at the environment string `pr=`, and if it finds the name of a printer definition file there, it uses it. The exclamation point (!) is necessary, because it tells one PC-Write control file to read another control file, and the printer definition file is a control file.

The next thing is to set the parameters for the serial port. The following command does this: `MODE COM2: 12,N,8,1,B`. The 12 is short for 1200, which is the baud rate, the rate at which the signal is sent from the computer to the printer. The N specifies the parity, in this case none. The 8 specifies the number of data bits. The 1 is the number of stop bits. Finally, The B specifies the type of retry action to take, and means to use continuous retries. This is the same as the "p" option in earlier versions of DOS.

Note: The numbers following the ":" after COM2 were chosen by me because they are the best for my particular printer. For example, using 8-bit code with my printer allows me access to the international character set. Faster printers, or printers with a larger internal buffer, would probably work better at a higher baud rate.

Finally, when I stop using the serial printer, I want to be able to go right into PC-Write and use the other printer. This means that I have to reset the environment, and have PC-Write look for the default printer definition file. In order to accomplish all of this, I added the following lines to the batch file which calls up PC-Write for use with my serial printer:

```
set pr=!pr.bro
mode com2: 12,N,8,1,B
ed %1
set pr=!pr.def
```

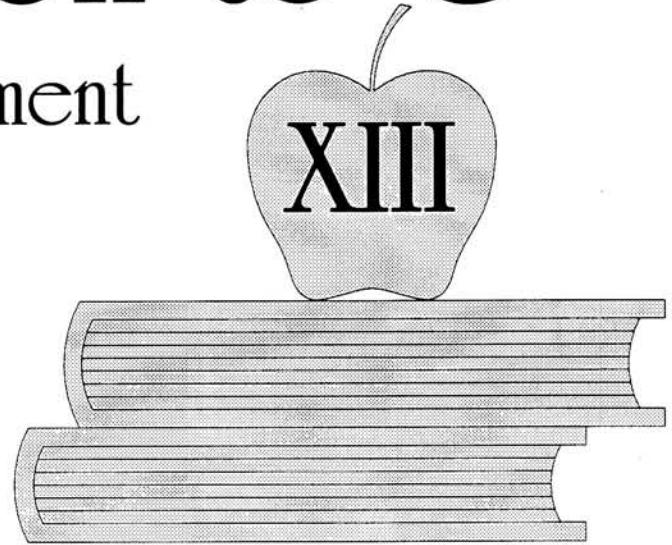
Sometimes it is necessary to type a few lines and send them to the printer, without actually entering a word processor to do so. Perhaps you want to write the name of a batch file before typing out the batch file. Say I have a ten-line long batch file called MYBATCH.BAT, and I want to type it out

Continued on Page 42

Introduction to C++

Thirteenth Installment

Lynwood H. Wilson
2160 James Canyon
Boulder, CO 80302
Copyright ©1991



The last two months I interrupted the (somewhat) orderly sequence of this tutorial to write about a problem in design which I was struggling with. It all turned out well, the client likes the program (which is nearly done), and I'm sure that writing about it helped me figure it out. I hope it helped you as well, or will in the future. This month I will return to the details of the language, including some further things to be done with classes.

Resources

The C++ book of the month is Jonathan Shapiro's "A C++ Toolkit". It's not really a toolkit, it's a book about how to build a toolkit. There are a bunch of useful classes in the book, but the most useful information is about how to develop your own tools. It's also quite well written.

There is one oddity, and that is a paragraph in the introduction which insists that you include a message giving credit in the documentation and screen display of all programs using code from this book. This paragraph is right after the one that says they take no responsibility for anything to do with the code. I think this is an unreasonable request. Seems to me that if you want your name in my programs you are going to have to guarantee your work and support it as well. And the people who write the commercial tools and support them, such as Borland and Blaise, don't ask me to advertise for them. I don't understand this, but I don't intend to use unmodified code from the book anyway, and if you change it then you are free of their copyright. As I understand it, anyway. Regardless, it's an excellent book.

Another book I read this month is Object Oriented Programming, An Evolutionary Approach by Brad Cox and Andrew Novobiliski. Brad Cox invented Objective C, an object oriented version of C which is much more like C than is C++. It's an interesting look from a different perspective. Heavy on software reuse. Mr. Cox originated the phrase "Software IC". Not important to your study of C++, but there is some useful information on design which is what I was looking for. Mr. Cox invented the idea that libraries of classes could be distributed to programmers free and the programmers would pay each time they used one of the classes in a program. It might be tough to enforce, but we do need some new ideas re distributing and paying for software components if we are to make full use of the opportunities which OOP gives us.

Classes and Data Storage

Suppose you don't want to allocate the memory space for the data in an object at compile time. To continue our example (from installment 10) of an object to hold personal data, we cannot know at the time we write the program how long each employee's name will be. The alternatives are to declare the array too long and waste memory or declare it too short and irritate all of the employees whose names are truncated. If you try to declare it just the right length you can cause both problems. Fortunately there is a nice solution, and that is to allocate the name memory for each object "on the fly" at run time, after we know how long the name will be. Here's how.

The operator `new` is used for dynamic memory allocation. It is followed by a description of a data item and it returns a pointer to a block of memory sufficient to store that data item. If `new` fails, for instance if the memory is not available, it returns 0. The pointer which `new` returns can be assigned as needed. For example, in the code below memory is dynamically allocated to store a string of 10 chars.

```
char *name;  
name = new char[11];  
strcpy(name, "John Galt");
```

First we define a char pointer called `name`. Then we call `new` for enough memory to store 10 characters (the request is for 11 to make room for the NULL), and assigned the returned pointer value to `name`. Finally we copy the string into the memory.

You may have noticed that the code above wastes a byte of memory, since it requests enough to store 10 chars and then only stores 9. We can easily change the 11 to 10, but to deal with the case where we don't know in advance, we can use the actual length of the data. Like this:

```
char *name;  
name = new char[strlen("John Galt") + 1];  
strcpy(name, "John Galt");
```

We must add one to the value returned by `strlen()` since that function does not count the NULL at the end of the string. This code finds the actual size of the string and then calls `new` to allocate enough memory for it.

Here is how the program from installment 10 works with dynamic memory allocation for the employee's name. (Figure 1)

Note that the character array to hold the name string is not defined in the class definition, instead there is a character

pointer. Then when an object of this type (an instance of the class) is created just enough memory is allocated to hold the actual data, and the pointer called name is made to point to it. Then the string is copied into the newly allocated space. We do need one buffer big enough for the biggest possible string, just to hold the names we get from the keyboard while we allocate memory for them, but the overall memory saving in a large collection of these objects will be substantial.

Note also the test on the value of the pointer name after the memory was allocated by new. If the value of name is 0 it means that new could not allocate the requested memory, and we need to know about it. Normally that means the free store is exhausted and that's fatal to the program.

Destructors

When an object (an instance of a class) is created, a creator function is executed. If you have not defined a creator for the class the compiler will supply a default creator. In the same way when an object is destroyed a destructor function is executed, and if you don't supply one the compiler will. Up until now we have had no reason to write a destructor function, but the dynamic allocation of memory gives us a reason.

A default destructor will not deallocate the memory allocated in the program above to store names, so we must write our own destructor.

There is an operator called delete which deallocates memory that was allocated with new, and makes it available for other uses. If you continue to allocate memory with new and never free any of it, eventually you will run out. Therefore, it is important to deallocate memory which is no longer being used.

The delete operator takes as its parameter a pointer value which originally came from the new operator. Deallocating memory which was not allocated by new can cause very elusive problems. Avoid it at all cost.

When calling delete to deallocate an array in memory allocated by new, we must tell delete that it is an array. This is done like this:

```
delete [] array_ptr;
```

Note that we don't have to tell delete how big the array is, just as we don't have to tell it how big other data types are. We need only include the square brackets to show that it is an array.

```
class Employee {
    char *name;
    int emp_num;
    float hr_pay;
public:
    Employee(void) { get_emp_data(); }
    Employee(char *new_name, int new_num, float
              new_pay);
    void get_emp_data(void);
    void write_emp_data(void);
    ~Employee(void) { delete [] name; }
};
```

The destructor has the same name as the class, preceded by a tilde (~). Destructors may not take arguments or return a value. They are called automatically when an object goes out of scope but unlike constructors, they cannot be called explicitly. There can only be one destructor for a class.

The destructor does not do anything useful in the little example program, but if we were to use the Employee object in a bigger program it would be needed. For example, if we created one or more Employee objects in a function, their destructors would be called when the function ended and the objects went out of scope. If there was no destructor for Employee objects, each time the function ran more memory would be allocated and never

```
#include <conio.h>
#include <iostream.h>
#include <string.h>

class Employee {
    char *name;
    int emp_num;
    float hr_pay;
public:
    Employee(void) { get_emp_data(); }
    Employee(char *new_name, int new_num, float new_pay);
    void get_emp_data(void);
    void write_emp_data(void);
};

Employee::Employee(char *new_name, int new_num, float new_pay)
{
    name = new char[strlen(new_name) + 1];
    strcpy(name, new_name);
    emp_num = new_num;
    hr_pay = new_pay;
}

void Employee::get_emp_data(void)
{
    char name_buffer[81];

    cout << "Name: ";
    cin >> name_buffer;
    name = new char[strlen(name_buffer) + 1];
    if(name == 0) {
        cout << "<< New could not allocate memory in Employee(). >>";
        exit(-1);
    }
    strcpy(name, name_buffer);

    cout << "Number: ";
    cin >> emp_num;

    cout << "Hourly pay: ";
    cin >> hr_pay;
}

void Employee::write_emp_data(void)
{
    cout << name
         << ", employee number "
         << emp_num
         << ", makes $"
         << hr_pay
         << " an hour.\n";
}

void main(void)
{
    Employee emp[5] = { Employee("Tom Paine", 0, 123.45),
                      Employee("John Galt", 2, 321.00)
                    };

    int n;
    while((n = getch() - '0') >= 0 && n < 5)
        emp[n].write_emp_data();
}
```

Figure 1

freed. Eventually the program could run out of memory and crash.

Inheritance

Hierarchies of classes is a principal fundamental to Object Oriented Programming. Inheritance allows us to create a class which inherits the attributes of an existing class and then add new data and functions to it. The code from the base (ancestor) class can be used in all of its derived (descendent) classes, but the code only appears once in the program. This simplifies reuse of existing code and improves the structure and readability of a program. Such programs are much easier to write, maintain and modify. These advantages become particularly evident in the case of large programs.


```

#include <conio.h>
#include <iostream.h>
#include <string.h>

class Employee {
    char *name;
    int emp_num;
protected:
    float hr_pay;
public:
    Employee(void) { get_emp_data(); }
    Employee(char *new_name, int new_num, float new_pay);
    void get_emp_data(void);
    void write_emp_data(void);
    ~Employee(void) { delete [] name; }
};

Employee::Employee(char *new_name, int new_num, float new_pay)
{
    name = new char[strlen(new_name) + 1];
    strcpy(name, new_name);
    emp_num = new_num;
    hr_pay = new_pay;
}

void Employee::get_emp_data(void)
{
    char name_buffer[81];

    cout << "Name: ";
    cin >> name_buffer;
    name = new char[strlen(name_buffer) + 1];
    strcpy(name, name_buffer);

    cout << "Number: ";
    cin >> emp_num;

    cout << "Hourly pay: ";
    cin >> hr_pay;
}

void Employee::write_emp_data(void)
{
    cout << name
         << ", employee number "
         << emp_num
         << ", makes $"
         << hr_pay
         << " an hour.\n";
}

class New_Employee : public Employee {
public:
    New_Employee(void) : Employee() {}
    New_Employee(char *new_name, int new_num, float new_pay) :
        Employee(new_name, new_num, new_pay) {}
    void raise(float amount_of_raise) { hr_pay += amount_of_raise; }
};

void main(void)
{
    New_Employee emp("Tom Paine", 0, 23.45);

    emp.raise(1.5);
    emp.write_emp_data();

    New_Employee emp1;

    emp1.raise(2.75);
    emp1.write_emp_data();
}

```

Figure 2

In this example we create a class that inherits all the attributes of the Employee class and in addition includes a function to give the employee a raise. Here's how. (Figure 2)

The first thing to notice is that I added the word protected: before the declaration of hr_pay in the declaration of the original Employee class. Had I not done so, that data would have remained

private (the default for classes) and thus would have been accessible only to functions in the original Employee class and not to functions in classes derived from Employee. Making it protected instead makes it available to functions in the derived classes as well, but the data is still not accessible to other code outside the classes Employee and New_Employee. This is something worth thinking about whenever you write a new class which may someday be used as a base for another class. There is no difference between private and protected as far as the class itself is concerned, the difference is in how the data and functions can be used by classes derived from this class. The best time to decide which data and functions should be private to the class and which should be accessible to any future derived classes is now, when you are writing the original class. You can always change it later, but now is when you will know the most about the class and its requirements. Changes made later will stand a higher risk of creating bugs.

In the declaration for the class New_Employee, the first line indicates that it is derived from Employee with the colon and the phrase public Employee. The word public here causes the data and code from Employee to have the same restrictions (public, private, protected) in New_Employee that they had in Employee, the base or ancestor function from which they were derived. Thus the public functions in Employee will be public in New_Employee and the protected data in Employee will be protected in New_Employee. Had I used this line instead:

```
class New_Employee : private Employee {
```

all the public and protected data and functions in Employee would have been private in New_Employee. The private data and functions (if any) in Employee would have been accessible only inside Employee in either case.

Note that the base function from which you are inheriting can be declared public or private but not protected.

The result of the declaration of New_Employee is that it inherits the private data from Employee, but only the functions of Employee can use it. New_Employee inherits the protected data from Employee and functions from both classes can use it. New_Employee inherits the public functions from Employee and they can be used both inside New_Employee and outside, in the rest of the program. And the new functions in New_Employee can be used inside or out since they are declared public.

The next thing to deal with is the constructors for New_Employee. Note first that the raise() function added in New_Employee doesn't require any new work on the part of the constructor, so the old constructors used in Employee will do fine. However the constructors from Employee will not be called automatically when an instance of New_Employee is created so we must call them explicitly. We do this by writing constructors for New_Employee which call the constructors for Employee. The first is the constructor without arguments.

```
New_Employee(void) : Employee() {}
```

Thus when an object of class New_Employee is created with no arguments, the first thing it does is call the constructor for Employee which takes no arguments. The curly braces with nothing inside represent the empty body of the New_Employee constructor. If there were work for the New_Employee constructor itself to do, that code would go between the curly braces.

Likewise in the case of the constructor which takes arguments.

it works, just what it does. The new ca-

```
New_Employee(char *new_name, int new_num, float new_pay) :  
Employee(new_name, new_num, new_pay) {}
```

If an instance of the class `New_Employee` is created with arguments of these types the first thing it does is call the `Employee` constructor and pass along the arguments. Once again, any work specifically for the `New_Employee` constructor would go in the braces. Note that the argument list for the `New_Employee` constructor does not have to be the same as that for the `Employee` constructor. The call to the `Employee` constructor must of course conform to its argument list.

Advantages of Inheritance

I have always tried to reuse old code in new programs, with varying degrees of success. Sometimes a function would be usable with no change, and that worked out fine. This happened more often as I became conscious of the kinds of functions I found the most reusable. I noticed that simple functions which were designed to be as general as possible were easiest to reuse, and I started to pay more attention to these factors when I wrote new functions. The principals of structured design, particularly the methods of evaluating the cohesion of a function and the coupling between functions, helped a great deal. (See *The Practical Guide to Structured Systems Design* by Meilir Page-Jones.)

The most common occurrence was that I'd find a function which was almost exactly what I needed, but I had to make just a few little changes to it. This often turned out worse than writing a new function from the beginning. One problem is that when you change the code in any way you must test it as thoroughly as you did when you first wrote it. It's hard to convince yourself that this testing is necessary, especially when it was such a tiny change. Another problem results from the fact that the internal workings of the function are often not documented very well, and you have to figure it out from the code before you can work on it.

The creation of a new class by inheriting code and data from an existing one goes a long way toward solving these two problems. Of course, if you find a class which does exactly what you need, you can use it directly in the new program just as before. If the class has much of what you need but requires additional capability you need only create a new class derived from the old one which inherits all the capability of the old class, and add to it whatever else you require. This is quite different from modifying an old function, since the data and code from the old class are not changed in any way and thus will not need to be tested. You don't even have to know how

pabilities must be tested, and you must be alert to possible unexpected interactions between the old and the new, but it is a lot simpler than changing part of an existing function.

But what if you find a class which is almost what you need, but a part of what it does is wrong for your application? Then you can override a function in the base or ancestor class by defining a new function of the same name in the derived class. Figure 3 is an example.

Note that `emp.func()` in `New_Employee` overrides the function with the same name in `Employee`. When we create an instance of `New_Employee` called `emp` and call `emp.func()` we get the one defined in `New_Employee`, which prints `New_Employee.func`. In order to call the `func()` from the class `Employee` we must use the `::` operator to specify the scope within which the function exists, which in this case is the class `Employee`. This is done by calling `emp.Employee::func()`. The ancestor class `Employee` is invoked with the dot as though it were a member of `New_Employee`, and the `::` says that we mean that `func()` which is visible from within the class `Employee`.

Another useful point is that you can derive a class from another without having the source code for the ancestor class. You need the declaration for the class and the .obj file but you do not need the source for the functions. This offers an opportunity for a developer to sell software components which you can use anyway you like, without giving away his source.

Relationships

There is a special relationship between base and derived classes. A member of a derived class can be treated as though it were a member of its base class, for some purposes. For instance, if class `D` is derived from class `B`, the address of an object of class `D` can be assigned to a pointer to objects of class `B` without a type cast. This is possible because objects of type `D` are, in a sense, also of type `B`. Here is an example which uses the classes from the next to last example.

```
void main(void)  
{  
    New_Employee emp("Tom Paine", 0, 23.45);  
    Employee *emptr = &emp;
```

```
#include <iostream.h>  
  
class Employee {  
public:  
    void func(void) { cout << "\nEmployee.func"; }  
};  
  
class New_Employee : public Employee {  
public:  
    void func(void) { cout << "\nNew_Employee.func"; }  
};  
  
void main(void)  
{  
    New_Employee emp;  
  
    emp.func();  
    emp.Employee::func();  
}
```

Figure 3

```
emp.raise(1.5);  
emptr->write_emp_data();  
}
```

The pointer `emptr` is declared as a pointer to objects of class `Employee`, and then assigned the address of `emp` which is an object of type `New_Employee`. This is possible because `New_Employee` is derived from `Employee`. We then use the pointer `emptr` to call the function `write_emp_data()`. This is possible because that function comes from the class `Employee`. We could not execute the function `raise()` like this:

```
emptr->raise(1.5);
```

because `raise()` is not a member of the class `Employee`, and `emptr` is declared as a pointer to class `Employee`. Even though we have made it point to an object of class `New_Employee`, it only knows about the members of class `Employee`.

Note that `emptr` does not need to know what class `emp` belongs to, only that it is derived from `Employee`. Thus it is possible to create objects at run time which are of several different types depending on the incoming data. We can then manipulate them with a pointer (or an array of pointers) to a common ancestor type without knowing or caring what class they belong to, as long as the functions which we need to use are derived from the ancestor class. We will see this idea again soon.

Sources

- A C++ Toolkit by Jonathan S. Shapiro, Prentice Hall, 1991.
- Object Oriented Programming, An Evolutionary Approach — 2nd edition by Brad Cox and Andrew Novobiliski, Addison-Wesley, 1991.
- The Practical Guide to Structured Systems Design by Meilir Page-Jones, Yourdan Press, 1988. ✱

DR DOS 6.0



Terry W. Wilk
61201 CR 687 East
Dowagiac, MI 49047
Copyright ©1991 Terry W. Wilk

Introduction

This discussion is written for the person familiar with DOS who is interested in the comparisons and differences between DR DOS 6.0 (Digital Research DOS) and MS-DOS 5.0 (Microsoft DOS). Terse overviews rather than in depth details of any single command will be presented.

At the end of this article is a list of products discussed, prices, and other sources of information.

I have always used MS-DOS in one form or another. I am a newcomer to DR DOS. The company I work for, Appraisers Choice Incorporated, got a copy of DR DOS and was interested in how it ran, as some of our clients were talking about DR DOS.

Needless to say, MS-DOS is so entrenched that no one was jumping out of their seats to install DR DOS. I was very curious about it after reading the manual; I installed it, and seriously used it.

The information in this article comes from comparing the DR DOS and MS-DOS manuals, sometimes running the functions side by side, and sometimes by experience. I have tried to be correct in my comparisons, yet keep with a terse format.

Rather than spelling out "DR DOS" and "MS DOS", the shorter "DR" or "MS" will be written with an assumed "DOS".

In discussing commands, the following formats are used:

COMMAND who: Use. Notes.

COMMAND who/who: Use. Notes.

COMMAND who, COMMAND who:
Use. Notes

COMMAND who: Discussion.

The "COMMAND" will be in upper case. "who" will be "dr" or "ms" or "dr/ms" for when both DOS' share this command. "COMMAND who, COMMAND who:" will be used where the two DOSs have different names for the same function. "Use" will usually be a one sentence comment on what the command does. "Notes", if any, will detail features or switches that one DOS has over the other. I will normally not go over any shared switches between the two DOS'. As an example, I will not say that both DOSs DIR commands have a /w switch for wide format. "Discussion" will be a small discussion on the command.

What is DR DOS?

DR DOS is a disk operating system for PCs using 8086, 8088, 80286, i386, or i486 microprocessors. It is single user DOS created by Digital Research. DR DOS has been around for many years and people have come to call it Doctor DOS. The company pronounces the two letters 'D' 'R' when referring to the product.

You do not need MS-DOS to run DR DOS. DR DOS is not an add-on, it is meant to be the DOS. Programs that run with MS-DOS will run with DR DOS as it is directly DOS compatible.

DR behaves like MS by using familiar commands such as DEL, DIR, COPY, MORE, RD, etc., right down to many command switches. MS CONFIG.SYS and BATCH file commands are also mimicked. AUTOEXEC.BAT is called during boot up. In using DR, you will not have to relearn, for

example, COPY, DEL, CHKDSK, FORMAT, XCOPY, CONFIG.SYS commands, or batch file commands.

The difference with DR 6.0 is in its extra commands and features that are not found or expanded with respect to MS 5.0. Some of them are DOSBOOK, DISKOPT, FILELINK, LOCK, MEMMAX, PASSWORD, SETUP, SSTSOR and XDEL. CONFIG.SYS commands CHAIN, GOSUB, GOTO, and SWITCH. Batch file commands GOSUB, and SWITCH. The cost of DR DOS is the same as MS-DOS.

Digital Research

Digital Research, founded in 1975, creates system software for microcomputers. In the early days of microcomputers you may remember the CPM operating system. This was from Digital Research. Multiuser DOS 386 is another operating system currently sold by Digital Research.

You may have seen in recent Novell ads that they are recommending DR to be used with Novell networks. The reason for this is that Novell and Digital Research merged.

Manuals

Both DOSs come with one large manual of over 660 pages. MS came with an extra "Getting Started" booklet and DR came with a booklet on ViewMax. Later, DR sent me by mail a booklet "Optimization and Configuration Tips" and an upgrade disk.

I did not find any major problems with either manual. However, the MS manual

appeared to have more information in it (it had smaller type, greater number of lines per inch, smaller left margin). If you crave every bit of information you can get on DOS, you may prefer the MS manual's style. If you loath reading manuals, you may prefer the DR manual's style.

INSTALL, SETUP and UNINSTAL

SETUP ms: Used to install MS. To install you run SETUP from the DOS prompt. MS can install to floppies or to the default boot partition. There are very little set up options to deal with.

INSTALL dr: Used to install DR. To install, you boot your computer with the DR start up disk in drive A. Once up, INSTALL is ran from the AUTOEXEC.BAT file. You can install to floppies or to the first hard disk partition. There is simple context sensitive help and you can choose default configurations or fine tune your system during install.

SETUP dr: You use SETUP to change many of the features of DR anytime after you have installed it. Some of the things you can set are, relocating DOS, task manager, ram drive, disk cache, dos shell, drivers, and security. However, to take full advantage of more powerful features you will need to edit your CONFIG.SYS and AUTOEXEC.BAT files yourself.

The DR package states that you need the correct disks for your drive A, either 1.2MB, 5.25-inch or 720KB, 3.5-inch. By redeeming a coupon from the package you can get the software on 360KB, 5.25-inch disks.

One matter I noted about DR install is that it will install on the first hard disk partition, which may or may not be your default boot partition. DR will tell you of this and ask you to change your boot partition with FDISK after installing. MS install (I used the MS-DOS upgrade package) did not have this peculiarity and would install to any partition set as the default boot by FDISK.

Both DOS' will take a bit over 2 Megs of disk space after install. Unused DOS programs can be deleted later to save space. MS install disks have compacted files that must be unpacked for use. DR does not have compacted files on the install disks. Both DOSs install are relatively painless and allow you to uninstall.

HELP

HELP ms: The HELP command without parameters gives you a list of the commands with a one or two line description of that command. HELP with the name of a command as a parameter, as in "HELP FORMAT", will give you a brief synopsis of the command's parameters and switches.

HELP dr: You access DR help by the HELP command (a batch file that calls DOSBOOK.EXE) with or without a topic

parameter. DOSBOOK is a topic-oriented help system. You can move around in the help system by pressing various keys or by highlighting a word or phrase. DOSBOOK has detailed information on DOS basics, troubleshooting error messages, and device drivers. I do not know why, but I could not find any help on CONFIG commands and batch file commands in DOSBOOK. Also, it was slow at times. Other than that, DOSBOOK was good.

All DR commands support a /h help switch (example: format /h). This gives a brief synopsis of the command's parameters and switches. Only a few commands in MS support a help switch.

Advanced Memory Management

A few memory terms will be briefly explained before going over memory management.

BASE: First 640K bytes of ram.

LOWER: Bottom part of base memory where DOS and device drivers are usually loaded. That is, DOS loaded normally and not loaded into upper or high memory areas.

UPPER: From 640K to 1M byte of ram.

EXTENDED: From 1M byte to end of ram. 286 or higher.

HIGH: First 64K bytes of ram starting at 1M. 286 or higher.

EXPANDED/EMS/EEMS: 8086 and higher. Method that gives DOS programs access to more memory through a 64K page frame that is normally kept in upper memory. You usually need a special memory board for EMS, but 286s and higher can emulate it in extended ram.

XMS (Extended Memory Specification): A standard interface needed by some programs to use extended memory. 286 or higher.

VCPI (Virtual Control Program Interface): A standard interface to allow a DOS extender program to coexist with another program in protected mode. 286 or higher.

DR has what is called MemoryMAX, which stands for a group of device drivers and programs used for advanced memory management.

HIMEM.SYS ms: 286 or higher. This driver lets you load DOS into high memory and supports XMS. Some of the switches you can use are, int15 support, A20 handling, specific computer brand support, and shadow ram on/off.

HIDOS.SYS dr: 286 or higher. This driver lets you load DOS into high memory and supports XMS. It supports EMS 4.0 upper memory blocks (EEMS) which will allow a 286 to load TSRs and drivers into the upper memory area. Extra support for a 286 running with certain chip sets is provided (Chips and Technologies). Some of the switches you can use are, relocation of extended BIOS data area, scan/include/exclude specified areas of ram for UMBs

(upper memory use), extend base memory into video memory, shadow ramming ROM, and settings for chip sets, EMS, or RAM. Note that some of the switches are only valid for the 286 with special chip sets.

EMM386.EXE ms: 386 or higher. This driver lets you use the upper memory area and emulates EMS in extended memory. HIMEM.SYS must be loaded with EMM386.EXE. EMM386.EXE must run in base ram. Some of the switches are, set EMS page frame, exclude/include upper memory areas, EMS size, and memory access.

EMM386.SYS dr: 386 or higher. This driver lets you use the upper memory area, emulates EMS in extended memory, supports XMS, supports VCPI, and can load DOS into the high memory area. DR does not need HIDOS.SYS loaded. EMM386.SYS can run in upper memory. Allows you to take 64K or 96K of unneeded VIDEO ram and add it to your base ram. This is for MDA, Hercules, or CGA/EGA/VGA and only with programs running in text modes. Some of the switches are, relocation of extended BIOS data area, EMS page frame, exclude/include upper memory areas, EMS size, extend base memory into video memory, and shadow ramming ROM.

MEMMAX dr: Allows you to enable/disable lower, upper, and video memory. Useful for turning off these areas of memory for programs that don't expect to find it open for use.

The main difference of DR over MS is that many of DRs drivers and programs can be loaded into upper memory. DR can expand the base ram into unused areas of VIDEO ram. You can actually have more than 640K of free system RAM for any program that runs in text mode. You can use MEMMAX at any time to shut off one or more of the DR advanced memory features for running a program that has problems with these features.

Both DOS' manuals mention that setting your system up for advanced memory management can be an experiment using trial and error. Depending on what software you use, computer type, and how far you want to push things, setting up for advanced memory usage can be quite easy or time consuming.

New Commands for CONFIG.SYS

DR has more commands for use within CONFIG.SYS. With MS you may be using extra CONFIG.SYS/AUTOEXEC.BAT enhancing software so that at boot time you could choose different configurations.

For example, if you are not using your hand scanner, then you do not need that driver loaded. The best configurations for running with and without task switching are different, so the best configuration can be chosen at boot time, depending on what you will be doing.

DR and MS share the following CONFIG.SYS commands. Except as noted, all switches and parameters appear to be the same between DR and MS for that command.

BREAK dr/ms: Changes how frequently DOS looks for CTRL-C/CTRL-BREAK.

BUFFERS dr/ms: Set up number of disk buffers. MS has a secondary buffer cache that you can specify for when you do not run SMARTDRV.SYS.

COUNTRY dr/ms: Set up for foreign countries.

DEVICE dr/ms: Load device driver.

DRIVPARM dr/ms: Set disk drive attributes. MS has an /i switch for electronically compatible 3.5-inch drives.

FCBS dr/ms: Set number of files that can be opened under the older FCB method. DR has a parameter for protected automatic closure value.

FILES dr/ms: Set maximum number of files allowed open at the same time. DR default is 20, MS is 8.

INSTALL dr/ms: Load TSR programs.

LASTDRIVE dr/ms: Set last drive letter.

REM dr/ms: Used for putting in comments. DR lets you use ';' in place of REM.

SHELL dr/ms: Specifies DOS command processor.

DR and MS share the following functions, but with different command names.

HIDEVICE dr, DEVICEHIGH ms: Load device driver into upper memory.

HIDOS dr, DOS ms: Load DOS into upper/high memory areas. MS has an extra parameter to turn on upper memory links.

HIINSTALL dr, LOADHIGH ms: Load TSR program into upper memory.

The following CONFIG.SYS commands are found only in DR.

? dr: A '?' at the start of a statement will cause DR to ask if you want the following command done. You may also specify your own prompt text. If you do not specify your own prompt, then the line as it is will be shown on the screen along with a prompt asking you to press 'Y' or 'N'.

:LABEL dr: Defines a label of up to 8 characters of your choice. Used by GOTO, GOSUB, and SWITCH.

CHAIN dr: Transfer control to another configuration file.

CLS dr: Clear the screen.

CPOS dr: Position the cursor at row, column (1 based).

ECHO dr: Display text on the screen.

EXIT dr: Exit from the config.sys file.

FASTOPEN dr: Set the number of files the fastopen table can hold. DR is different from MS in that with DR this command is internal. MS has a fastopen.exe TSR that takes up around 11K plus the table. MS allows one drive to be set where DR includes all drives.

GOSUB dr: Jumps to a label and continues processing until the next RETURN. When RETURN is found, processing jumps

back to the first label and continues.

GOTO dr: Jumps to label and continues processing.

HIBUFFERS dr: Place the disk buffers in high memory. Note that MS will load the buffers into high memory if DOS=HIGH is in effect.

HISTORY dr: Turns on extended command line editing. MS has doskey.com TSR which takes up 3K plus table. HISTORY is internal to DR.

RETURN dr: Used to trigger a return from GOSUB or SWITCH.

SET dr: Used to set master environment variables.

SWITCH dr: Used to jump to a subroutine given number key pressed by user. The allowed numbers are 1 through 9 so that up to 9 subroutines can be called.

TIMEOUT dr: Used to set a time limit (in seconds) on the response to '?' or SWITCH commands. When time is up, the '?' is skipped or the first SWITCH jump is done.

Batch File Commands

DR and MS share the following batch file commands. Except as noted, all switches and parameters appear to be the same between DR and MS for that command.

@ dr/ms: Command prefix. Used to stop standard output of command from being seen.

:LABEL dr/ms: Defines a label of up to 8 characters of your choice. Used by DR for GOTO, GOSUB and SWITCH. Used by MS for GOTO.

CALL dr/ms: Calls another batch file, then processing continues with that file, then processing continues with the file the CALL was made from.

ECHO dr/ms: Set ON/OFF display of commands or to display a message.

FOR dr/ms: For repeating a command on a set of files. This command is also valid to use at the DOS prompt.

GOTO dr/ms: Jumps to label and continues processing.

IF dr/ms: Used for conditional processing. DR has an extra condition called DIREXIST.

PAUSE dr/ms: Pauses processing until a key is pressed.

REM dr/ms: Used for putting in comments or displaying a comment. DR lets you use ';' in place of REM.

SHIFT dr/ms: Used for accessing more than 10 command line replacement variables.

The following batch file commands are found only in DR.

GOSUB dr: Jumps to a label and continues processing until the next RETURN. When RETURN is found, processing jumps back to the label and continues.

RETURN dr: Used to trigger a return from GOSUB or SWITCH.

SWITCH dr: Used to jump to a subrou-

tine given number key pressed by user. The allowed numbers are 1 through 9.

Miscellaneous DOS Commands

DR and MS share the following DOS commands. All of the DR commands support a /h help switch, MS does not in most cases. Except as noted, all switches and parameters appear to be the same between DR and MS for that command. Other DOS commands not listed here are discussed later.

APPEND dr/ms: Lets programs find files outside of current directory without specifying a path. The /e switch has an ON|OFF parameter in DR.

ASSIGN dr/ms: Change physical drive (letter) to be used as logical drive (letter). DR switch /a and MS switch /status do the same thing.

ATTIB dr/ms: View and set file attributes. DR has a /p switch for pausing long displays.

BACKUP dr/ms: Used for backing up files.

CHCP dr/ms: Select code page.

CHDIR (CD) dr/ms: Change current directory.

CHKDSK dr/ms: Report disk size/use, and check/correct disk problems. DR works on SuperStor compressed drives.

CLS dr/ms: Clear screen.

COMMAND dr/ms: Command processor. DR has a switch for loading in base/upper/high memory. MS DOS has a /msg switch for storing all error message text in memory.

COMP dr/ms: Compare files. MS has switches /d decimal, /l line numbers, /c case insensitive.

COPY dr/ms: Copies files. DR has switches /s include system/hidden files, /c ask for confirmation, /z strip 8th bit.

CTTY dr/ms: Redirect standard input/output.

DATE dr/ms: Display and set system date.

DEL (ERASE) dr/ms: Delete files. DR has a switch /s to include system files.

DIR dr/ms: List files. Only the /p and /w switches share the same function. MS DIR is more powerful than DR.

DISKCOMP dr/ms: Compare disks. DR has switches /a beep when done, /m multiple comparisons, /v verifies that the whole diskette can be read. DR will use expanded, extended, or hard disk to hold all the disk image so that you do not have to swap disks as you do with MS when dealing with one drive (above 360K).

DISKCOPY dr/ms: Copies disks. MS has a /v verify switch. DR has switches /a beep when done, /m multiple copies. DR will use expanded, extended, or hard disk to hold all the disk image so that you do not have to swap disks as you do with MS when dealing with one drive (above 360K).

EXE2BIN dr/ms: Convert EXEs to binary

format (COM file). DR has switch /s for segment fixups.

EXIT dr/ms: Exits extra command processors or exits from a program DOS shell call.

FASTOPEN dr/ms: Caches recently accessed file names and their full path location so that DOS can find them faster next time. DR uses this in config.sys (see section on CONFIG.SYS). MS has it as a TSR program.

FC dr/ms: Compares files. Switches /a, /c, /l, /b, /w are the same. All other switches are not.

FDISK dr/ms: Hard disk configuration. DR has a switch /d to allow deletion of existing non-DOS partitions.

FIND dr/ms: Locates text in files. DR /u matches MS /i. DR has switches /b changes display format, /f show only names of files, /s search subdirectories. DR allows wild cards in the filename parameter.

FORMAT dr/ms: Format disks for use. DR has a switch /a for beep when done. MS has switches /b make space for system files, /q quick format. DR does a quick format automatically if DR sees that the disk was previously formatted.

GRAFTABL dr/ms: Load character sets (foreign or extra) for ASCII values 128-255. MS can handle 852 Slavic (Latin II).

GRAPHICS dr/ms: Allow graphic screens to be printed. DR handles IBM-compatible graphic printers. MS supports a variety of printers and has extra switches.

JOIN dr/ms: Joins a drive letter to a subdirectory.

KEYB dr/ms: Load foreign keyboards. The code and codepage are the same but all other switches are not. DR has a switch to load in base/upper/high memory. MS supports more countries.

LABEL dr/ms: Lets you modify a disk volume label.

MEM dr/ms: Show how memory is used. Only the /d switch is shared. MS has switches /p and /c. DR displays line borders around the output which make understanding it easier. DR has switches /b OS BIOS DOS mem MCBs, /s disk buffer chain, /p pause, /m graphic display of RAM ROM EMS, /a show all.

MKDIR (MD) dr/ms: Create subdirectory.

MODE dr/ms: Set device modes. Both DOSs are the same for shorter forms of parameters. MS allows a long form on some parameters.

MORE dr/ms: Show data one screen at a time.

NLSFUNC dr/ms: Set code pages. DR has a switch to load in base/upper/high memory.

PATH dr/ms: Set search path.

PRINT dr/ms: Set up a print queue (print spool) for background printing.

PROMPT dr/ms: Set DOS prompt. DR has a /x switch that runs a command every

time a program returns to the DOS prompt. You specify the command to run by setting the PEXEC environment variable.

RECOVER dr/ms: Recover files from a bad disk.

RENAME (REN) dr/ms: Rename files.

REPLACE dr/ms: Allows you to selectively copy files. DR has switches /m merge, /n preview.

RESTORE dr/ms: Restore files backed up with BACKUP. DR /r matches MS /d.

RMDIR (RD) dr/ms: Remove directory.

SET dr/ms: Set environment variable.

SHARE dr/ms: Support for file sharing. MS has a /f switch to allocate file space. DR has switch /x to disable share, and can be loaded into base/upper/high memory.

SORT dr/ms: Sort text file data.

SUBST dr/ms: Allows you to specify a drive letter to another drive or path.

SYS dr/ms: Copy DOS system files to a disk.

TIME dr/ms: Display and set system time. MS allows 'a' or 'p' format. DR has a /c switch to continuously display the time until a key is pressed.

TREE dr/ms: Display directory paths. Only the /f switch is the same. All other switches perform different functions. MS shows a graphical display by default, DR you must use the /g switch.

TYPE dr/ms: Display file contents. DR has a /p switch for pause and a format for passwords.

UNFORMAT dr/ms: Allows you to recover data that has been formatted over. MS lets you rebuild a corrupted hard disk partition.

DR and MS share the following functions but with different command names.

SID dr, DEBUG ms: Debuggers. The commands are not all the same but many are. Both look and act much the same.

HILOAD dr, LOADHIGH ms: Load program into upper memory.

The following commands are found only in MS.

QBASIC ms: BASIC interpreter with on-line help. This program is also used as the MS editor as EDIT.COM calls QBASIC in the editor only mode. MS comes with some BASIC programs ready to run.

EXPAND ms: Needed to expand the compacted files on the install set.

SETVER ms: Used to change the DOS version number for programs that will not run if they see version 5.0 from DOS.

SWITCHES ms: Makes an enhanced keyboard act like a standard keyboard. For programs that have problems with an enhanced keyboard.

The following commands are found only in DR.

CURSOR dr: TSR that sets the cursor flash rate for better viewing on computers with LCD displays.

DELQ (ERAQ) dr: Deletes files but will

prompt you for delete of each. This functionality is duplicated in the DEL /c command.

MOVE dr: Move a file, group of files, or subdirectories from one location to another. Supports a variety of switches for read-only files, dates, file attributes, verifying, etc.

SCRIPT dr: Provides PostScript support.

TOUCH dr: Change the date and time on existing files. Supports a variety of switches for read-only files, include all subdirectories, etc.

XDEL dr: Expanded DEL command. Supports a variety of switches for deleting and or removing subdirectories, read-only files etc. Also has a switch for physically overwriting erased file data from the disk so that it cannot be found with a sector editor or from being unerased and viewed.

XDIR dr: Expanded DIR command. Supports a variety of switches for file attributes, sorting, computing checksums, subdirectories, compression ratio, etc.

RAM Drives and Disk Caches

Both DOS come with RAM drive and disk cache programs.

VDISK.SYS dr, RAMDRIVE.SYS ms: Create a RAM disk in base/extended/expanded memory of a specified size. Both can be loaded into upper memory.

SMARTDRV.SYS ms: Support for disk caching in extended/expanded memory. Can be loaded into upper memory. You can set a cache size from 128 to 8192 Kb, and minimum cache size.

SUPERPCK.EXE dr: Support for disk caching in base/extended/expanded memory. Can be loaded into upper memory. SUPERPCK is loaded from autoexec.bat or command line. If you run WINDOWS in Standard or Enhanced mode, you must load PCKWIN.SYS also. There are more than 30 options to specify. Some of the options are, do not cache drive specified, memory use, enable/disable specified amount of memory lending, reserve specified amount of memory for other programs, display disk access light, use advanced support for writing, use optimal algorithm for disk transfers, set specific cache size, and suppress writes of identical data. The options that can be called after SUPERPCK is running are, disable/enable cache, flush cache, display measurements, display options in effect, and unload program from memory.

Editors

EDLIN ms: Simple line-oriented editor for ASCII files.

EDIT ms: Simple full-screen editor with help. Supports the mouse, has pull-down menus, and dialog boxes. There is a 255 character per line limit and it can handle files larger than what will fit in ram. Editing

functions can be accessed with hot keys or with menus and dialog boxes. Graphic characters can be entered. You need QBASIC.EXE to run the editor as EDIT.COM calls QBASIC.EXE.

EDITOR dr: Simple full-screen editor with help. The editor functions by using CTRL key sequences as well as the special keys (HOME, UP, DOWN, etc.). Lines can be longer than 255 characters. Files can be larger than what will fit in ram. Graphic characters can be entered as well as CTRL codes.

The DOS editors are good for simple text editing. The MS editor has a nice user interface and looks colorful. The DR editor has no nice user interface (unless you really love the old CTRL key interface) no mouse, and no color.

Data Recovery

Both DOS's have a set of commands to help you recover data.

RECOVER dr/ms: Lets you recover the good sectors, at least, of one or more files from a disk that has bad sectors. DR allows wild cards in the file name.

UNDELETE dr/ms: Will undelete deleted files by using one or more methods which can be specified. DR allows a date or days past specifier and can be menu driven or command line driven. DR's menu mode shows deleted files, allows you to move around directories, and shows by what method deleted files can be undeleted. DR can undelete subdirectories.

UNFORMAT dr/ms: Lets you get back data on a disk that has been reformatted with a safe format (no /u). MS can also rebuild a corrupted hard disk partition and has other switches to show what it can do before you do it.

DELWATCH dr, MIRROR ms: Both are TSRs that help you recover deleted files on the specified drive or drives. This command for MS will also help in unformatting a disk. Both have a variety of switches and will save information on files you delete so that you can recover the data if needed later.

DR and MS use different methods to keep track of deleted files on the specified drive. DR flags the deleted file as a pending delete. This means all the data to the file exists and can not be copied over or used by DOS. DIR will not show the pending deleted files. CHKDSK will show the number of pending deletes and space taken up. You must use DELPURGE to really delete the file and free disk space. MS used a different method which keeps track of the disk areas used by the deleted file. This method takes up a small fixed amount of disk space per disk. However, DOS can copy over and use the deleted file's disk area, thus MS may not be able to recover any of the file or only part of it.

DELPURGE dr: Used to actually delete

(free up disk space of pending deleted) files. There is a variety of switches to use.

DISKMAP dr: Allows you to save, take a snap shot, of the specified disk's file structure. UNDELETE can then use this information to recover deleted files. Unlike DELWATCH, this is not a TSR and it cannot guarantee that a deleted file can be recovered.

Security

DR has two methods of protection for your system. One is PASSWORD and the other locks the hard drive from being used at all.

You can lock your system's hard drive from being used by calling SETUP or during install. You give two passwords, one is the user password that will have to be given at boot up to gain access to your computer's hard drive. The other is the master password that will allow you to disable the lock so that no password is needed. Even if you boot from a floppy with DR or MS, you will not have access to the hard drive.

LOCK dr: Temporarily lock your operating system while you are away but want to leave the computer on. A password is needed to unlock the system.

PASSWORD dr: With this command you can protect a single file, multiple files, or directories. There are three levels of protection you can set for files and one for directories. To gain access to protected files, you can specify a global password for the system to try on all files, or unprotect them with PASSWORD, or use a special syntax with the file name (example: "type safe.txt:june" note the ';' and the password june).

Your files are somewhat safe if the people using your system are not programmer types or know how to run software that might allow them to break PASSWORD protection. Using the SETUP method to lock your hard disk is more secure than PASSWORD.

Command Line Editing

HISTORY dr, DOSKEY ms: DR expanded command line editing is turned on in the config.sys file. MS has a TSR program called DOSKEY. Both allow easier editing of the command line and can save old commands which you can quickly bring back without retyping.

The only major difference is that MS has a macro capability and DR does not.

On the minor differences, DR has extra editing keys like CTRL-K delete to end of line and such. MS can list all the commands but DR only lets you use the UP/DOWN arrows.

Both DR and MS have a feature where you enter one or more characters, then press the search key to match the last command beginning with those characters. DR has an extra search mode that

stays on. Thus, as you type, DR keeps updating the command line with the last command matched.

Another DR feature is when the expanded command line editing is in effect, programs that use the standard input also have all the expanded command line editing features. With MS you only have the normal keys like F3 and so on, but not the expanded features.

Hard Disk Defragment

DISKOPT dr: This is a hard disk optimizer that will de-fragment the files on your disk. It is simple to use and will also sort your files by name, extension, date, or size. There is a cluster sort option that can speed up the process. While the program is running you can graphically see how things are progressing. You can stop the process at any time.

Disk Compression

DR comes with SuperStor which allows for disk compression. With this you can, on the average, double your disk space. You can compress all or part of a hard disk partition. Up to 8 partitions can be compressed. Once the partition is set up it acts like a normal disk. Your programs deal with the disk in the normal way and the SuperStor device driver takes care of compressing the data when disk writes are done and decompressing data when disk reads are performed.

SSTORDRV.SYS dr: This is the disk compression device driver.

DEVSWAP.SYS dr: Used to change drive assignments so that compressed drives retain the normal drive letter.

SSTOR dr: This is a disk compression utility. It lets you set up a partition by compressing it. You only need to do this once. You also can remove the compressed drive and get statistics on compressed drives.

SuperStor is compatible with Microsoft Windows but Windows 3.0 swap files cannot be stored on the compressed volume.

When you create a compressed drive, the existing files will be compressed. However, if you change a compressed drive back to normal, your data will be lost. Save the files first, if needed.

FILELINK

FILELINK dr: This is a program that allows you to transfer files between two computers using the serial ports. You will need an RS-232 Null Modem cable for this. FILELINK runs in one of two modes, master or slave. In the slave mode the computer acts only on requests from the computer running in master mode. In the master mode you request transfers to and from the computer running in slave mode. FILELINK is command line driven. Transfer baud rates from 110 to 115,200 are supported.

The following commands can be given to FILELINK.

SETUP: Set com port and baud rate to use.

DUPLICATE: Install FILELINK on a slave computer. If the slave computer has no floppy drives, you can use this command to copy FILELINK over to it.

SLAVE: Set the computer in slave mode. A switch tells FILELINK to not overwrite any files on the slave computer.

DIRECTORY: Displays a list of files on the slave computer. Switches can be used to show archive/hidden/system files, subdirectories, modified since date, and pause for long displays.

TRANSMIT: Allows you to send files to the slave computer. A variety of switches can be used which allow you to copy files only with certain file attributes, copy only if file does not exist, prompt, subdirectories, or update.

RECEIVE: Allows you to copy files from the slave computer. The switches that can be used are the same as for TRANSMIT.

QUIT: Tells the slave computer to stop running FILELINK.

Task Switching

Both DOSs have a task switching feature. This is not multitasking, it just lets you save the state of a currently running program to disk/extended/expanded, and allows you to switch between two or more of these saved states and continue running the selected program. DR lets you save up to 20 programs, MS up to 14 programs. Both DOSs can switch tasks in a reasonable amount of time (few seconds), and are easy to operate. Hot keys are used to bring up the task switcher menu. The list of tasks saved is shown in menu form. From the task menu you can jump to any task, delete a task, or start a new task. Both DOSs let you use the mouse while in the task menu and can handle TSR programs.

DOSSWAP ms: MS task switching is accessed only through its DOSHELL program. The task program, called DOSSWAP, cannot be ran from the DOS command line. You start new programs by pressing the task hot key, then with a dialog box you tell MS what program to run.

TASKMAX dr: DR task switching is started by using the TASKMAX command. TASKMAX loads as a TSR and can be put into upper ram. You must have SHARE loaded before TASKMAX is ran. Once running, you can access the task switcher through a hot key or through VIEWMAX (DR's dos shell). There is a variety of switches that let you control where TASKMAX stores programs (disk/extended/expanded) and change the name of a task as shown on the task

menu. You can also copy a rectangular block of screen data from one task to another task. You start new programs by pressing the task hot key, then select new on the menu. After this you are brought back to the DOS command line prompt where you either call a new program or do normal DOS commands.

DOS Shells

DOS shells allow you to run programs, copy/ren/delete files, view directories and other various DOS functions using menus, dialog boxes, selection lists, keyboard, and mouse. The DOS shells are meant to make using DOS easy (compared to the DOS command line) and to look visually appealing. Both DOS shells have the following functions in common:

- View drives, directory trees, and file lists.
- View two file lists at the same time.
- Modify how file information is shown.
- Modify DOS shell's colors as well as various operating characteristics.
- Run the DOS command processor.
- Switch, add, delete tasks.
- Select one or more files in the same directory or across directories.
- Run programs or batch files.
- Find files.
- Search, view, copy, delete, print, change attributes of files.
- Create directories.
- Associate file types to a program.
- Access on-line help about DOS shell functions.

DOSSHELL ms: DOSHELL can be run in a variety of text and graphics modes. The number of lines displayed can be changed. In graphics mode, files have an associated ICON based on file type. You can move files. Programs/files can be set up as a group and the group can be assigned an access password. Help messages can be assigned to a group. You can specify other properties to programs such as memory requirements, video mode use, reserve hot keys, and prevent task switching.

VIEWMAX dr: VIEWMAX is a graphics mode program. Formatting, undelete, and setting a new search path can be done. An ICON mode or text mode can be set. Directories are called folders. You can copy programs by dragging them with the

mouse to another folder. You can set programs and data files to any of 20 pairs of icons. DR type passwords can be set. A simple calculator and clock with date and alarm can be accessed.

The MS shell (even in graphics mode) acts faster than the DR shell. The DR shell appears a bit more graphically-oriented than MS.

Using DR With Novell Network

In the DR manual is an appendix on using DR with the Novell network. In the DR booklet "Optimization and Configuration Tips" there is a section on Novell and other networks that DR will run with. On the last install disk is a subdirectory called NOVELL. In this directory is a set of files for networking use. Basically, you replace your old IPX.COM and NET5.COM files with a new SHGENed version of IPX.COM and NETX.COM from the install set. Also, you may use a version of NETX.COM that runs in EMS or XMS memory.

Other Device Drivers

Below are other device drivers that come with both DOSs.

ANSI.SYS dr/ms: Standard interface for video and keyboard functions. MS supports 20 video modes. DR supports 7 video modes.

DISPLAY.SYS dr/ms: Support code page switching for EGA/VGA. MS has an lcd type. DR can have up to 12 additional code pages.

DRIVER.SYS dr/ms: Create logical drive. MS supports 2.99MB type, 1-99 heads, 1-999 tracks. DR has /N for permanent media.

PRINTER.SYS dr/ms: Supports code page switching for parallel ports. DR supports more printer types.

The following is found only in MS.

EGA.SYS ms: Save/restore EGA display for MS task swapper.

The following is found only in DR.

EMMXA.SYS dr: Supports LIM 4.0 for IBM PS/2 computers.

Memory, Speed Tests, and Compatibility

I noted no speed differences between the two DOS's when running various speed testing programs on them and running a

		DR	MS	
DOS loaded in BASE ram.	System free:	587,040	591,264	bytes
	CHKDSK *.* /v:	13.8	10.8	seconds
	find all *.COMs:	3.3	3.6	seconds
	COPY C:.* A::	44.2	52.2	seconds
	TFIND xyzzy -s. *.c	25.0	22.0	seconds
DOS & DOSKEY/HISTORY.	System free:	585,632	587,104	bytes
DOS loaded in HIGH ram.	System free:	627,280	636,464	bytes
DOS loaded with EMM386.	System free:	641,680	628,048	bytes
DOS using VIDEO ram.	System free:	739,984	N/A	bytes

Figure 1

stripped down configuration. However, these speed tests probably do low level stuff and are not specific to DOS. So, I did some other simple tests. Figure 1 shows some simple DOS type speed tests and memory specs I did. These were done on a 25/386. TFIND is a program that finds text across files in subdirectories, and I included it as it is not a DOS command but works the DOS directory-IO code.

If you play around with advanced memory and system configuration (more than I have shown above) you will find that sometimes DR will have more BASE ram and sometimes MS will have more. If you can run DR's feature to add some of the unused video ram to BASE ram, DR definitely comes out better over MS.

I really have not done much testing of speed or memory of the two DOSs. So I cannot say where DR and MS differ in all cases and set ups. This would be an article in itself.

I have been running Windows 3.0 with DR and on a SuperStor compressed drive. I have not noticed any major speed degradation due to running on the compressed drive, although there is the overhead of compressing and uncompressing the data.

DR is internally compatible with DOS version 3.3. If a program requests the version from DR, it will get 3.3 back.

My older Zenith Data Systems 286 had a problem with HIDOS.SYS version 2.00 where the computer would lock up at times. DR sent me a new HIDOS version 2.02, and that fixed the problem.

Another problem was with a 386 running EMM386.SYS, SUPERPCK.EXE and Codeview. Codeview is a symbolic debugger from Microsoft. Codeview does not appear to see the extended ram when running EMM386.SYS. Yet, Codeview has no problems if I run HIDOS.SYS. Also, Codeview and SUPERPCK.EXE seem to have a problem. I have a work-around for now, but the problem has not been solved as of this writing and is being looked at by DR. I do not know if it is a problem with DR's EMM386.SYS or with MS's Codeview and neither DR nor MS could tell me.

If you have been up on the news, you know that DR will not run the beta 3.1 WINDOWS and MS is refusing to help or give DR a beta copy. NOVELL said that they will give DR all the help it can as MS is helping them and they have a beta 3.1 copy. I assume, since DR runs WINDOWS 3.0 now, that they will in time have it running with 3.1. You will have to keep up on the latest news in this area if you will be moving up to WINDOWS 3.1, when it gets out, and using DR.

I run a variety of software at home and at work with a NOVELL network. So far I have had no other problems with using DR.

I had no problems running some MS drivers with DR. DR can run MS

RAMDRIVE.SYS and SMARTDRV.SYS. DR would not run MS EMM386.EXE though.

Last Words

Even with the problem with Codeview, which I hope will be fixed, I still favor DR. DR's extra features and loads of switches are very useful. Some of the add-on software I was using with MS DOS 5.0, I took off my computer as it was no longer needed with DR DOS 6.0. The fact that every command has a /h help switch is nice as well as all the information in DOSBOOK. I like having over 640K of free system ram (See ADVANCED MEMORY MANAGEMENT) and that is with a ram disk, disk cache, mouse, EMS, and the network shells loaded. The SuperStor (disk compression driver) is great and gives me more disk space without buying a larger disk drive. If you are on the spending conscious side, DR gives you more features for the same price as MS. I am looking forward to what the new versions of DR will bring.

Microsoft has had little competition with their DOS. With DR DOS' 6.0 features and the backing of the large NOVELL company, should promote more competition to make DOS' better and more affordable for people. Even if you never buy DR DOS, MS DOS may be made better just to keep up with some of the features in DR DOS.

Whether DRDOS 6.0 is the best choice for you or not, I cannot say. The software you run, your particular needs and preferences, and your past experiences should be weighed by yourself. I hope only that this article has been of some assistance in

the process of looking at more than MS-DOS or at least for the knowledge of what DR DOS 6.0 is.

Other Sources of Information

On COMUSERVE you can go to the Digital Research Forum (GO DRFORUM). There are many files related to DR 6.0.

DR DOS 6.0
Digital Research Inc.
70 Garden Court
P.O. Box DRI
Monterey, CA 93942

Retail Price:	\$100
Discount Price:	\$65
Lowest I've Seen:	\$65

MS-DOS 5.0
Microsoft
16011 NE 36th Way
Redmond, WA 98073-9717
(206) 882-8080

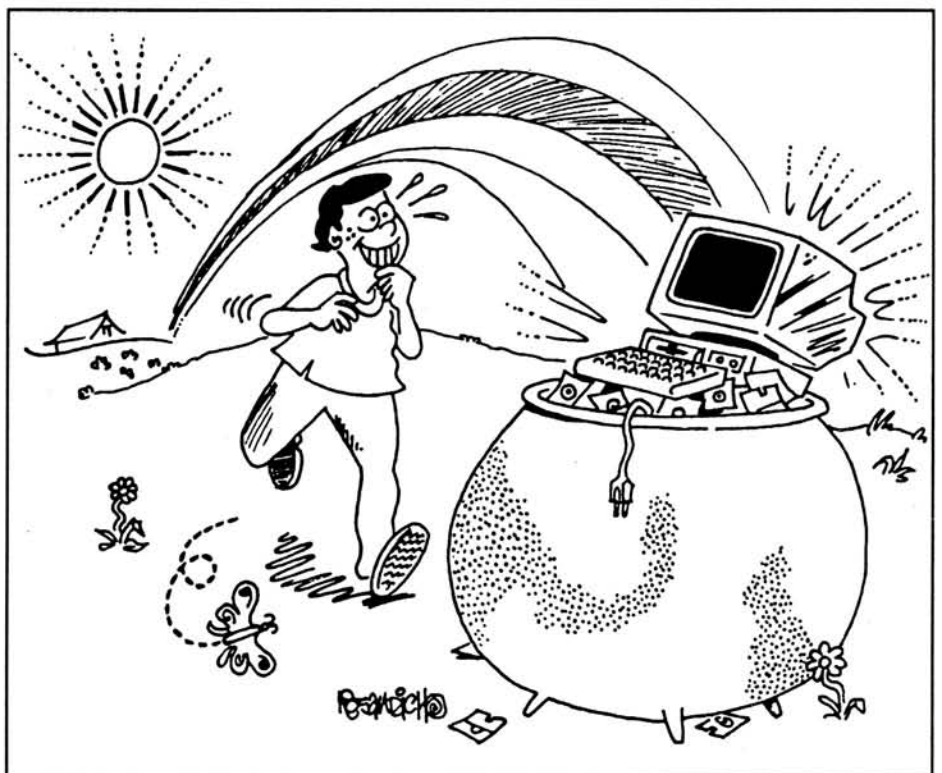
Retail Price	\$100
Discount Price	\$59
Lowest I've Seen	\$39

Books on DR DOS 6.0 are:
Stepping Up to DR DOS 6.0
ISBN 1-55755-129-4

Abacus	\$14.95
1-800-451-4319 for orders	

Power of ... DR DOS 6.0
Tom Godell
ISBN 1-55828-128-2

MIS: PRESS	\$24.95
(800) 488-5233 for orders *	



Sorting, Searching and KWIC Indexes

Part 2

John Day
59, rue Nationale – Montreal
75013 Paris, France

This is the second article on efficient sorting of large arrays in core. The first article explained the different ways of searching through a table, and the basic sort algorithms. This time, I shall cover the high-speed algorithms that are efficient once the array size goes over fifty entries or so, and wind up with a full example of a program that can sort 15,000 lines to a print file in less than ten minutes. As in the previous article, all times were measured using a Zenith Data Systems 386-SX running at 20 MHz, using the Basic maths emulation package whenever I had to include decimal values in the code.

Speeding Up Sorts

Simple sorting gets very slow as arrays get larger, because sort times increase with the square of the array size. Many methods exist for splitting the array into smaller sub-arrays which can be sorted quickly, then merging them. If you have 30 000 entries to sort (estimated running time about 15 minutes), you could split this into eight sub-arrays of about 4 000 entries, sort them rapidly (15 seconds per array gives two minutes estimated running time), then do a fast merge to build one sorted array. This works extremely well, but... you're going to double your memory requirements, because you will have to merge into a separate array.

A partly-ordered array can be sorted much faster than a random array, because there is much less moving around to be done. If an array is nearly in the correct order, then the standard sort algorithms will zip down it in next to no time. I had to be careful with my timing runs to be quite

sure that my arrays were really random, to get meaningful times. An efficient solution based on this fact was conceived in 1959 by Donald Shell, in the sort algorithm that bears his name. He interleaved the sub-arrays; for a split into eight, the first sub-array was the first main array entry and every eighth entry following, the second was the second array entry and every eighth entry following, and so on. At the end of the sub-sorts, the main array wasn't sorted, but it had been roughed into shape. Now do the same thing twice more, but splitting the array into four, then two sub-arrays. The array is now nearly sorted, and a final, full insertion sort will rip down the array in record time. The method was subsequently improved, between 1965 and 1969, with some insight and a lot of benchmarking; it goes even faster using either prime numbers of sub-arrays, or numbers from the sequence 29,524, 9841, 3280, 1093, 364, 121, 40, 13, 4 and 1: each of these numbers is 3 times the following plus 1. Best results were obtained starting with sub-arrays containing 3 to 10 values each; for this, start with the second value in the series which is smaller than the array size

— example, for 30 000 entries, start with 9,841 sub-arrays of three or four entries each. Since all sorting is of arrays which are either small or nearly sorted, straight insertion is as good as binary insertion, and much easier to code. (Figure 1)

As you can see, a Shell sort is not much more complicated than a straight insertion sort, but it does require four nested loops. Array *ST* contains the sub-array sizes; the first value, just smaller than the largest array you intend sorting, will never be used, but must be there since the first usable value is always the *second* in the array. The loop "WHILE TMax < ST(F)" finds the first sub-array size smaller than the array to search (and won't work if TMax is smaller than 5!). Then the four loops run:

```
DATA 29524, 9841, 3280, 1093, 364, 121, 40, 13, 4, 1
DIM ST(9) ' Shell sub-arrays
FOR I = 0 TO 9
  READ ST(I)
NEXT I
F = 0
WHILE TMax < ST(F)
  F = F + 1
WEND
FOR G = F+1 TO 9
  Increment = ST(G)
  FOR H = 0 TO Increment - 1
    FOR I = Increment+H TO TMax STEP Increment
      Temp = T(I)
      J = I - Increment
      WHILE J > 0 AND T(J) > Temp
        T(J+Increment) = T(J)
        J = J - Increment
      WEND
      T(J+Increment) = Temp
    NEXT I
  NEXT H
NEXT G
```

Figure 1

"C" controls successive sub-array sizes, from the second smaller than the array size down to 1. Example, for a 500-entry array to sort, F will be set to 4, and G will loop from 5 through 9 to give 121, 40, 13, 4 and 1 as successive values for "Increment".

"H" loops through all the sub-arrays of a given size. For Increment = 4, for example, H will loop through the four sub-arrays starting at T(0), T(1), T(2) and T(3).

"I" and "J" are the same as we saw in the short insertion sort. Wherever there was a step of "1" in the short code, we now put a step of "Increment". Since Increment = 1 for the last pass, this means that the final run will be the full short sort itself.

Figure 2 shows the measured running times in seconds (250 integer array entries are sorted in less than a timer tick!):

	250	500	1 000	2 000	4 000	8 000	16,000	32,000
Direct array sorting								
Integer keys	-	,1	,1	,3	,6	1,4	3,3	13,6*
Decimal keys	,3	,7	1,7	4	9,3	22	50	
String keys	,1	,3	,7	1,6	3,9			
Address table sorting								
Decimal keys	,3	,7	1,7	4,1	9,6	22,5	52	
String keys	,1	,2	,4	1	2,5			

Figure 2

* The 32,000 entry integer sort was run with double-precision address calculations. This was necessary as for the first pass the loop on *l* stops when *l* is much greater than TMax, the overshoot being up to 9,841. This means that loop calculations using two-byte integers will overflow when the array size exceeds 22,926 entries. Double-length integers nearly double the running time, but the sort still clocks in at a quarter of a minute, against a quarter of an hour for the corresponding insertion sort.

With Shell sorting, nothing I could squeeze into RAM took more than a minute to sort. The only drawback is having to have the entire array in memory before beginning the sort. However, if you want to read values and sort them at the same time, you can *batch* several insertions together. Read forty or fifty values for insertion into a small, separate array and sort them. I don't know how long it takes to sort forty entries, because it's less than a timer tick. Now insert all forty values into the main array, starting with the last and moving array entries down 40 positions; once it's in place, switch to the 39th and move entries down 39, and so on. This batched approach will take less than twice the time to insert one unbatched value, because working unbatched, you will move half the array down on average, and the worst case in batching will be when you have to shift the whole array. In the sample KWIC program, I shall show you how to handle this by batching all the keys generated for several magazine titles into one insertion pass. My

sample uses an address table, and you will see how all key and satellite data can go straight into the main table, without requiring extra storage. The only extra is an auxiliary table to sort the address pointers before batching them into the main table. If you don't use an address table, you will have to have auxiliary arrays for all your keys and associated data to sort them before inserting them into your main arrays, which is more complicated.

If you want to go faster still, **CombSort** was published in the April 1991 issue

	Desktop Publishing – Improving your Im	Jan 91
Improving your	Image	Jan 91
p Publishing –	Improving your Image	Jan 91
Desktop	Publishing – Improving your Image	Jan 91

Figure 3

and print the result – a nice demonstration of how to handle a large-scale sort. There will be several compromises, because we also want the solution to be simple.

With any such program, the first thing to do is design the output layout; this way, you know right from the start what data the program must handle. I decided straight away to line up the keywords down the middle of the page – this makes browsing easier if you're not sure which keyword

you're looking for – and right-justify the issue date. Look what this gives with the first article from 1991, "Desktop Publishing – Improving your Image". (Figure 3)

Obviously, for keywords near the beginning or the end of the title, you may lose a few words off the end. This doesn't matter very much if it's the tail end of the title you lose, but it is

a nuisance for the start of the text, where the punch words are usually found. So, I decided to wraparound; anything shifted off to the left is printed to the right, after a wraparound marker " + ". Visually lining up "Image" on the second line with the

of "Byte", and subsequent correspondence dealt with optimizing some of the internal calculations. It's a scorcher; it cranks up bubble sorting to run 15% to 20% faster than Shell sorting, for about as much code. As with the Shell method, you have to have the whole array in memory before you start. It's the only algorithm mentioned here which isn't in Donald Knuth's classic book "Sorting and Searching", published by Addison-Wesley in 1973. I found this book most helpful, in particular for the basic principles of the Shell sort. Knuth goes on to explain other algorithms, including **QuickSort** – which still beats CombSort. Read the book if you want to use the QuickSort algorithm; it is a bit more complicated than the Shell sort because it uses the result of each comparison to decide which keys to compare next – and it lives up to its name!

The KWIC Index

Now to wrap everything up into an applications program. Our aim is to maintain a disk file with titles and issue dates for magazine articles; maintain a second disk file with a list of not key words; extract all words from the first file, check them against the second, sort those which are keywords

KWIC Index to the first article in 1991

	Desktop Publishing – Improving your Im	Jan 91
Improving your	Image + Desktop Publishing –	Jan 91
p Publishing –	Improving your Image + Deskto	Jan 91
Desktop	Publishing – Improving your Image ...	Jan 91

Figure 4

issue date isn't easy, so we'll add some dots to guide the eye across the page. It will also look nicer with a page heading and a page footing; the heading can be supplied by the user, and for a footing I like the two extreme keywords, dictionary-style. (Figure 4)

Desktop – Publishing

Trial runs using a first version of the program showed that this layout is quite usable. Sorting on twenty characters is adequate; there are very few lines which are not in the correct order after such a sort, and when the inversion is after the twentieth character you hardly notice. One thing did show up, though: the necessity for cut-out / cut-in codes. With regular series like "On the Leading Edge", it is better to use the sub-title, like "Computer Model Numbers", and not generate entries based on

the series name. I chose square brackets to handle this, and modified the program so that everything between square brackets is printed, but not analyzed:

[On the Leading Edge]

Computer Model Numbers
only generates three keywords, Computer, Model and Numbers.

You have to decide what to put in your list of not keywords. Start with:

A, An, And, Are, By, Computer, Do, For, From, Getting, How, I, In, Is, It, Its, New, No, Now, Of, On, S, The, To, Up, With, You, Your

These are the most usual meaningless words in REMark titles (everything's about computers, anyway. If you're indexing a general-interest magazine, drop "Computer" from the list). To check your list, do a dry run on your title file — outputting to disk if possible — look down the print file to see which keywords serve no useful purpose, and add them to your not-key file.

Now to the file layouts to handle all this. The easiest title file to maintain is standard ASCII, which can be handled well by any word processor; the same applies to the list of not key words. Fifty not key words were adequate for test runs on live data, which is well within the capacities of a core sort; but the titles are going to run to a thousand or more values of up to a hundred characters each, which is more than we can handle in memory. There's always tag sorting, but that only works with random files — or does it? Any file can be opened as binary, either with "OPEN filename FOR BINARY AS #n", from QuickBasic version 4 on, or "OPEN filename FOR RANDOM AS #n LEN = 1" for older compilers. This wouldn't be much help for a file arranged in lines... except that we have to scan the whole file, character by character, to pick up the keywords anyway. Whilst we are doing this, we can quite well pick up the CR/LF pairs and build a table of record addresses, giving us what is effectively a random-organized file. The table will give the byte number for the start of each record and not the record number, which is fine because a random file with variable-length records suits us even better than a fixed-length structure.

This lets us fix a format for the main file. The easiest to handle is:

issue date,title....

giving

Jan 91,Desktop Publishing — Improving your Image

This is fast to key in, because you can repeat the issue date down the left of the page and fill in the spaces. It is flexible, because the first comma on the line marks the end of the issue date, and then everything that follows (including other commas) is part of the title down to the CR/LF line terminator — all characters accepted. It's also easy to code.

Since we are generating lines as we read the title file, I opted for a sort working on-the-fly. Large insertion sorts are not efficient, so I went for batch insertions, where a certain number of output lines are sorted together, then inserted into the main table.

Arrays in memory for the main sort are designed to contain minimal information — there isn't room in 640 Kb for more. The sort key is spread over four long-integer arrays to give 20 effective bytes. The disk byte address table could be either decimal or long integer; short integers are too short, as they would limit the disk title file to 32 Kb; the live test took 40 Kb for ten years of back-numbers, and the "maximum capacity" test required over 200 Kb of input. One more array gives the relative start byte and length for the key within the disk record; we need this to line up the keys, and put bold-on/bold-off sequences either side. Both values were squashed into one integer, on the principle of not wasting space. "C = 256 * CL + CP" shifts CL left 8 bits into C, and puts CP into the low-order part. CP is limited to 8 bits or 255, and CL to 7 bits or 127 - it mustn't get mixed up with the sign bit! "C AND 255" masks out CL to leave CP, and "C \ 256" shifts CL to the low-order position whilst shifting CP clean out.

This fixes the maximum capacity of the program. Decimal and long integer arrays are both limited to 16K entries, so that will be our maximum number of keywords and lines on output. We've already seen that standard integer arithmetic is OK for binary searching up to 16K, so this limit suits us. There is no reason to limit the size of the title file or the records in it, although a decimal address pointer can only point up to a megabyte or so (6 digit accuracy). Since we are batching all sort insertions for each title, we must set arbitrary limits for the number of keywords per title (I've chosen 30). Not key words are limited too; I chose an array limit of 200, which is four times what I actually need for typical titles. Finally, because of the structure of the start-byte/length array, no keyword can start after byte 255 in the title; it doesn't matter if a keyword is more than 128 characters long, because the program only bothers with what can fit on the printed line, less than forty characters. I don't think this limit will ever be reached in practice. Four long-integer arrays for the keys, one for the pointers, and the address table and offset/length table add up to 384 Kb for the big arrays plus the program and its own data segment. This will fit into available memory as long as you aren't using too many TSR

programs.

The page header has to be put

somewhere. The best place I found was as the first line of the titles file, where it can be keyed in with the corresponding data. If you are indexing several magazines, each disk file will have its own title attached. The disk format for our example is now:

KWIC Index to the first article in 1991
Jan 91,Desktop Publishing — Improving your Image

This program illustrates the necessity of matching data and sort techniques. My first idea was to batch all the print lines from one title. This is very easy to code: read a title, generate the print lines, sort them, and insert them into the main file, then loop back to the next title. The first runs with live data showed that this was an inefficient solution. On average, one title only generates three or four keywords, so batching will do little better than double the sort speed; there aren't enough values to make it worth while. The second solution I tried was adapted to the actual input data used: read titles until forty or more keywords have been generated, then sort them all and insert them in the main array. If you make assumptions about data structures in your programs when optimising the code, do check when you've finished whether your assumptions were warranted or not.

Before going any further, you will want to know what volumes and run times you are letting yourself in for. As soon as a file was available, I ran timing tests on live data to see if sorting was reasonably fast. Since I have back files for the "Scientific American" covering 25 years, this was the obvious choice. I ran the program using a file containing all the main titles from the decade 1980 - 1989. There were 979 titles, a little less than 40 Kb. The 2,817 resulting print lines were generated and sorted in 25 sec., still using a Z-386SX running at 20 MHz. I continued by generating test files to output up to 15 000 lines, which would correspond to 50 years of back-numbers for a monthly magazine. For obvious reasons, I diverted print output to a disk file; I speeded up output by putting the input file on a RAM disk, thereby avoiding read-write conflicts on the hard disk and also avoiding a lot of head movements seeking the file lines in sorted order. This has no effect on the time to sort, but speeds up recovering the file in sorted order afterwards. I tested with the first, inefficient version (at less than three lines per batch insertion run with the data I was using), and then with the second version, batching at over forty lines per insertion run; for the second version, I also took the start-to-finish time to sort and

Lines output:	3 000	6 000	9 000	12 000	15 000
V 1 sort:	30 s	1m 10	2 m	3 m	4m 10
V 2 sort:	25 s	52 s	1m 20	1m 50	2m 20
V 2 overall:	1m 30	3m 10	5 m	7m 5	9m 15

Figure 5

generate a print file on disk. (Figure 5)

These figures tell us a lot about the program. I would expect batches of 40 insertions to run about ten times faster than batches of 3. Since the actual gain is less than twice as fast on long files, and practically nothing on short files, we can deduce that the time taken to sort is now much less than the time taken to analyze the title and generate the sort data. This means that we've reached the point of diminishing returns; further improvements to the sort algorithm aren't worth it, as they can only have a marginal effect on the total time. Most of the compute-and-disk time — 75% of it — is now taken up by building the output lines, so any improvements should be made to the print module and not to the sort techniques. All these values will be insignificant anyway, once you start printing the file. This depends on your material, but with an impact printer you'll be running well under 3 lines per second. 3,000 lines will take a quarter of an hour or more to print, so whether the sort takes one minute, 30 seconds or 10 seconds doesn't really matter. The only advantage to cranking the sort up to top speed is if you want to preview your print file on disk before sending it to your printer.

The program is given at the end of this article. It has been reduced a bit in size, by cutting out a few options which you can put back in whichever format you best like. If you want to use the program often,

- use the command line `COMMAND$` to give the input, nulls and print file names; use switches to change margins, printer control sequences and special characters at run time;
- check files for existence as you open them, and query before overwriting a disk output file;
- if you're planning to index big files, add occasional screen output to tell you that the program is actually doing something during your two-minute wait.

Comments on the code: All the tables are shared across all subroutines. The big tables are in separate 64 Kb or 32 Kb segments: AT for the address table pointers, F for the disk byte pointers, C for the key pointer and length, and K1...K4 for the sort keys. The other tables are in the standard data segment: Nuls\$ for the not-keys, CSeq for the collating sequence, CAT for the addresses of lines being batched to the sort, CT for the title converted to the collating sequence, and CK to build the sort key over 20 integers before compressing into K1...K4. There are also several global variables. LFile is the length of the title file, and is used to stop generating at end-of-file — EOF(*filename*) is tricky to use for binary and random files. LNuls is the number of not-keys actually read. LCTab is the number of entries used in the main tables. Not all these are sorted, however; entries are

placed in the main table as soon as they are generated. TCTab marks the number of entries, in the AT array, that have been sorted. Further entries will be addressed through the CAT table, LCat being the number of entries; TCTab plus LCat gives LCTab. As an example, if 90 lines have been sorted and a further 20 are being batched together, LCTab will indicate 110, the next slot to use in C/F/K1...K4. TCTab will show 90, the number of entries in AT; from AT(90) onwards will still contain zeroes. CAT(0) to CAT(19) will contain the values 90 through 109, the unsorted pointers to the part of C/F/K1...K4 that is being generated. When the latest block of entries is sorted into AT, TCTab will be updated to the new sorted length.

The collating sequence table "CSeq" contains the 256 entries for the full character set, loaded from DATA. Characters coded "0", like quotes and periods, are skipped whilst parsing the title line, so "U.S.A." within quotes will have exactly the same sort key as USA without quotes or periods. Characters coded "-1" delimit keys, so commas, brackets and so on will close the keyword being constructed. "-2" marks CR and LF, but this version of the program picks them up directly using their ASCII codes. "[" codes "-4" and switches off parsing; "]" codes "-3" and switches it on again. All positive values are collating sequence equivalents, 1 to 10 for the digits and 11 to 36 for the alphabet. Note the accented characters in the second half of the table; you will find a total of 6 entries coding to "19", "l", "i" and the four "i"s with accents. If you have special requirements, like the Greek characters beyond 224 ASCII, you can add up to 27 more codes.

The two functions compare two keys in the main tables (FNKeyComp), and return the smaller of two values à la Fortran (FNMin).

Subroutine "LoadNuls" builds a sorted table Nuls\$, length LNuls, containing the collating equivalent of the nulls file "NUL-KEYS"—you must use the collating sequence so as to make the nulls table case-insensitive. The temporary string T\$ is used to build the collating equivalent of the word; letters "A" or "a" would both be recoded 11 in the string, the VT character. Once the string is built, a binary search and insertion sort place it in the array Nuls\$. Note that duplicates are discarded; the not-key is only inserted in the table if the binary search returns "not found".

After loading the nulls table, the program opens the title file "TITLES" and reads the page header "PageTitle". It then reopens the file in binary mode, and seeks

forward to the first character after the page header and the CR/LF that follows it.

"LoadTitles" is the first workhorse. It starts by initializing a few variables, then loops through the file. The loop logs "KeyF&" as the disk pointer to the start of the line; this will be the entry in the array "F" for all keywords in the current title. The routine usually handles up to 30 keywords, but will stop at less if the core table is nearly full. It first skips forward to the comma that separates the issue date from the text, logging the length of the issue date as "Offset". I included an end-of-file test, in case there are some lines without proper dates; if you omit the required comma between the issue date and the title text, the program could loop indefinitely looking for it. It then reads up to 300 bytes, converting them to the collating equivalent and loading them into array CT, before positioning the file just after the next CR/LF pair. The rest of parsing is carried out in core using CT.

The processing loop runs forward through CT until it gets a positive or zero entry, which is the start of a word — punctuation signs to ignore are all negative. It notes the position of the starting byte for the word, then builds the collating equivalent of the keyword. "CheckNul" is a simple binary search of the nulls array, which returns "NoGo" true if the word is found. "Scan" is false between a left square bracket "[" and a right square bracket "]", and turns parsing off. If parsing is active and the word is not null, then the routine opens a new slot in the main arrays at LCTab, and logs the markers. It builds a twenty-byte key, consisting of the keyword, a zero byte, and the remaining non-zero collating characters. The zero byte gives better handling of keywords. Without it, you would get entries like those shown in Figure 6, whereas the zero byte moves "Air" in front of "Airplanes" where it belongs. Once the key is finished — including a zero filler if it goes beyond the end of the text line — it is loaded

Ultralight	Airplanes	Jul 82
rolling Indoor	Air Pollution + Cont	May 88
	Air Pollution by Particles	Aug 87

Figure 6

into K1, K2, K3 and K4, and L is updated to the number of keywords generated. There are two emergency exits, included to avoid dud values in "C". If the keyword starts beyond byte 255, it is dropped because it can't be handled. If it is more than 127 bytes long, its length is reduced — only a very wide page would let you see this in the printout.

Each keyword is loaded directly into the main arrays at the next available slot. Its array address is placed in CAT (maximum size 70), and LCat is updated to the number of slots used. At the end of the title, the

program checks that LCat is less than 40 — i.e., that there are at least 30 slots free out of the 70 ready for the next title. If not, it calls "SortTable" to sort CAT and insert it into AT. At end-of-file, it calls "SortTable" once more to add the final few lines.

"SortTable" does all sorting for the main file. Like all routines for batching insertions, it is in two halves. The first part does a straight sort of CAT. The second part does "LCat" insertions into the AT table. For the first insertion, each tail-end entry in AT is moved down LCat slots, and the last CAT entry is fitted into the last slot freed up. "TCTab" is now reset to the end of the unshifted part of the table, and searching and shifting starts again — this time, for "LCat-1" slots. This goes on until all of CAT has been correctly inserted. Finally, TCTab is set to the new length of the sorted table. Subroutine "KeyComp" can be used to sort either "AT" or "CAT", because the key values are always in the arrays K1...K4; AT and CAT both point to this array, AT to the sorted beginning and CAT to the unsorted end. Note also that although insertion sorting requires moving an entry out of the table to temporary storage, to free up a slot, in this case only the pointer (AT or CAT) is moved to the temporary variable "Temp", whilst the actual key entries stay in place in K1...K4.

The key arrays K1...K4 are only used during generation and sorting. All printing is done using arrays C and F only, together with AT to give the correct order in which to read these two arrays. Subroutine "PrintIndex" sweeps through AT, printing one line per entry. Initial variables can be set to whatever margins you want; my full version of the program allows for resetting them from the command line with switches. The first LPRINT statement sets the printer to elite mode, 12 cpi. "PrintMargin" is the number of spaces to the left of the line, 5/6". "PrintKey" is the column to receive the start of the keyword, 2-1/2" from the margin (col. 40, less a 10 col. margin gives 30 print characters). "PrintEnd" is the last column on the page, and will receive the last character of the issue date for a page width of 7-1/2" — a bit over 6-1/2" for the print width. "PageLength" is the number of lines per page, including headings and footings. "BoldOn" and "BoldOff" are the printer controls for setting and clearing double-strike emphasized text. "Mark\$" is the wrap-around separator.

Building the print line looks as though it ought to be very complicated. It isn't, as long as you break the line into its five component parts: a left margin, text to the left of the key, the key itself, text to the right, and the issue date. "PrintIndex" saves the last key in case it is needed for a page footing, then moves to the start of the title in the disk file using the "F" pointer and reads in the issue date. The next test looks

at the character position of the keyword, and decides whether the text preceding it will fit entirely on the line or not. If it will, then "PrLsp\$" is set to the number of spaces necessary to center the keyword on the page; otherwise, "PrLsp\$" is set to the left margin, and "Wrap" is set to the number of characters to drop on the left. "PrLTx\$" is then read in; it is that part of the title to the left of the key which can be printed. "PrKey\$" is the keyword — unless it is too long to fit on the page, in which case it is truncated on the right. "LPrRTx" is the number of characters remaining between the keyword and the issue date, with a two-column margin. "PrRTx\$" is built up progressively to this length. It first receives the title to the right of the keyword. If there are more than three characters free and the title should be wrapped, it then receives the "Mark\$" separator and the beginning of the title. If there is still some space, the program adds up to six spaces, then the hanging periods " ... ". At this point, we have the whole line: "PrLsp\$" on the left, then "PrLTx\$", "PrKey\$" and "PrRTx\$", followed by the issue date "Issue\$" on the right. The whole line is printed, with double-strike/emphasized set before the keyword and cleared after. The checks for a page change are made before printing; if the page is changed, then the keyword is held for the footing line on the new page. For the last page, the program spaces down to the footing line and prints it, followed by a page eject.

The sample code was checked out using the Microsoft QuickBasic 4.5 compiler; you can't run it in the Quickbasic environment without reducing all the table sizes. It was compiled as a stand-alone program, occupying about 47 Kb after linking — I didn't want the 80 Kb run-time library in core. It should run under version 4.0 as it stands. I avoided most of the language features introduced with V 4.0, but there are a few that I did use, because they make the program much more efficient. To run it under version 3, you should make certain changes:

- Replace long integers, xxxx&, with decimal values xxxx#. Unfortunately, this will pull in the floating math package when you link the program, and the size will bounce up.
- Open the TITLES file for RANDOM access with a record length of one byte, and add a FIELD statement. You'll either have to fix it so that the string name in the FIELD statement is shared across modules, or put one FIELD statement at the beginning of each module that references the file.
- Remove the string name in each GET# statement.
- Probably — replace the INPUT\$(n) function, which returns n bytes from the binary file, with a loop to GET# n charac-

ters and add them to the string one by one.

- rethink the structure of the key tables K1...K4. You will have to use eight tables of short (2-byte) integers, and modify the FNKeyComp function. If you put two characters in each key integer... $Kn(m) = 64 * CK(j) + CK(j+1)...$ you will have a sort across sixteen characters only, and the resulting print-out might be a bit jagged. I have not tested the following formula, but it ought to work and will spread five characters across two integers, giving you the same twenty-character capacity:

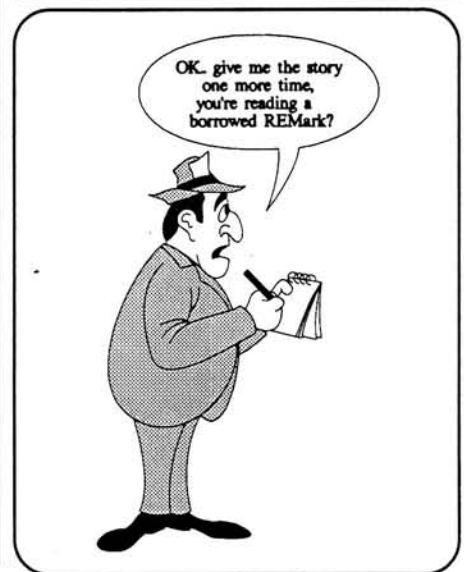
$$Kn(m) = 512 * CK(j) + 8 * CK(j+1) + CK(j+2) \setminus 8$$

$$Kn(m+1) = 4096 * (CK(j+2) \text{ AND } 7) + 64 * CK(j+3) + CK(j+4)$$

If you are using an earlier version than 3.0, you might find upgrading your compiler easier than modifying the program. I have made extensive use of the "DO... EXIT DO... LOOP" construction, and this came in with version 3. Some of the loops could be replaced with "WHILE... WEND"s, but the "EXIT DO" statements would need careful recoding.

Conclusions

Efficient sort algorithms, such as the Shell sort, run up to a hundred times faster on long arrays than simple sorts, for less than double the number of instructions. By spreading values across several arrays, you can sort up to 16,000 articles in core in a matter of minutes. Although disk sorting *per se* was outside the scope of this article, a disk file of up to 16,000 records, and allowing of random or binary access, can be ordered by a core sort of the keys. Efficient sorting is compact; the code example given in this article included a binary insertion sort, a fast batched insertion sort and a binary table look-up which all together — and including a special function for comparing multiple keys — took 89 lines of code.




```

-----
SUB LoadNuls
DO WHILE (LNuls < 200) AND (NOT EOF(1))
  INPUT #1, N$
  TS = ""
  FOR I = 1 TO LEN(NS)
    M = CSeq(ASC(MID$(NS, I, 1)))
    IF M > 0 THEN
      TS = TS + CHR$(M)
    END IF
  NEXT I
  Found = False
  LTab = .1
  HTab = LNuls
  DO WHILE LTab <> HTab - 1
    J = (LTab + HTab) \ 2
    IF Nuls$(J) < TS THEN
      LTab = J
    ELSEIF Nuls$(J) = TS THEN
      Found = True
      EXIT DO
    ELSE
      HTab = J
    END IF
  LOOP
  IF NOT Found THEN
    FOR I = LNuls-1 TO HTab STEP -1
      Nuls$(I+1) = Nuls$(I)
    NEXT I
    Nuls$(HTab) = TS
    LNuls = LNuls + 1
  END IF
LOOP
END SUB
-----
LoadTitles
Scan file TITLES and prepare sort files:
Note list c address for table F.
Scan forwards to beyond list comma.
Scan on to CR/LF; each time a word is split off, check it against Nuls$,
then add the character offset from F to table C, and the word itself to
K1, K2, K3 and K4.
-----
SUB LoadTitles (FPoint&)
LCat = 0
LFile& = LOF(1)
KPoint = 0, pointer to next slot in memory
KS = "", tables
DO WHILE FPoint& < LFile&
  KeyF& = FPoint&
  I = 0, number of characters in title
  TableLimit = FNMin(30, 16383-TCTab), don't overflow
DO
  GET #1,,K$
  FPoint& = FPoint& + 1
  LOOP UNTIL K$ = "" OR FPoint& => LFile&, skip over issue number
  Offset = FPoint& - KeyF&
  DO, load CT array with collating, difference between CT array and line
  GET #1,,K$
  FPoint& = FPoint& + 1
  CT(I) = CSeq(ASC(K$))
  I = I + 1
  LOOP UNTIL K$ = CHR$(13) OR I > 300, fill CT up to CR
DO
  GET #1,,K$

```

```

FPoint& = FPoint& + 1, skip LF
  ready for next text line
  J = 0, scan CT up to CR
  JL = 0, build up to 30 key entries
  Scan = True
  DO WHILE J < I AND JL < TableLimit, got the start of a word
  IF CT(J) => 0 THEN
    PWord = J, mark start for tables
    Words = ""
    DO WHILE CT(J) => 0
      IF CT(J) > 0 THEN
        Words = Words + CHR$(CT(J))
      ENDIF
      J = J + 1
    LOOP
    IF Words = "" OR NOT Scan THEN, punctuation signs or cutout
      NOGo = True
    ELSE
      CALL CheckNul Words, NOGo, see if not key word
    ENDIF
    CP = PWord + Offset + 1, starting byte position
    CL = J - PWord - 1, length in bytes
    IF CP > 255 THEN
      NOGo = True
    ENDIF, beginning of line
    IF NOT NOGo THEN
      CAT(LCat) = LCTab, add keyword to tables
      IF CL > 127 THEN
        CL = 127, limit max length of key
      ENDIF
      C(LCTab) = 256*CL + CP, combine both values
      F&(LCTab) = KeyF&, pointer to this title text
      MWord = False, keyword not scanned
      M = 0
      DO WHILE M < 20, length of sort key
        IF PWord => I THEN
          CK(M) = 0, beyond end of text
          M = M + 1
        ELSEIF CT(PWord) < 0 THEN, punctuation
          IF NOT MWord THEN, set off keyword from following text
            CK(M) = 0
            M = M + 1
            MWord = True, keyword scanned
          ENDIF
          PWord = PWord + 1, further punctuation is skipped
        ELSEIF CT(PWord) = 0 THEN, skip non-delimiting punctuation
          PWord = PWord + 1
        ELSE
          CK(M) = CT(PWord), add collating code to sort key
          M = M + 1
          PWord = PWord + 1
        ENDIF
      LOOP
      K1&(LCTab) = 16777216*CK(0) + 262144*CK(1) + 4096*CK(2) +
        64*CK(3) + CK(4)
      K2&(LCTab) = 16777216*CK(5) + 262144*CK(6) + 4096*CK(7) +
        64*CK(8) + CK(9)
      K3&(LCTab) = 16777216*CK(10) + 262144*CK(11) + 4096*CK(12) +
        64*CK(13) + CK(14)
      K4&(LCTab) = 16777216*CK(15) + 262144*CK(16) + 4096*CK(17) +
        64*CK(18) + CK(19)
      LCTab = LCTab + 1, next slot in main tables
      LCat = LCat + 1, next entry in CAT
      JL = JL + 1, count up to 30
    END IF
  ELSE, skip punctuation, check for specials

```



```

IF CT(J) = -4 THEN
  Scan = False
ELSEIF CT(J) = -3 THEN
  Scan = True
END IF
J = J + 1
END IF
LOOP
IF LCat > 40 THEN
  CALL SortTable
ENDIF
LOOP
IF LCat > 0 THEN ' last few keys to sort
  CALL SortTable
ENDIF
END SUB
*****
PrintIndex
  Print the file from the sorted tables
*****
SUB PrintIndex (PageTitles$)
  PrintMargin = 10 ' standard margins
  PrintKey = 40
  PrintEnd = 90
  PageLength = 61
  BoldOn$ = CHR$(27)+"E"+CHR$(27)+"G"
  BoldOff$ = CHR$(27)+"F"+CHR$(27)+"H"
  Mark$ = ""+BoldOn$+" "+BoldOff$+" "
  LenMark = Len(Mark$)
  WIDTH "LPT1:",255
  LPRINT CHR$(27)+"M" ' set 12 cpi
  LineNum = 99
  K$ = ""
FOR I = 0 TO LCTab-1
  Issue$ = ""
  LastKey$ = PrKey$
  SEEK #1,F&(AT(I))
  STX = 1
  DO
    GET #1,,K$
    STX = STX + 1 ' length of issue date
    IF K$ = " " THEN
      EXIT DO
    ENDIF
    Issue$ = Issue$ + K$
  LOOP
  CP = C(AT(I)) AND 255
  CL = C(AT(I))\256
  IF CP - STX > PrintKey - PrintMargin - 1 THEN
    PrLsp$ = SPACES(PrintMargin)
    Wrap = CP + 1 + PrintMargin - STX - PrintKey
  ELSE
    PrLsp$ = SPACES(PrintKey-STX-CP-1)
    Wrap = 0 ' extra margin, all left text printed
  ENDIF
  SEEK #1,F&(AT(I))+STX+Wrap-1
  PrLTx$ = INPUT$(PrintKey-Len(PrLsp$)-1,#1)
  IF CL > PrintEnd - PrintKey - Len(Issue$) - 4 THEN
    CL = PrintEnd - PrintKey - Len(Issue$) - 4
  ENDIF
  PrKey$ = INPUT$(CL-1,#1)
  LPrRTx = PrintEnd - PrintKey - Len(Issue$) - CL - 2
  PrRTx$ = ""
  IF LPrRTx > 0 THEN
    FOR J = 1 TO LPrRTx

```

```

      GET #1,,K$ ' rest of title
      IF K$ = CHR$(13) THEN
        EXIT FOR
      ENDIF
      PrTX$ = PrTX$ + K$
    NEXT J
  IF LPrTX - Len(PrTX$) > 3 AND Wrap > 0 THEN
    PrTX$ = PrTX$ + Mark$
    LPrTX = LPrTX + LenMark - 3
    Wrap = FNMin(Wrap,LPrTX-Len(PrTX$))
    SEEK #1,F&(AT(I))+STX-1 ' back to start for wraparound
    PrTX$ = PrTX$ + INPUT$(Wrap,#1)
  ENDIF
  IF Len(PrTX$) < LPrTX THEN
    PrTX$ = PrTX$ + SPACES((LPrTX-Len(PrTX$)) MOD 7)
    DO WHILE Len(PrTX$) < LPrTX
      PrTX$ = PrTX$ + " ..."
    LOOP
  ENDIF
  IF LineNum = 99 THEN
    FirstKey$ = PrKey$
    ' first time through, for footing
  ENDIF
  IF LineNum = PageLength-1 THEN
    LPRINT ' page footing
    LPRINT SPACES(PrintMargin);BoldOn$;FirstKey$ - "LastKeys;BoldOff$
    LPRINT CHR$(12);
    FirstKey$ = PrKey$
    ' for next page
  ENDIF
  IF LineNum > PageLength-2 THEN
    LPRINT SPACES(PrintMargin);BoldOn$;PrKey$;BoldOff$;PrTX$ "Issues
    LineNum = LineNum + 1
  ENDIF
  LPRINT PrLsp$;PrLTx$;BoldOn$;PrKey$;BoldOff$;PrTX$
  NEXT I
FOR J = LineNum to PageLength-1
  LPRINT
  ' final page - space down to footing
NEXT J
LPRINT SPACES(PrintMargin);BoldOn$;FirstKey$ - "PrKey$;BoldOff$
LPRINT CHR$(12);
END SUB
*****
SortTable
  Linear sort of CAT, then
  Batched binary insertion sort of AT
*****
SUB SortTable
  FOR I = 1 TO LCat-1 ' linear insertion sort of about
    Temp = CAT(I) ' 40 values in batch table
    J = I - 1
    WHILE J > 0 AND FNKeyComp(CAT(J),Temp) = 1
      CAT(J+1) = CAT(J)
      J = J - 1
    WEND
    CAT(J+1) = Temp
  NEXT I
  FOR I = LCat-1 TO 0 STEP -1
    LTab = -1
    HTab = TCTab
    DO WHILE LTab <> HTab - 1
      J = (LTAB + HTab) \ 2
      IF FNKeyComp(CAT(I),AT(J)) <= 0 THEN

```



Baylor HomeCare's Linda Embry, R.N., uses a MinisPort notebook while seeing a patient. Photo: Thom Gabrukiewicz, Baylor University Medical Center

PCs Assist Home Health Nurses in Patient Care

David Dalton
Manager, Corporate Communications
Zenith Data Systems

Stethoscopes and thermometers aren't the only equipment home health nurses at Baylor University Medical Center carry when they visit homebound patients. They now tote MinisPort HD notebook personal computers (PCs), too.

The nurses use the laptop PCs to increase the time available with patients and reduce the amount of time spent on paperwork, according to Janet Neff, R.N., B.S.N, clinical computer educator for the Baylor HomeCare program in Texas.

"Many of our patients receive Medicare benefits," Neff said. "Because HomeCare is reimbursed by the government for our services, we must complete numerous forms each time we see a patient."

Before they began using the MinisPorts, Neff's colleagues would return to the program's Dallas office several times a week to fill out paperwork, losing valuable time with patients, she said. With the laptops and a software program developed by Baylor's computer services department, the nurses can complete the necessary information and send it via modem from their homes or a patient's home directly into

Baylor HomeCare's billing system.

The software program also is designed to help nurses think through each patient's situation and assist in gathering information, according to Neff.

The program first asks for the patient's name, identification number and vital signs. It then queries the nurse about the patient's cardiac, respiratory and other systems. The program does not require nurses to enter large amounts of data, but rather to select from among several choices for each item.

The program also includes prompts that assist nurses as they teach patients about their conditions or medical problems.

Some patients don't feel comfortable with the high-tech approach to patient care, Neff said. "In those cases, the nurses leave the PCs in their cars and enter data after leaving the home," she said.

Neff estimates that before using the laptops, Baylor's home care nurses spent between 10 and 15 minutes recording information after each patient visit. Now, it takes three to five minutes and doesn't require the nurse to return to the office to submit documentation. "With each nurse

seeing an average of five to seven patients a day, the amount of time spent on paperwork added up quickly," she said.

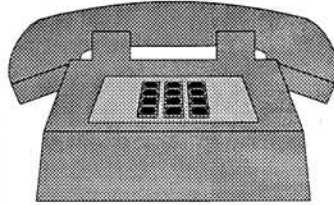
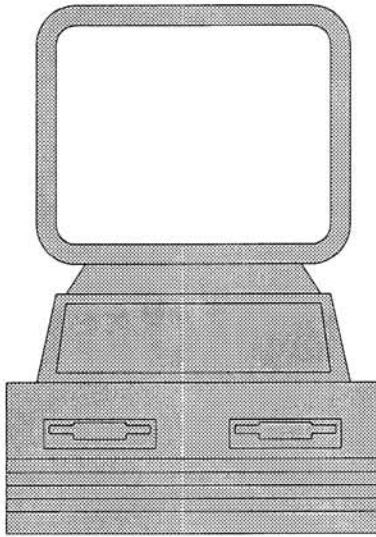
In addition, the laptops allow the nursing staff to spend less time developing patient treatment plans. What previously took 25 or 30 minutes per patient can now be done in 10 to 15 minutes. And, with instant access to patient information, the program's office staff can more quickly process bills.

Twelve MinisPorts are now in use. Neff's long-term goals are to have all 50 home care nurses equipped with PCs and to automate more Medicare forms. Eventually, she hopes to transmit billing information electronically to Medicare.

The laptops have had an additional, unexpected benefit, according to Neff: increased job satisfaction. "With a critical shortage of nurses today, retaining staff is a major concern for us," she said. "The PCs have given our nurses something new and challenging to learn and have made their jobs easier. They can now do more of what they are trained to do — nursing — rather than paperwork." ❄

Troubleshooting

Modem Problems



Robert C. Brenner
9282 Samantha Court
San Diego, CA 92129

© 1992 Brenner Information Group

Many things can impact modem data transfer—cables, switch settings, software, standards, and protocols. But, most communication problems are caused by operator error, or by incorrect hardware or software configuration.

When you attempt to link two geographically separated computers using modems at each end, a complex set of inter-related operations occur. Although transparent to you, these operations involve the way each computer and modem works, how the cable between each PC and modem is wired, the physical way that the dial-up telephone system moves information, and how the two parties have agreed to transfer data (standards and protocols).

Some modems incorporate self tests that check the system during power up and when troubleshooting. These can usually tell you if the modem or comm line is the problem source. Some modems include remote diagnostic capability so testing can occur without being physically present. External modems usually have status indicator lights that visually show the configuration and progress of a data transfer. And comm programs and operating system software can generate error messages to help isolate and identify a problem in getting information across.

Before you initiate a modem session, you should be convinced that the cables are connected properly and that interface cards are secured tightly in their correct sockets. You should also have an understanding of what interface protocol and transfer speed the other end can use.

Carefully follow the setup instructions when you install comm software and connect a modem. Some computers differ in

how they handle serial data through the RS-232 ports. The communications software can introduce unique requirements. For example, there are over a dozen variations of the XModem protocol. Yet many programs just label their version "XModem." One version of XModem may not be compatible with another version of XModem. The XModem byproduct XModem DOS is sometimes labelled just XModem in some comm programs.

Because of space limitations, this article assumes that you've successfully used your modem before, that your modem and hardware have been properly configured and problems such as switch settings, adapter card configuration, serial port assignments, and cable pinouts have already been handled. Therefore, we'll assume proper installation and checkout.

Now, when your modem communication session won't work, you know that nothing has been changed in the hardware setup, and you can begin a logical process to find and correct the problem.

Clues include what you see (or don't see) on your computer screen or on your modem status lights (if using an external modem) and by what you hear (or don't hear) from the modem speaker.

Just like searching for "Carmen Sandiego," you must closely analyze each clue (symptom) and understand the problem in general terms. Note any sign of activity (e.g., flickering or de-activated LEDs, screen error messages, etc.). Then isolate the problem. Go from the visual to the audible to the testable. If you have an external modem, observe the status lights. Listen for proper disk operation, dial tone, or connect sounds. Separate the problem from those functions that do work cor-

rectly. Note what is and is not being affected. Determine what happens just before something goes wrong. Check the modem and software manuals for correct settings. Try changing the parameters (baud rate, parity, number dialed, phone cable, etc.) one at a time to see if you can get the desired outcome.

When you think you've isolated and corrected the problem, confirm the correction by repeating the failure conditions and verifying that the same problem symptoms occur. Then, correct the problem, note it for future reference, and return the system to full operation.

Because a communication system is extensive and complex, many things can impact successful data transfers. Problems can occur at either end of the link. Often, the modem compensates for system limitations without you even knowing it. Noise on the telephone wires may cause your modem to automatically fall back to a slower speed. The inability of the modem at the other end to handle advanced data compression or error detection and correction protocols may cause your modem to automatically revert to the best transfer protocol conditions possible during that session. Yet there are some problems that the modem and software cannot correct without your intervention.

Try dealing with a problem at your end by confirming the protocol and data format and then terminating and restarting the session. You can do this with most modems without breaking the telephone connection. Use all of the information available.

The status light indicators on the front of an external stand-alone modem are invaluable as diagnostic troubleshooting aids.

Note the auto-answer (AA), send/transmit data (SD/TD), receive data (RD), and the high speed (HS) indicators. These lights show when the auto-answer software is functioning properly, when data is moving, and when the transmission is at a slower speed than that possible. The RD and TD lights flicker (rather than glow) as information flows into or out of a modem. A flickering RD light with nothing showing on your computer screen can indicate a failure in the serial link between the modem and the computer. The carrier-detect (CD) and off-hook (OH) lights indicate when a caller has disconnected or a connection has been broken. Noise on the line can cause the HS or CD indicators to go out.

Sounds can also help you locate a problem. A strange tone just before a session is terminated can mean a Call Waiting signal disruption. A pre-recorded message heard on your modem speaker when you dial from your keyboard can be a wrong number, or an incorrect dialing sequence such as forgetting the comma on a "dial 9 first" line.

The computer monitor's screen display is another good source for clues.

Garbage characters on your screen can be caused by transfer speed, duplex, and parity mismatch settings, a loose cable, or even by someone picking up an extension phone during a session. Missing characters can be caused by flow control mismatch, incorrect transfer speed, a disabled interrupt on your serial card, or an incompatible error control scheme in your comm program. If three to six characters are missing at the beginning of a line, suspect a slow screen scrolling routine. Lack of incoming data or key action display can be caused by duplex mismatch. This mismatch can also cause double characters. Double-spaced lines occur when both ends send a line feed command. Overwrite can occur when your program doesn't initiate a line feed. Incoming data can disappear off the right side of the screen when wraparound is not active. And boxes can appear at the beginning of each line of text when the incoming data contains line feed or other control characters that are meaningless to your comm program.

An inability to operate at the modem's fastest speed can be caused by the telephone line, your software, or the modem at

the other end. A file transfer that locks up your system can be caused by incompatible protocols or insufficient free space in your computer. If a download aborts with a CRC, Checksum or Mismatch Error, suspect a protocol problem, although CRC can also be caused by a noisy line. Keep a list handy of the protocol error messages associated with your comm program so you can take appropriate action when they occur.

Your modem's built-in diagnostics can check dialing and ring operation, the RAM and EPROM inside the modem, the phone/modem switching ability, and even the connected telephone line and serial port cables.

Modem communication problems can be challenging. But they CAN be successfully corrected.

Modems Made Easy \$19.95
 Brenner Information Group (+\$3/S)
 9282 Samantha Court
 San Diego, CA 92129
 (619) 538-0093 ✱

Continued from Page 20

and label it. Normally, I would do the following: (line 1) COPY CON PRN (line 2) This file is called MYBATCH.BAT (line 3) CTRL-Z (hold the Ctrl key down and press Z). That would look like this:

```
copy con prn
This file is called MYBATCH.BAT
^Z
```

This would immediately write the second line (This file is called MYBATCH.BAT) on the printer, and then stop, without ejecting a page. I could now write:

```
copy mybatch.bat prn
and the entire batch file would be printed, under the label.
```

If you want to do this on the serial printer, however, you have to make a small but important change, and write COM2 (or COM1 as the case may be) instead of PRN:

```
copy con com2
This file is called MYBATCH.BAT
^Z
```

Getting it All Together

One more problem remains, but one which is easily solved. How to write German on the serial printer, and be able to use a mouse. All you have to do is combine the batch file from my previous article, the batch file for German and the batch file for the serial printer:

```
@echo off
cd \german
mousesys
msysmice
\dos\MODE CON CP PREPARE=((850)
c:\dos\EGA.CPI)
\dos\mode con cp select =850
set pr=!pr.bro
mode com2: 12,N,8,1,B
ed %1
\dos\mode con cp select=437
set pr=!pr.def
```

Naturally, you can name this file anything you like, but I have chosen to name it GERBRO.BAT, since it reminds me that I am about to write in GERman on my

BROther printer.

Software products discussed in this article:

WordStar
 WordStar International
 201 Alameda del Prado
 Novato, CA
 (415) 382-0606

DESQview
 Quarterdeck Office Systems
 150 Pico Boulevard
 Santa Monica, CA 90405
 (213) 392-9701

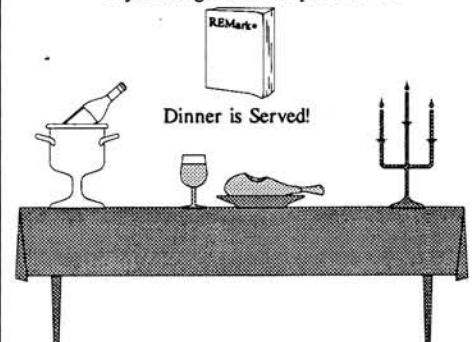
PC-Write
 Quicksoft, Inc.
 219 First Avenue N +224
 Seattle, WA 98109
 (206) 282-0452 ✱

Continued from Page 39

```
HTab = J
ELSE
  LTab = J
ENDIF
LOOP
FOR K = TCTab-1 TO HTab STEP -1
  AT(K+I+1) = AT(K)
NEXT K
AT(HTab+I) = CAT(I)
TCTab = HTab
NEXT I
LTab = 0
TCTab = LCTab
END SUB
```

* move table down several slots
 * add in value from batch table
 * mark new end of sorted unshifted
 * temporary to first free slot ✱

If you hunger for Computer news...



By Your Command

Chapter 2

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

Pushing the Envelope

Introduction

It slides smoothly out of the printer, as if on cue. Formatted, spell-checked, PROFESSIONAL in appearance; certainly neater than anything you could write longhand. Satisfied, you scrawl your signature, carefully fold the page, slip it into an envelope... and address it with a pen!

What's wrong with this picture? For starters, there's the mixed message you are sending your correspondent: "I produce quality products; up to a point." New wine in old bottles caused problems for vintners of old; does it make any more sense to work diligently on both contents and appearance of a message, then rob from its overall effect by enclosing it in a handwritten envelope? Cursive writing has an honored place in the world of correspondence, but I'm not sure that inconsistent appearance makes much of an impression on your recipients.

Point out this inconsistency to a colleague, and you'll likely hear "You know, I've thought about (finding|buying) an envelope printing program somewhere, but I just haven't gotten around to it yet." Others may have a program which prints beautiful mailing labels, but only use it for large jobs because "it's so much trouble to pull the paper out of my Epson and run that mailing label stock through it for just this one address!" Depending on your model of printer, feeding individual envelopes can be just as disruptive as loading label stock. What's a user to do??

You could always buy envelope-printing capability; there are dozens of inexpensive products on the market that urge you to "throw away your typewriter!" But I like to think that REMark readers have a different attitude; "why buy when I can make it myself?" And that's what we're about in this

Chapter.

In this article, we'll take the "By Your Command" approach to solving this problem; we'll *build* a solution using tools you already have. Join me in piecing together envelope printing routines for Epson 9-pin dot matrix and Hewlett-Packard LaserJet laser printers using DOS, your word processor, GW-BASIC, and QuickBASIC (with a brief note about Turbo Pascal). Not all folks like to work in "formal" programming languages, but at least one of the approaches we'll take today should appeal to you. We'll break the printing problem into manageable parts, and solve each one in turn for each printer. As you watch the solutions unfold, focus on how you might adapt one of these to your specific combination of available tools and printer.

The following routines were all built and tested on a Zenith Z-248/12 with 3 MB RAM and 80 MB of hard disk space, though they'll easily run on my old Z-159 XT as well. I use an Epson FX-86e dot matrix printer and a Hewlett-Packard LaserJet IIIIP laser printer at home. The Epson is our four-year-old faithful workhorse, while the IIIIP was added recently to respond to the desktop publishing side of Cathy's new business. You likely have a similar or compatible printer available, so choose the model that fits your situation, and let's begin!

Defining the Problem

There are two common envelope sizes we deal with in U.S. correspondence; standard (6-1/2 inches wide, 3-5/8 inches high) and legal-size, or "commercial" (9-1/2 inches wide, 4-1/8 inches high). Placing your return address 1/2 inch down and 1/2 inch in from the upper left-hand corner of the envelope gives a readable block on

both sizes of envelope. Placement of the recipient address block varies for these two types of envelopes; for legal size envelopes, I like to start printing two inches down from the top and 4-1/2 inches over from the left-hand edge. On standard envelopes, I prefer to start 1-3/4 inches from the top and 2-1/2 inches from the left-hand edge. Since we'll be dealing with two different printers and four approaches for each, let's simplify at this point by focusing on commercial-size envelopes. If you prefer using smaller envelopes, a little experimentation with the instructions given below will get you the desired result.

Getting the envelope into your printer can be simple or tricky, depending on your model. On my Epson dot matrix printer, I lift the tractor cover, open the pinfeed covers, pull the paper release lever and the paper bail lever to the front, and remove the continuous feed paper I normally print on. Returning the paper release lever to its original position, I raise the paper guide, slide the left-edge tab 1-3/4 inches to the right, and slide an envelope down around the roller (applying firm pressure as I turn the paper feed knob) until the top edge is just aligned with the printhead. Close the tractor feed covers at this point to avoid interference with the envelope as the printer feeds it up during printing.

That's the old technology way to do it, of course. Many modern dot matrix printers have special handling for envelopes that reduces the above steps considerably. LaserJet printers have an optional envelope feeder that greatly simplifies the process; fill the tray with envelopes and your loading is done! For those of you with access to a LaserJet IIIIP without an envelope feeder, HP has provided a straightforward way to load envelopes for manual

feed in the "Multi-Purpose tray".

First, you need to install the Front output tray. This tray doesn't hold as much paper, envelopes, or label stock as the top (standard) tray, but reduces curl in the media used and helps prevent paper jamming. After installation, flip the output tray selector down. Stack envelopes address side down with the flap opening to the right of the tray. Slide the paper width adjuster over for a snug fit, and you're ready to print!

Telling Your Printhead Where To Go

The key to proper placement of your return address (if desired) and the recipient address on the envelope lies in controlling the movement of the printhead in your printer. Both Epson and Hewlett-Packard printers have extensive command sets that allow precise print positioning. Our programming language examples will make use of these commands. But getting useful results can be as easy as building an image of what you want the envelope to look like and sending that image to the printer. Both the DOS and the "word processor" approaches we'll take use this concept, which you may find more straightforward than learning "Yet Another Command Set."

Issuing Commands (DOS)

Figure 1 is a listing of a short file built to print a single envelope. The return address block starts in column 1 of the first 3 lines, and is followed by 5 <RETURN> keystrokes, then the recipient address, which is indented 40 spaces. You could create a similar file using an ASCII editor (like DOS 5.0's EDIT or PC WatchWord, or even <gulp> EDLIN), or just build it from the

```
COPY CON ADDRESS.TXT
return address line 1
return address line 2
return address line 3
(5 <RETURN>s )

... (40 spaces)  recipient address line 1
                  recipient address line 2
                  recipient address line 3
<CTRL-Z>
```

Figure 1

DOS command line by typing

My Epson printer is connected to the second parallel port (LPT2), so printing this file once I have an envelope loaded is as easy as typing COPY ADDRESS.TXT LPT2. You'll get default top and left margins of half an inch and the font and quality you select on your printer's front panel. Our default settings print draft quality Courier font at 10 characters per inch (cpi); I chose NLQ font from the font panel, and the result resembled Figure 2.

Printing to the IIIP requires a bit more setup, mostly because envelopes feed into

O'Connor
848 Fenske Drive NE
Olympia, WA 98506

Jim Buskiewicz, Grand Looma Looma
Zenith User's Group
PO Box 217
Benton Harbor, MI 49023-0217

Figure 2

O'Connor
848 Fenske Drive NE
Olympia, WA 98506

Jim Buskiewicz, Grand Looma Looma
Zenith User's Group
PO Box 217
Benton Harbor, MI 49023-0217

Figure 3

a LaserJet in landscape orientation and use the manual feed option if (like me) you haven't purchased an automatic envelope feeder. Your LaserJet manual has complete descriptions of working with the Control Panel to prepare the printer for envelope printing. For the IIIP you take the printer OFFLINE, press MENU until JOB SIZE appears, press + 4 times to select COM10 (shorthand for commercial-size envelopes), save this with ENTER, press MENU until

ORIENT appears, press + to select LANDSCAPE and ENTER, press MENU until MAN FEED appears, press + to select ON, and press ENTER. Put the printer back ONLINE, and then type COPY ADDRESS.TXT LPT1 (if your LaserJet, like ours, is connected to LPT1, the primary parallel port). The printer will go OFFLINE and display the message ME FEED COM10. If you've successfully loaded envelopes into the Multi-Purpose

tray as above, simply press ONLINE and your envelope will print with the default font (often Courier at 10 cpi).

This is a good time to note that envelope quality is important to the health of your laser printer and the success of your efforts. The heat from the printer's fuser will force envelope flap glue to stick together in cheap envelopes, and you may even experience wrinkles along the top edge if the paper is of poor quality. Hewlett-Packard manuals give detailed advice about the proper types and weights of materials to send through their printers, advice which

you should definitely take!

Issuing Commands (Word Processor)

In some ways the task of envelope printing is made a little easier by a decent word processor (we use WordPerfect Corporation's LetterPerfect). I first built a file looking very much like Figure 1 and sent it to the Epson printer. But I discovered that the default spacing for top and left margins was different when printing this way (more like 3/8 inch), so some experimentation was in order. Using a 10 cpi Sans Serif font, I found that the following procedure worked best for the Epson: set top and left margins to 1/8-inch, type the return address, change left margin to 4-1/8 inches, press <RETURN> six times, type the recipient address, and send the "page" off to the printer. The 3/8-inch offset combined with these margin settings will place the address blocks very close to the desired positions.

For LaserJet printers, LetterPerfect makes it easier by including several predefined forms in the Paper Size/Type section of the FORMAT menu. Another difference you'll note when using a word processor for envelopes is that margins are specified exactly without concern for "offsets." I chose the Envelope form (9-1/2 X 4 inches), set base font to CG Times 10 point, set the left and top margins to 1/2 inch, typed a return address, changed the left margin to 4-1/2 inches, enlarged the font size to 15 point, typed the recipient address, and printed the result. We have LetterPerfect configured to expect Manual Feed for envelopes, so I was asked to press G to continue the manual feed operation.

The printer received the data, displayed ME FEED ENVELOPE and once I pressed ONLINE, delivered an envelope that looked like Figure 3.

Issuing Commands (GW-BASIC)

It's quite simple to duplicate the envelope-printing work we did from DOS by writing a BASIC program. For those of you with the GW-BASIC interpreter, I've listed a program in Figure 4 that will print our sample envelope correctly on any dot matrix printer that can emulate Epson printer codes. By way of review, CHR\$(27) repre-

```

10 REM GW-BASIC program to print envelopes
15 REM on an Epson FX-86e dot matrix printer
20 LPRINT CHR$(27);"@
30 LPRINT "O'Connor"
40 LPRINT "848 Fenske Drive NE"
50 LPRINT "Olympia, WA 98506"
60 FOR L=1 TO 5: LPRINT CHR$(10);: NEXT L
70 LPRINT CHR$(27);"1";CHR$(40)
80 LPRINT "Jim Buszkiewicz, Grand Looma
Looma"
90 LPRINT "Zenith User's Group"
100 LPRINT "PO Box 217"
110 LPRINT "Benton Harbor, MI 49023-0217"
120 LPRINT CHR$(27);"@
130 SYSTEM

```

Figure 4

```

10 REM GW-BASIC program to print envelopes
15 REM on a Hewlett-Packard LaserJet 3
20 LPRINT CHR$(27);"E";
30 LPRINT CHR$(27);"&l81ale10";
40 LPRINT CHR$(27);"&a5L";
50 LPRINT "O'Connor"
60 LPRINT "848 Fenske Drive NE"
70 LPRINT "Olympia, WA 98506"
80 LPRINT CHR$(27);"&a12r40L";
90 LPRINT "Jim Buszkiewicz, Grand Looma Looma"
100 LPRINT "Zenith User's Group"
110 LPRINT "PO Box 217"
120 LPRINT "Benton Harbor, MI 49023-0217"
130 LPRINT CHR$(27);"E";
140 SYSTEM

```

Figure 5

sents the ESCape code. <ESC-@> (ampersand) initializes an Epson printer by resetting the printer mode to default and clearing the printer buffer of any printable data. LPRINT commands are used to print the return address lines, followed by a loop to add five line feeds <CHR\$(10)>. The left margin is then reset 40 characters to the right with <ESC | CHR\$(40)>, where the letter is 'ell'. The address block is then printed, and the printer is re-initialized for whatever job needs to print next.

Adapting your BASIC program for the HP LaserJet is a bit more complicated. Luckily, the folks at Hewlett-Packard included a very clear example in their LaserJet IID manual about using their PCL (Printer Command Language) commands to select and print an envelope. The listing in Figure 5 is based on their sample BASIC program,

extended to include use of a return address block. The odd codes you see printed before the recognizable text are escape sequences that send specific PCL commands to the printer. Previous REMark articles, and even entire books have been written about using PCL; let's limit our discussion here to clarifying what these particular commands do.

The CHR\$(27)"E" is an <ESC-E> sequence, which resets LaserJet printers to their default settings (useful if the last person to use your machine sent any strange and exotic codes). The next two lines position printing for the return address, using commands from the 'l' ('ell') family for form and orientation, and commands from the 'a' family to set the left margin. An ampersand begins each string of commands from a particular family, and the final letter in the string is capitalized. '81a' chooses form=COM10 (the envelope form), '1e' sets the top margin to one line, and '1o' sets orientation to "landscape." In the 'a' family, '5L' sets the left margin to five spaces (or one-half inch, at ten characters-per-inch). The return address lines are then printed.

To reposition for the recipient address, the escape sequence of line 80 is used. '12r' resets vertical position to two inches down from the top of the envelope (at a default 6-lines-per-inch vertical spacing), and '40L' resets the left margin to the 40th column, or 4 inches from the left edge of the envelope. The recipient address is then printed, and a final <ESC-E> is sent to re-initialize the printer.

If your LaserJet is connected to LPT1, you can enter GW-BASIC, type in these lines (substituting perhaps real addresses), save your work, and type RUN to print an envelope with the default fonts and spacing.

Issuing Commands (QuickBASIC)

You can easily think of improvements worth making to that BASIC code; Figure 6 is a QuickBASIC listing of a version I use at work which is especially useful for doing multiple envelopes. A DO-WHILE loop is used to print as many envelopes as needed at one sitting, using the envelope feeder we have attached to our office LaserJet. This program first prompts the user to optionally enter their last name and up to 5 lines of recipient address block. The envelopes we use at work have return addresses stamped with the agency logo, and so this program prints the user's last name (if

provided) above that return address. After the '81a' and '1O' to select envelope form and landscape orientation, I use '9L' to set the left margin to line up with the logo.

The next step taken depends on the presence of a last name for return addressing. If one has been entered, the top margin is set to one line with the '1E' code, and then the next two escape series simply (!) set the font to Line Printer 16.67 cpi. (This small font is more matched in size to the logo we use.) The return address name is then printed, followed by the '10R' command from the 'a' family, which moves 10 lines vertically to prepare for printing the recipient block. (If no return name has been provided, the top margin is set to 12 lines instead.)

The next three complicated-looking escape sequences reset the left margin to 40 characters and the font to Courier 10 cpi. The recipient address lines are printed, followed by a reinitialization sequence <ESC-E>. The user then can choose to specify another envelope or quit the program.

If you use the Control Panel on your HP LaserJet to print out a sheet of all fonts available on your printer, you can discover other font code escape sequences you can use to customize this program to reflect your own taste in appearances. By experimenting with these and the precise character positioning possible on a LaserJet, you can customize this program to work nicely with the envelopes you use, with or without a logo.

What about our Epson users? By now you can see that the difference in approaches is related solely to the specific escape codes you direct to your printer. If you compare the Epson and PCL commands used in Figures 4 and 5, you can see how to adapt the QuickBASIC program to print to your Epson dot matrix printer.

Final Notes

My local Turbo Pascal expert had trouble developing a TP 6.0 program similar to Figure 6. It seems that the '12E' top margin command gets ignored, although the other commands in the 'l' family seem to work just fine. I leave this as an exercise for the Turbo Pascal fans out there; certainly this must be doable in your favorite language, so do it and share it with the rest of us via Buggin' HUG or a full-length article.

For others among you, the choice is simple: stick with the manual way, or try something new! Any of these approaches can improve the appearance of your envelopes, and you know what they say about first impressions. So pick a method, fire up your computer, and start pushing a few envelopes of your own!

Correction

'Very Simple Envelope Printer Program for HP LaserJet IID with Feeder.
'This version allows for a one-line return address.

```

OPEN "LPT1:" FOR OUTPUT AS #1
DO
  CLS
  raname$ = ""
  add1$ = ""
  add2$ = ""
  add3$ = ""
  add4$ = ""
  add5$ = ""
  PRINT : PRINT : PRINT
  PRINT TAB(16); "Envelope printer for HP LaserJet IID with Feeder"
  PRINT : PRINT TAB(21); "Maximum of five address lines allowed"
  PRINT : PRINT : PRINT
  PRINT TAB(13); "*****"
  PRINT TAB(13); "Use <CR> when address is done.... Use <CTRL>C to abort"
  PRINT TAB(13); "*****"
  PRINT : PRINT : PRINT

  LINE INPUT "Enter name for return address; else, just <CR>: "; raname$
  PRINT : PRINT

  LINE INPUT "First line of address: "; add1$
  IF add1$ = "" THEN GOTO printe
  LINE INPUT "Second line of address: "; add2$
  IF add2$ = "" THEN GOTO printe
  LINE INPUT "Third line of address: "; add3$
  IF add3$ = "" THEN GOTO printe
  LINE INPUT "Fourth line of address: "; add4$
  IF add4$ = "" THEN GOTO printe
  LINE INPUT "Fifth line of address: "; add5$

printe:
  PRINT #1, CHR$(27); "E";
  PRINT #1, CHR$(27); "&l81a10";
  PRINT #1, CHR$(27); "&a9L";

  IF raname$ <> "" THEN
    PRINT #1, CHR$(27); "&l1E";
    PRINT #1, CHR$(27); "(OU"; CHR$(27); "(s0p12.00h10.0v0s0b8T";
    PRINT #1, raname$
    PRINT #1, CHR$(27); "&a10R";
  ELSE
    PRINT #1, CHR$(27); "&l12E";
  END IF

  PRINT #1, CHR$(27); "&a40L";
  PRINT #1, CHR$(27); "(8U"; CHR$(27); "(s0p10.00h12.0v0s0b3T";
  IF add1$ <> "" THEN PRINT #1, TAB(26); add1$
  IF add2$ <> "" THEN PRINT #1, TAB(26); add2$
  IF add3$ <> "" THEN PRINT #1, TAB(26); add3$
  IF add4$ <> "" THEN PRINT #1, TAB(26); add4$
  IF add5$ <> "" THEN PRINT #1, TAB(26); add5$
  PRINT #1, CHR$(27); "E"

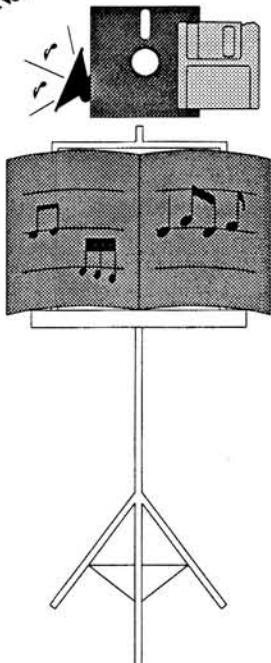
  PRINT : PRINT : PRINT "Another envelope? (Y/N) "
  More$ = UCASE$(INPUT$(1))
LOOP WHILE More$ = "Y"
END

```

Figure 6

In the July, 1991 REMark, I inadvertently gave the wrong phone number for Central Point Software's Technical Support in an article about CP Backup. The current number is (503) 690-8080. Unfortunately, the number I gave is for Public Relations, and although they're pleased to talk with customers, they should not be your first choice for questions concerning installation and use of Central Point products! ❄

New Software Product!



The Electronic Clavier
P/N 885-6016

Continued from Page 11

I can call at night when the rates are lowest, and the lines are rarely busy. Messages that I leave are usually answered within 24 hours — difficult questions may take two or three days. The answers are always thorough and competent. Reading the questions and answers from other users is like a mini-tutorial. One feature I particularly like is that the BBS will, on request, compress all the unread messages and transmit them to me in "Zipped" format saving time and money.

The International Smartnet Network also carries the "Desqview Product Sup-

port Conference." If you have a local Smartnet system in your area, you can leave and receive messages through them.

Desqview has spoiled me. When I run into a problem, I feel frustrated — like a dog on a leash. Contacting tech support and resolving the difficulty leaves me elated — like a lion tamer who has at last found the whip crack which will subdue the most ferocious of his big cats.

Products mentioned:

DESQview 386
Quarterdeck Office Systems
150 Pico Boulevard

Santa Monica, CA 90405
(310) 392-9851

TAME
PowerSoft, Inc.
P.O. Box 956338
Duluth, GA 30136
PowerSoft BBS (404) 928-9294

Info Select
Micro Logic Corporation
POB 174
Hackensack, NJ 07602
(201) 342-6518 ❄

Making the Connection

Selecting the Correct PC Cables

William Wills
202 Meadowbrook Avenue
Boardman, OH 44512-3004

When it comes to selecting cables to connect our computer with our peripherals, there usually isn't much to think about. We just look in a catalog, pick it, order it, and connect it.

Before we run out and buy those cables, we should ask ourselves: Is that printer cable going to run more than 15 feet? Will the cable be running through an area with electromagnetic equipment? Are the cables going to ruin our neighbors' TV and radio receptions?

We also should consider where we're going to put our peripherals and computer. Will the printer be sitting close to a wall where space is tight? Are we placing the computer on the desk with the keyboard and monitor?

Finally, what is the cable going to connect? Will it connect a serial printer, parallel printer or modem? Is it being used to transfer data between two PCs? How many pins do we need at the connector?

Cables

PC cables are made up of two parts; the cable and two connectors, one on each end of the cable. The cable is made up of several, either 24- or 26-gauge, insulated stranded copper wires called lines. These lines are covered with an outer sheathing of PVC. The lower the gauge of the line, the thicker and stronger the cable and the less resistance to the data being sent through them.

Besides the PVC cover, some cables have two or more coverings of electrically conductive material — either a covering of braided copper stranding or a layer of aluminum/mylar foil or both. These cables are known as shielded cables.

A shielded cable has two functions. The first is to contain electromagnetic fields generated within the cable that could cause trouble as radio frequency interference (RFI). The second function is to protect the data carried within the cable from outside

electromagnetic interference (EMI), which can come from fluorescent lights, appliances and even a laser printer.

Not all cables are shielded. Another type is known as ribbon cable. Most ribbon cables are 28-gauge or less. You will recognize ribbon cable as the cables used for connecting the disk drives inside your computers. Ribbon cables are vulnerable to EMI and won't carry a signal for a long distance. They are great for the inside of your computer because of the short lengths required and the computer provides the protection and shielding.

Connectors

Without the connectors on the ends of the cable, the cable would not be of much use. Most PC cable connectors are one of three types. The D-shell connector, which comes in different sizes and numbers of pins. The Centronics connectors, which are used to connect parallel printers. A DIN connector, the tubular plug that attaches keyboards to the computer.

The size and type of connectors are set by industry standardization. These different connectors are used to handle the specific needs of various peripherals. They also make it difficult to plug the wrong cable into the wrong connector.

The gender of a connector, whether it's male or female, also must be known. Most PCs use a male D-shell connector (with pins) and parallel printers use a female connector (with sockets for the pins). The cable connector that plugs into them must be of the opposite gender.

A connector must provide a good means of electrically connecting the lines in a cable to the circuits inside the PC and peripheral. You will find that most connectors have gold-coated pins and sockets. This coating provides excellent conductivity and doesn't tarnish.

The connector also must provide a strong mechanical connection between

the cable and the peripheral. A good soldered connection between the lines in the cable and the pins or sockets in the connector will provide this strength. The real strength in the cable should be between the sheathing and the connector hood, not the solder connection.

Most cables have two styles of connector hoods. One style has an assembled hood. They are two pieces of plastic or metal screwed together. The other style of cables comes with a one-piece molded hood and a flexible stress relief.

D-shell cables that attach to the PC or the peripheral have two screws or thumb-screws (which eliminate the need for a screwdriver) that make the connection secure and ensure a good ground for the cable shielding. The Centronics connector that attaches to the printer is usually held on by two spring clips.

Knowing how a cable is constructed helps you judge the quality of the cable, but you also must know about the different kinds of cables. You should know the difference between a serial cable and a parallel cable.

Serial Cables

Serial cables are so named because they carry data one bit at a time over a single data line. These cables connect more types of peripherals than any other type of cables. Modems, mice and serial printers all use serial cables.

The standard serial cable has a connector that is a 25-pin D-shell, also called a DB-25 connector. You will also find 9-pin D-shell or DB-9 connectors. The reason for this is that only nine or fewer lines are used with any IBM-compatible computer. Most of your mice and scanners use a DB-9 connector.

When you order your serial cable, you must know whether you'll be using it with a 9- or 25-pin port, and if the port is male or female. The IBM standard is male for the PC

connector and female for the cable. Most serial peripherals use DB-25 connectors, but the gender varies. When reading an ad for cables, the first gender refers to the end that plugs into the computer.

A modem cable is a serial cable and the connectors are the same DB-25; but you cannot use a serial printer cable with your modem. This is because there are two types of serial peripherals, data terminal equipment (DTE) and data communications equipment (DCE). The lines in each are configured differently. A serial printer is usually configured as DTE, while modems are DCE.

If you are thinking about connecting two computers between their serial ports, you will need a null modem cable. This is a serial cable, but again it has a different configuration. I am not going to get into how these cables are wired, since that would be another article. Fortunately, you don't have to know the wiring details. You can specify the cable you need by the peripheral you will be connecting to your computer.

If you happen to have a serial cable just lying around, you don't need to buy a new cable if it's the wrong gender. You can purchase gender changers. These changers will let you change a male connector to a female connector or a female to a male. You can also purchase an adapter that will change a DB-25 to a DB-9 or a DB-9 to a DB-25.

Parallel Cables

There is normally only one type of printer cable configuration, a male DB-25 connector on the PC end and a male 36-pin Centronics connector on the printer end. The only problem that you might have with these cables are their lengths.

Parallel printer cables use digital TTL signals that range between 0 and 5 volts and don't carry as far as serial signals. Parallel signals are subjected to more EMI and RFI than serial signals, so the recommended length is between 10- and 15-feet for these cables. If you must run your cable a longer distance, it would be wise to purchase a signal booster.

You can buy parallel cables with Centronics connectors of either gender on both ends for use in hooking up printer switch boxes and buffers. A 90-degree Centronics connector that guides the cable directly out to one side rather than straight back is also available. You must specify right- or left-handed when ordering these cables.

Miscellaneous Cables

Monitor cables depend on the brand and type of monitor. There are two standard connectors at the PC end, a 15-pin D-shell connector for VGA and Super VGA monitors, and a 9-pin D-shell connector for

CGA and EGA. Most monitor cables are connected directly to the monitor, but if yours is not and you need a cable your best bet is to go to the manufacturer or the vendor that sold you the monitor.

If you are not going to put your computer on the desk, you will need extender cables in lengths of 6- or 10-feet. You can also purchase these cables in a kit, one cable for the keyboard, one for the monitor and a monitor power cable.

When you purchase a keyboard extender cable make sure that it is a DIN-5 connector and not a DIN-9 connector that is used on an IBM PS/2. If you don't have a choice, you can also get an adapter that will change the DIN-9 connector to a DIN-5 connector.

When you are ready to purchase your cables, purchase them from a vendor that specializes in cables. You will also find cables that are UL listed and some vendors will even offer a life time warranty. But the most important thing is don't be afraid to ask questions if you are not sure.

Mail-Order Vendors for Cables

C-Gate International
3529 Ryder Street
Santa Clara, CA 95051
(408) 730-0673
Hours: Mon to Fri: 7am-5pm;
Sat: 8am-12pm (Pacific Time)
Accepts VISA and MasterCard
Free Catalog

Cables To Go
26 West Knottingham, Suite 200
Dayton, OH 45405
(800) 225-8646
Hours: Mon to Fri: 8:30am-5:30pm
(Eastern Time)
Accepts VISA and MasterCard
Free Catalog

National Computer Accessories
1510 McCormack Street
Sacramento, CA 95814
(916) 441-1568
Hours: Mon to Fri: 7am-5pm (Pacific Time)
Accepts VISA and MasterCard
Free Catalog

Power Plus
4932 Sharp Street
Dallas, TX 75247
(800) 878-5530
Hours: Mon to Fri: 8am-10pm;
Sat: 11am-4pm (Central Time)
Accepts VISA and MasterCard
Free Catalog

Glossary of Cable Terms

9- to 25-Pin Adapter: A fitting that lets a cable with a 25-pin female connector attach to a 9-pin male serial port. Other configurations are available.

Centronics: A connector that is com-

mon on all PC-compatible parallel printers. Standard parallel printer cables have a Centronics connector at one end and a DB-25 connector at the other end.

D-Shell: The most common type of connector used on PC cables. D-Shell connectors with two rows of pins starting with the letters DB followed by the number of pins. DB-9 and DB-25 are the most common connectors.

EMI: Electromagnetic Interference can be caused by appliances, fluorescent lighting, office machinery and laser printers. EMI can modify the data carried by a cable so that it cannot be recognized at the receiving end.

Gender: A term used to determine the different types of sockets. A female connector has pin sockets, while a male connector has pins. To connect, two connectors must be of the same type but of opposite gender.

Gender Changer: An adapter that will let you change the gender of a cable. A male to male adaptor, for example, will change a female connector to a male connector. You will find gender changers for DB-9 connectors, DB-25 connectors and Centronics connectors.

Hood: Two pieces of metal or plastic that is fastened by screws over the connector to protect the solder connections and to reduce the strain on the connector. Most cables sold have their connector covered with a one piece molded plastic hood.

Null Modem Cable: A special serial cable that will allow you to connect two PCs to transfer data between them. This cable cannot be used with another type of serial equipment.

Parallel Cable: A cable that connects a PC's parallel port with a parallel printer. The cable has a DB-25 connector at the PC end and a Centronics connector at the printer end.

Ribbon Cable: A cable with insulated wires arranged flat in a single layer. Ribbon cables has no shielding, but it is flexible. Most ribbon cables are used inside the PC to connect disk drives.

RFI: Radio Frequency Interference refers to the electromagnetic energy emitted by electronics, computers, or other electrical equipment that is in the same frequency ranges used for communication broadcasting.

Serial Cable: A cable used to transmit data one bit at a time over a single data line, between two serial devices, such as a modem or serial printer.

Shielded: A covering of conductive material, such as copper braid or metal/plastic foil is often used to shield data lines in a cable from EMI. The shielding also prevents electromagnetic energy from escaping from the cable to become RFI. ✱

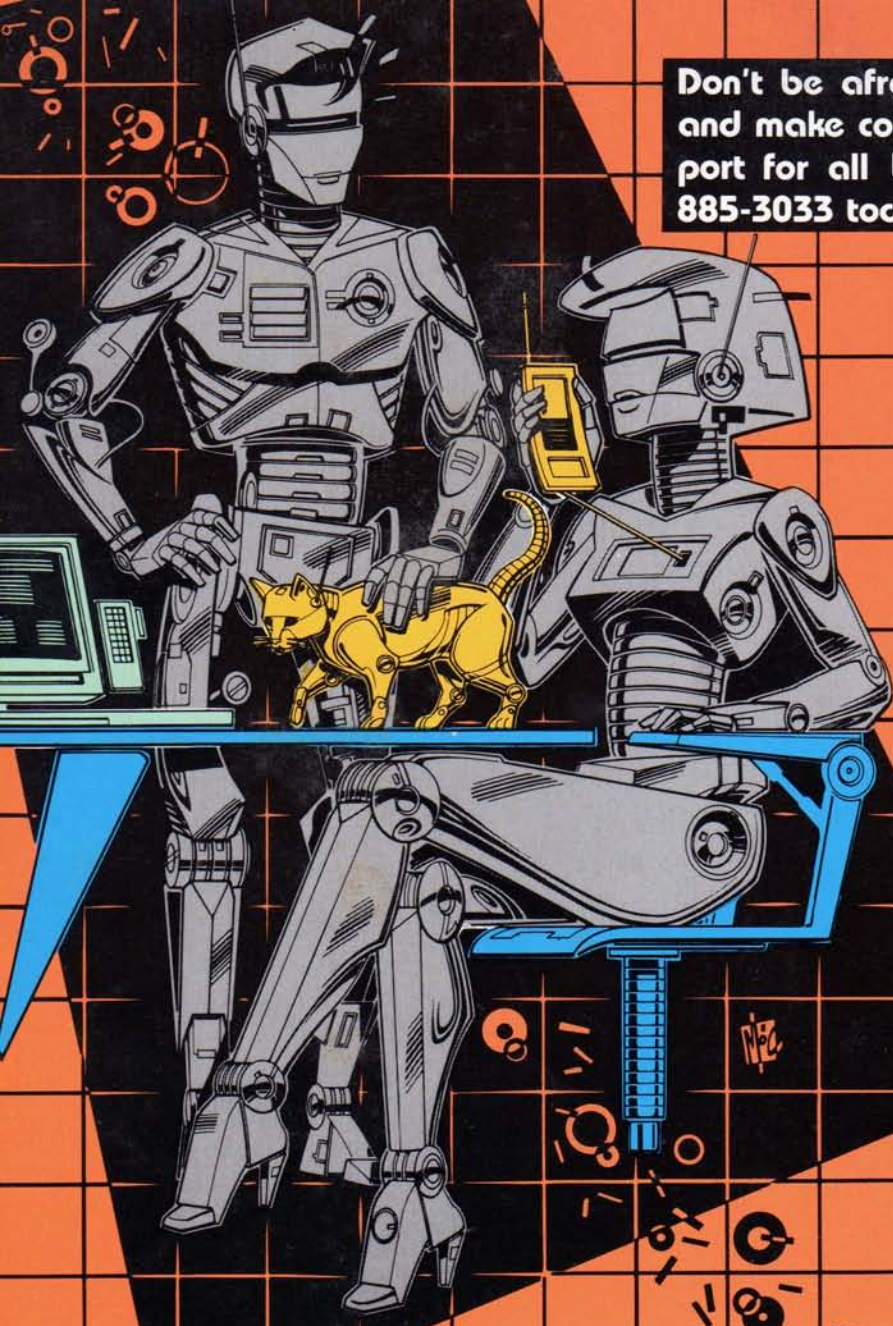
A decorative border of stylized yellow and orange flames surrounds the central text.

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 Laptops, order HUG Part number 885-3033 today.

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

Storage Buffer = 524288 Bytes
Storage Buffer Usage = 0 Bytes

Select Message (0-0), (F1) To List, Anything Else To Abort --> _
F1-Hlp F2-Msg F3-Bufr F4-Sav F5-Snd F6-Cfg F7-Clr F8-Print F9-Exit COM
```

```
HUGMCP Configuration Help #1
1- This Section Allow The Baud Rate To Be Changed, Depending Upon Which
   Mode You're In. Normally It Will Allow Higher Baud Rates
   For The Moment Than It Would Allow Under Normal Operation.
2- This Section Allow The To Change The Word Parity. Normally You
   Can Choose No Parity, Even Or Odd Parity. This Is Acceptable To Most
   Serial Systems, And It Is Also Necessary For XMODEM Protocol To Work Properly.
3- This Section Allow The Changing Of The Word Length. Normally The
   Length Should Be Set To 8 Data Bits. This Value Is Acceptable To Most
   Serial Systems, And It Is Necessary For XMODEM Protocol To Work Properly.
4- This Section Allow You To Enter Messages Which Can Be Automatically
   Sent With The (F1) Key. Up To 16 32-Character Messages Can Be Stored.
   Messages Are Formed In Lines (0) To 1599. Special This Section Can Auto-
   matically Be Used When The Program Is First Installed By Selecting The
   Proper Option During Setup.

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

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

Select #-C, (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: NONE
Modem Port Set To: COM1

F1-Hlp F2-Msg F3-Bufr F4-Sav F5-Snd F6-Cfg F7-Clr F8-Print F9-Exit COM
```

ZENITH
data systems 
Groupe Bull

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

POSTMASTER: If undeliverable,
please do not return.

\$2.50
P/N 885-2146