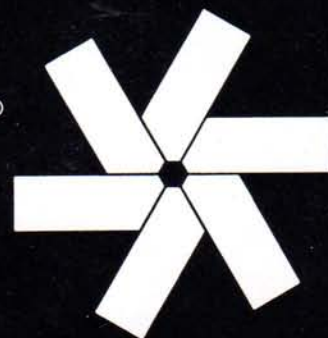


# REMark<sup>®</sup>

February 1992



The Official Zenith Data Systems Computer Users' Magazine







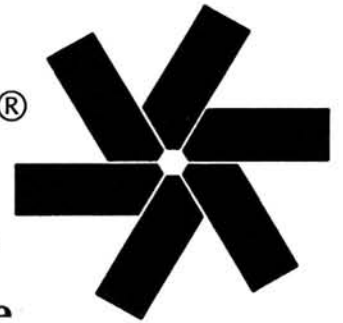
## *Share the Knowledge!*

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<sup>®</sup>

February 1992



The Official Zenith Data Systems Users Magazine

## EISA-Compatible LAN Server

*Nathan E. Baker* ..... 5

## The ZLS Workstation Series LANs Best Friend

*Nathan E. Baker* ..... 7

## Inside DOS 5.0... New Commands

*Mark Haverstock* ..... 9

## Lying to Your Mouse

*Daveed Shakar* ..... 13

## Trapping that Mouse

*David W. Lind* ..... 15

## Add Seven Hard Drives to Your System

*Frederick O. Smetana* ..... 23

## Getting Started With...

### Q&A

*Alan Neibauer* ..... 27

### Multi-Edit

*Dan Jerome* ..... 31

### Sorting, Searching, and KWIC Indexes Part 1

*John Day* ..... 37

### Words of a Feather

*James Hoffer* ..... 42

### Textbooks for the 1990's

*Frederick O. Smetana* ..... 43

## Advertising

Page  
No.

<i>FBE Research Co., Inc.</i> .....	14
<i>QuikData, Inc.</i> .....	26
<i>Surplus Trading</i> .....	22
<i>WS Electronics</i> .....	25

## Resources

Software Price List .....	2
Renewal Form .....	36
ZUG New Products .....	47



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

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

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

**Secretary**  
Lisa Cobb  
(616) 982-3463

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

**ZUG**  
**Software Orders**  
(616) 982-3463

**Contributing Editor**  
William M. Adney

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

**Contributing Editor**  
Robert C. Brenner

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

**To Locate your Nearest:**  
Dealer ..... 1-800-523-9393  
Service Center ..... 1-800-777-4630

	U.S. Domestic	APO/FPO & All Others
<b>Initial</b>	\$22.95	\$37.95*
<b>Renewal</b>	\$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

PRODUCT NAME	PART NUMBER	OPERATING SYSTEM	DESCRIPTION	PRICE
<b>H8 - H/Z-89/90</b>				
ACTION GAMES	885-1220-[37]	CPM	GAME	20.00
ADVENTURE	885-1010	HDOS	GAME	10.00
ASCIRITY	885-1238-[37]	CPM	AMATEUR RADIO	20.00
AUTOFILE (Z80 ONLY)	885-1110	HDOS	DBMS	30.00
BHBASIC SUPPORT 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 1	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	20.00
UDUMP	885-8004	HDOS	UTILITY	35.00
UTILITIES	885-1212-[37]	CPM	UTILITY	20.00
UTILITIES BY PS	885-1126	HDOS	UTILITY	20.00
VARIETY PACKAGE	885-1135-[37]	HDOS	UTILITY & GAMES	20.00
WHEW UTILITIES	885-1120-[37]	HDOS	UTILITY	20.00
XMET ROBOT X-ASSEMBLER	885-1229-[37]	CPM	UTILITY	20.00
Z80 ASSEMBLER	885-1078-[37]	HDOS	UTILITY	25.00
Z80 DEBUGGING TOOL (ALDT)	885-1116	HDOS	UTILITY	20.00
<b>H8 - H/Z-89/90 - H/Z-100 (Not PC)</b>				
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
<b>H/Z-100 (Not PC) Only</b>				
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
<b>H/Z-100 and PC Compatibles</b>				
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
<b>PC Compatibles</b>				
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.



# Zenith Data Systems' 486-Based LAN Server



---

---

# Zenith Data Systems' Z-486/33ET EISA-Compatible LAN Server

Nathan E. Baker  
Zenith Data Systems  
Technical Writer

This article is intended as part of an ongoing series dedicated to Intel 80486-based computer products produced by Zenith Data Systems. To preface the series, I'll talk a little bit about the 80486 and its primary features before I get into the specific hardware targeted by this installment.

## The Intel 80486 Microprocessor: An Overview

The 80486 microprocessor is an evolutionary step-up from the 80386. It incorporates over a million transistors in a single chip measuring a little over an inch and a quarter square. For you techies out there, heres an at-a-glance description of its primary features:

- 32-bit address bus
- 32-bit data bus
- real address mode and protected address mode operation
- 32-bit pipelined internal bus path
- internal floating point unit (80387 equivalent)
- internal 8K cache memory unit (4-way set associative)
- internal memory management unit (paged and virtual)
- 25 Mhz or 33 Mhz clock speed
- burst transfer speed of 106M per second

The 80486 uses the 80386 command structure, but includes extensions designed for its new features. These command enhancements and features give it better than twice the performance of an 80386 microprocessor.

The 80486 register set includes all of the registers found in 80386 and 80387 processors, which eliminates the need for an I/O cycle when processing floating point instructions. The internal floating point unit also incorporates new instructions and new error reporting modes which guarantee DOS compatibility.

## The Z-486/33 ET Server: An Anatomy Lesson

The Z-486/33 ET is an 80486-based computer that uses an EISA (Extended Industry Standard Architecture) bus. Internally, the computer is composed of a system board, a VGA video board, a hard disk controller board (SCSI-compatible), a power supply, floppy and hard disk drives, and a keyboard.

If we were to look at this computer as a living thing, the first thing we might do is decide where the heart and the brain reside. Neither one of these organs can exist for long without the other. Not many people would argue that the brain functions are taken up by the processor and memory circuitry. The heart then, would seem to be the clock mechanism that keeps the whole thing running as one carefully balanced and integrated machine.

## The Heart

The heart of the Z-486/33ET is constructed of crystal and metal. This oscillator runs the CPU at 33 million beats per second (MHz), which nudges instructions and code through the 80486 microprocessor about every 30 nanoseconds. The 80486 probably thinks this is a leisurely pace (some designs have boosted this rate to 50 MHz in hand-picked specimens). The 80486 controls the clock to the rest of the system, and can alter the speed by inserting wait states if it must. This is termed "slow mode", and is required by some software that depends on certain timing loops (copy-protected software for the most part).

## The Brains and Short Term Memory

The silicon brain of the Z-486/33ET is mounted in a pin-grid array package, which in turn is located in a socket on the system board. It keeps an eye on everything that happens in the body of the beast. To do this

it needs a little help, from an external coprocessor (if present) to quickly execute a particularly troublesome math problem, or from the ROMs and memory, where it gets the instructions it needs to do almost everything.

The system board contains a socket for an optional coprocessor. Though the CPU has an internal coprocessor, an external coprocessor can be used which can greatly speed up mathematical computations. The Weitek 4167 numeric coprocessor is substantially faster than the 80387-equivalent floating point unit inside the 80486.

The BIOS (basic input/output system) ROMs (read-only memory) contain the code the CPU needs to boot a disk and initialize all of its internal circuits, much like we drink a coffee and look for our car keys every morning we go to work. It also contains the instructions needed to handle the usual interruptions that occur during every normal day. This computer contains 128K of ROM memory, contained in two 64K devices. Like all Zenith Data Systems computers, this code is transferred to a write-protectable portion of system memory where it can be executed at a faster speed (the slushware concept).

The system memory (short term memory compared to long term hard disk drive memory) in this computer is composed of DRAM (Dynamic Random-Access Memory) circuitry. The memory controller circuits can control 2M SIMMs (single in-line memory modules) and/or 8M SIMMs, which are installed in sockets on the system board. The maximum system memory that can be placed on the system board is 8M (if 2M SIMMs are used), or 64M (if 8M SIMMs are used), or various combination in between, depending on what is the most cost-effective. If more memory is required, it can be added by

installing memory expansion cards in the EISA-expansion slots.

Since the brain operates at such a blistering speed, it sometimes runs faster than the memory circuitry would like. The system memory would slow it down if not for a little piece of hardware that helps the brain and its slow system memory get along. This hardware is special fast memory circuitry called cache memory. The 80486 has 8K of cache memory inside it. The system can also be augmented by the addition of another (optional) 128K cache subsystem in a socket on the main board.

Even with all that system memory and ROM space at its disposal, sometimes the brain requires more, especially if it has to remember what boards are in its expansion bus, what each board does, how each board does it, and not to mention the various and sundry password schemes that are currently in vogue. To do all this (and to keep it fresh while the power is turned off), SRAM (static RAM) and NVRAM (Non-Volatile RAM) are used. The computer uses an 8K block of static RAM that is dedicated to storing system configuration parameters as determined by the setup and EISA configuration utility. These parameters include the particulars about the expansion cards installed in the backplane, as well as other system details. The NVRAM is used primarily for the password functions.

The Z-486/33ET uses three levels of password. These are:

- A power-on password. This activates during a reboot or when the system power is turned on.
- A network password. This password can be set to lock the keyboard when the computer is booted with MS-DOS.
- A separate software utility password. This is an option piece of software that can set up a special keyboard password, which can be invoked to lock the keyboard during lunch or break times (for instance) in order to prevent unauthorized use.

Last in the memory round-up, but certainly not least, is the real-time clock/calendar. This piece of hardware stores time and date information in a small chunk of RAM. The real-time clock/calendar circuitry also contains 50 spare bytes of RAM that are used to store configuration information. When the computer is turned off, the information is kept current by a battery backup circuit, which also keeps the clock and the calendar running.

### Long Term Memory

Long term memory in the Z-486/33ET is composed of circuits that convert code and data into magnetic bits, which are sent to electro-mechanical components that record the bits on rotating platters of plastic or metal; the disk drives.

The floppy drive controller circuitry handles the interface between the system and the floppy disk drives. This computer comes standard with a 1.4M, 3.5-inch floppy drive, but can control an additional 1.2M, 5.25-inch floppy disk drive or another 1.4M, 3.5-inch floppy disk drive as well. Note that it also has the capability of controlling the lesser versions of both floppy disk drives (360K and 720K, respectively), but that option is so rarely implemented that no formal upgrade exists.

This computer comes with a 338M SCSI hard disk drive. The hard disk drive controller card provides the interface between the system board and the drive. The card also provides a connection to external SCSI devices, via a cable that can be attached to a connector on the back of the computer. The SCSI controller circuitry on the hard disk controller card can control a maximum of seven additional SCSI devices. For information about SCSI devices and their wide range of abilities, refer to my earlier articles "Small Computer System Interface (SCSI)", and "Son of Small Computer Systems Interface," in previous issues of REMark magazine.

Along with the increased options that a SCSI interface provides, there is an upgrade available that allows this computer to interface with a high performance disk array. A disk array is a set of redundant disks that operate together to provide a fool-proof data retention scheme. Any of a number of schemes are available, from simple mirroring of data to much more complex methods. Disk arrays — sometimes called RAID's (Redundant Arrays of Inexpensive Disks) — will be examined closer in a future article — keep an eye out for it.

### Eyes and Ears

Just as a human being needs eyes and ears (and a nose) for sensory input to tell him what is happening in the world around him, the computer needs somewhat of an equivalent. The Z-486/33ET comes with a 101-key keyboard and a Microsoft mouse that can be used to tell the computer what to do. Both items connect to the back of the computer through mini-DIN — sometimes called PS/2 — style connectors. To make life easier, Zenith Data Systems pre-loads the mouse driver software on the hard disk drive, so all you have to do is connect the mouse and fire up an application that uses it.

### The Mouth

Just as the mouth accepts food (and in some cases — especially if you eat government-hosted dinners in Japan — spews it out), the I/O ports accept information and the video circuits display information. This computer has several "mouths". First, let's examine the I/O ports.

The system board has two 9-pin RS-232C compatible serial ports accessible from the back of the chassis, each configurable as COM1, COM2, or disabled. There are also two 25-pin, bi-directional (Centronics style) parallel ports, also in the back of the chassis, that can be configured as LPT1, LPT2, or disabled.

Video circuits tell the operator what he's doing, via keyboard and program feedback. The VGA video board supplied with the Z-486/33ET is a fast ISA (Industry Standard Architecture) circuit board with a 16-bit bus interface. It plugs into one of the expansion connectors and outputs VGA-compatible color video. It can emulate the other popular video formats (CGA, EGA, and MDA) if required. Video memory is located on the video board, and is composed of 256K of DRAM. This memory is upgradeable to 512K. The video board also has a features connector which provides a connection to a compatible high-resolution video board that can co-exist with the VGA board.

The astute reader may notice I've left out several major organs. The reason I did this was somewhat arbitrary (mostly for the purpose of good taste and responsible writing). Leaving the organic analogies behind, let's take a look at some other features of this computer.

### EISA Expansion Capabilities and Power

There are tern expansion connectors on the system board, each conforming to the EISA specification, which means that ISA and/or EISA boards can be installed in this computer with equal ease. (Note that these connectors are limited to 2 amperes at each +5 volt power pin.)

The 395 watt power supply is the standard auto-switching, auto-sensing design used in most previous Zenith Data Systems computers. It has an internal cooling fan and provides the following power taps:

+12 VDC at up to 13 A  
-12 VDC at up to 1.0 A  
+5 VDC at up to 45 A  
-5 VDC at up to 0.5 A

### Software and Miscellaneous Stuff

Zenith Data Systems provides the latest version of MS-DOS (currently 5.00) with the computer, on a set of accompanying disks. If for some reason you don't want MS-DOS, the system is also OS/2 compatible. The system is also compatible with UNIX and XENIX. MS-DOS is not pre-installed on the hard disk drive for this reason.

The base computer (with one hard disk drive and one floppy disk drive) weighs about 85 lbs. There is space for up to six half-height storage devices to be installed inside the cabinet. The system is UL listed

Continued on Page 35



---

# The ZLS Workstation Series



**Nathan E. Baker**  
Zenith Data Systems  
Technical Writer

## LANs Best Friend

Local Area Networks (LANs) are becoming the corporate work environment, and most analysts predict that all serious businesses that use computers will be LAN based in the years to come. LANs provide a very efficient way to maximize an expensive computer resource, such as a high-output laser printer, and provide an excellent means to reach all employees at the touch of a button (via electronic mail or other notification programs). A disappointing side-effect to all this is that they also offer a nurturing environment to computer viruses and the big brother temptation (work can be monitored in a LAN-based environment — keystroke-by-keystroke if desired). But down-side notwithstanding, LANs are here to stay.

For those who are uninitiated with the LAN concept and terminology and theory, here's a crash course.

The simplest LAN is a main computer that holds all the resources (such as software and disk drives) which is connected to another computer (in another part of the building, for instance) via a cable. The main computer is called the server. The computer at the other end of the cable is called a workstation. The actual data entry takes place at the workstation, while the server doles out parts of its resources to the workstation. The resources can include access to hard disk drive space in the server, access to external peripherals (like the aforementioned printer), and many other things.

Many workstations can be attached to a single server. Workstations can also be attached to multiple servers as well. The physical layout in which the workstations

and servers are connected is called the LAN topology. Basic topologies include star, linear bus, and ring, among others. Each topology has its own advantages and disadvantages.

The topology in a LAN environment is dependent on the network interface card (NIC). The NIC provides the physical connection and the logic required to interact with the rest of the network. Each workstation and each server must have their own NIC.

LAN software uses the NIC to transfer and receive data packets, which comprise the actual working of the LAN.

### Workstations

Since network resources are shared between workstations, the number of workstations and the type of workstations being used can have dramatic impact on the performance of the network.

The simplest workstation has nothing but the NIC, a display, a keyboard, and the logic required to use the NIC. Such a workstation depends heavily on the memory and storage capacities of the server. It needs to boot from the server, store information on the server, and execute programs from the server. As the number of workstations increase, or the amount of work increases, more and more server time is devoted to each workstation. Overall network performance in such an environment can be abysmally slow. To ease the load on the server, many functions can be handed-off to the workstations, if the workstation hardware can accommodate the function.

A PC (with an NIC) can be a suitable

workstation. If it has 640K of RAM, a floppy disk drive, and a hard disk drive, its requirements from the server can be minimized. The only time such a workstation needs server attention is when it uses a shared resource (such as the high-output printer), when it is storing data on the server's hard disk drive, or when it is using programs in the server RAM. Certain data is always sent to the server (like login information), but most of the data for a lot of work environments can be kept local, in the hard drive in the workstation. This eases the requirements of the server. However, PCs are less than state-of-the-art. They're slow and have serious memory limitations.

An AT makes for a better workstation, but assigning the expense of buying and setting up AT computers for use only as a workstation is enough to hamper most company budgets. But since speed and performance need to be optimized, many companies are opting for this solution, especially when server functions and shared resources increase in number.

There is obviously a niche for a high speed workstation that contains the necessary features (without extraneous features) that can be had at a decent price. Enter the Zenith Data Systems LAN machines.

### Compact and Efficient: The ZLS Workstation Series

Zenith Data Systems has targeted the workstation niche with two entries: the ZLS Workstation, and the ZLS/20 Workstation. Their standard features are compared and explained in the following paragraphs.

*Physical features* — Both computers come in a compact, small-footprint design

that minimizes the space the machine takes up on your desk. The power button is located on the front. There is a metal bracket on the back of the computer that can be used to physically restrain the machine by attaching it to a cable-lock device (so no one walks off with your computer).

**Keyboard**—Both machines come with a 101-key keyboard with programmable function keys and automatic key repetition.

**Processors**—Both computers contain an Intel 80386SX microprocessor for a CPU. The CPU clock speed is 16 MHz in the ZLS Workstation, and 20 MHz in the ZLS/20 Workstation. Both computers provide an empty socket for installation of an Intel 80387SX numeric coprocessor.

**Monitor ROM**—Both computers contain a monitor ROM that holds the basic I/O software (BIOS). This code tells the computer how to initialize internal circuitry and how to boot an operating system. As in other Zenith Data Systems computers, this code is transferred to a write-protected area of system RAM during the power-up sequence—where it becomes slushware—and executes at a much faster speed.

**RAM**—The ZLS Workstation comes with 1M RAM standard. The system board can accommodate additional 1M SIMMs, up to a maximum system memory of 8M. The ZLS/20 Workstation has 2M standard. Its system board can accommodate additional 1M and/or 4M SIMMs, up to a system maximum of 10M. Expanded memory (LIM version 3.2) and extended memory are supported by both workstations.

**Password**—Both computers provide an electronically erasable read-only memory

chip (EEPROM) for storing password information. The computer can be set up to allow operation only if the correct password is entered (if the wrong password is entered the keyboard goes dead and the drives and memory cannot be accessed).

**Note:** The ZLS/20 Workstation uses a two-level password scheme. The first level password is set up by the system supervisor, who may then loan the computer out to a second party. The second party can set a user password. The user password, if forgotten or not cleared when the computer is returned, can be flushed out by the system supervisor by invoking his supervisor password. The supervisor can also save a default configuration, so he can reset the computer to a known-good configuration at the same time (in case the second party decided to alter the computer's hardware configuration in the setup menu).

**Disk Drives**—Both computers can be purchased with a floppy disk drive or a hard disk drive, or a combination of both. In high security environments these disk drives could be removed and the computer could boot from the network.

**Video**—Both computers come with built-in color video circuitry that is 100% VGA register-level and BIOS-level compatible. The video circuits support EGA, CGA, HGC, and MDA video formats as well, with no jumpering or DIP switch alterations required.

The ZLS/20 Workstation comes with 512K of video RAM standard. The ZLS Workstation, being somewhat cost-reduced, comes with 256K of video RAM standard, but it can be expanded to 512K by the user.

The computer isn't sold with a VGA monitor, but most dealers will bundle the machine with a compatible monitor when you go "workstation hunting", so try to get a good deal and go for the color monitor. The computer can drive a monochrome monitor with no problems, but monochrome VGA just isn't quite the same (especially since word processing packages and electronic mail are beginning to make extensive use of color—bells and whistles galore!).

**Mouse**—Both computers have a mouse port located in the back of the cabinet, for use with a PS2-style mouse (or other pointing device that has a PS2-type connector).

**Serial and Parallel**—Both computers contain the standard serial and parallel I/O ports. The serial port can be configured as COM1 or COM2, or disabled. The parallel port can be configured as LPT1, LPT2, LPT3, or disabled. The parallel port can also work in bidirectional mode for PS2 compatibility.

**Backplane Board**—Both computers contain a backplane board with two standard AT-compatible expansion connectors. In a LAN environment one of these con-

nectors would be occupied by a NIC, but the other connector is available for expanding the basic machine with a variety of AT-compatible expansion cards (contact your dealer to get a list of available AT-expansion cards).

**Power Supply**—Both computers come with an auto-switching power supply capable of producing a maximum of 65 watts.

**Battery and Real Time Clock**—The backup battery provides power to the real time clock, which constantly keeps track of time and date. The battery is mounted on the system board, and can be easily replaced when it runs down (which isn't very often).

**Options**—Options for both computers include the following:

- AT-compatible circuit cards
- Memory
- Numeric coprocessor
- Hard disk drive
- Video RAM (for ZLS Workstations only).
- Cover lock and key

Note that these options can be installed by the user.

Due to the fact that different network topologies require different cards, neither computer is supplied with a NIC installed. It's just too difficult to second-guess what the customer's requirements are in view of all the options out there. Documentation supplied with the NIC explains how to install it in a generic fashion, but that's basically unneeded since the manuals supplied with the Workstation explain how to install all the options in a systematic, step-by-step manner.

### Making Your Choice

Choosing the correct workstation is relatively simple.

The ZLS Workstation is a good choice if the work is not particularly complex and speed is not a factor. It's a machine that's suitable for simple text entry and document creation functions, and straight-forward data entry (although this isn't to say it can't do more complex functions). This computer is also the better choice if your budget is helping you make the decision. (Remember that things like system memory and video memory can be easily upgraded if you decide you need more in the future.)

The ZLS/20 Workstation is more suitable for environments where the user will be manipulating and laying out text (such as in desktop publishing), or creating or using memory intense programs.

It's interesting to note that both models make decent stand-alone computers as well, if you purchase one with a disk drive installed. Whichever one you choose, you'll have the backing of the Zenith Data Systems technical response department to help you with your technical needs, as well as the reliability inherent in all Zenith Data Systems products. ✻





---

---

# Inside DOS 5.0 ...

# New Commands

**Mark Haverstock**  
**6835 Colleen Drive**  
**Youngstown, OH 44512**

In the first article of this series, I discussed the pros and cons of changing your system to DOS 5.0. Without a doubt, Microsoft's new DOS is smarter, leaner and more powerful than earlier versions. To make this possible, they enhanced some older commands and added several new ones. This article will highlight the new commands and give you some insight on how they can be used.

The fourteen new DOS commands covered in this article are organized into four groups: Working With DOS, File Recovery, Working With Text and Memory Configuration. You'll also find a brief description on how to run the QBasic interpreter. Included is the syntax for each command and examples where applicable.

## **Working With DOS**

### **DOSKEY**

One of DOS 5's most exciting new features is DOSKEY, a terminate-stay-resident program (TSR) that stores a history of the commands you typed in memory. The number of commands it can store depends on the buffer size (default is 512 bytes). This is a big improvement over the F3 function key shortcut, because you can recall previous commands by using the PgUp and PgDn keys.

DOSKEY also can help you create and store macros in a way that's similar to creating batch files. In the macros, you can include any valid DOS command, including batch files. By typing in the name of a macro you've created, you can perform several commands with a few keystrokes.

Since creating macros and editing is a lengthy subject, I've just included a basic description of DOSKEY and its syntax.

#### **Syntax**

```
DOSKEY [/REINSTALL] [BUFSIZE=size] [/MACROS] [/HISTORY]
        [/INSERT|/OVERSTRIKE] [macro={text}]
/REINSTALL    Install a new copy of DOSKEY
/BUFFER=size  Set aside memory for DOSKEY to store commands
              and macros (default=512 bytes)
/MACROS       Display all currently defined macros
/HISTORY      Display command history
/INSERT       Defaults to insert mode for editing
```

```
/OVERSTRIKE  Defaults to typeover mode when editing com-
              mands
macro        Macro name
text        Macro definition
```

Example: To create a macro for moving files, use this form:  
DOSKEY MOVE=COPY \$1 \$2 \$T DEL \$1

## **DOSSHELL**

Like it or not, we're still stuck with the DOS prompt. Eventually, we find ourselves going back to the C:\> to do some tasks. One concession to those of us who want to avoid the DOS prompt is the new DOS 5.0 Shell. It brings a Windows-type interface to every PC user, and is intended to make operation easier. You can use the mouse or keyboard to perform nearly every DOS task.

The DOS 5.0 shell presents a graphic directory tree in one window and a list of files for the current directory in the other. This simplifies many common tasks, such as copying or moving files, moving around in directories and running programs.

There are some things you can do from the shell that you can't do from the DOS prompt. For example, you can rename a subdirectory. You can task swap between applications, although it suspends the swapped-out program. A file association feature lets you link programs with their data files. When you click on a data file, the Shell will load both the application and the file.

#### **Syntax**

```
Graphics mode: DOSSHELL [/G[:res[n]]]
Text mode: DOSSHELL [/T[:res[n]]]
res           Specifies screen resolution. Values are:
              L    low
              M    medium
              H    high
n             Specifies a screen resolution number. The value de-
              pends on the type of video adapter used.
/G            Starts DOS shell in graphics mode
/T            Starts DOS shell in text mode
/B            Starts shell in black and white color scheme
```

Example: To start DOSHELL in the graphics mode, you'd type the following:

```
DOSSHLL /G
```

## EXPAND

The files on your DOS disks are shipped in compressed form. This enables more data to be stored on fewer disks. Normally, the installation program expands them for you. But what if you accidentally delete a file? You can use EXPAND to transfer the file from the original DOS disk without having to reinstall.

EXPAND is similar to the COPY command. However, EXPAND is a one-way copy.

### Syntax

```
EXPAND [d:] [path] [filename1] [...] [d:] [path] [filename2]
filename1 Name of file or files to expand
filename2 Name of expanded file
```

Example: To expand FORMAT.COM from the DOS 5.0 floppy disk to C:\DOS\, you'd type the following:

```
EXPAND A:FORMAT.CO_ C:\DOS\FORMAT.COM
```

## HELP

With all the new and different commands, you can easily forget which switch works with which command, or the proper syntax of a command. Finally, DOS has an online help feature for each DOS command, both internal and external. While the help isn't as extensive as that in the reference manual, you'll probably find enough information to navigate through most commands.

### Syntax

```
HELP [command] or [command]/?
command The name of the command you want help with.
```

Example: HELP XCOPY or XCOPY/?

To get a list of all the DOS 5.0 commands, type HELP at the DOS prompt.

## SETVER

Some programs require specific DOS versions to run properly. SETVER sets the DOS version to be emulated with a particular program. DOS 5.0 contains a version table, which lists the names of programs and the number of the MS-DOS version with which they are designed to run. You can modify the version table to add additional programs that are not listed.

To enable SETVER, you need to install SETVER.EXE as a device in your CONFIG.SYS file—it can be loaded in conventional or high memory. If you add or delete entries from the version table, you'll need to restart the system for changes to take effect.

### Syntax

To install SETVER as a device, when SETVER is located in C:\DOS, add the following to your CONFIG.SYS file:

```
DEVICE=C:\DOS\SETVER.EXE
```

To view version table at the command line:

```
SETVER
```

To add a program to the version table:

```
SETVER [d:path] [filename.ext] [n.nn]
```

Example: SETVER C:\WORD\WP.EXE 3.3

To delete a program from the version table:

```
SETVER filename.ext /DELETE /QUIET
```

Example: SETVER WP.EXE /DELETE /QUIET

```
d:path Drive and path of program
filename Filename of application program to add/delete from
version table
n.nn DOS version
/DELETE Remove entry from version table
/QUIET Suppresses messages when /DELETE is used
```

## File Recovery

### MIRROR

Regular use of the MIRROR command can help you recover from possible disk disasters. It performs three basic functions. One, it creates a backup copy of a hard disk's partition tables. Two, it creates a mirror file that is used by UNFORMAT to recover the data on a formatted disk. Three, it performs delete-tracking, which helps the UNDELETE function help recover difficult files.

### Syntax

To create a backup copy of the current disk's partition tables:

```
MIRROR /PARTN
```

To create a mirror file for use by the unformat command on the current drive:

```
MIRROR /1
```

To make mirror memory resident and have it perform delete-tracking for the UNDELETE command:

```
MIRROR /Tdrive [-entries]
```

To remove MIRROR from memory:

```
MIRROR /U
```

```
/PARTN Create backup copy of partition tables
```

```
/Tdrive Perform delete tracking for the specified drive
```

```
entries Number of entries in the delete-tracking file for the
corresponding drive (range=1 to 999)
```

```
/1 Retains only the latest information about the disk
```

### UNDELETE

The UNDELETE command lets you restore deleted files by using either the delete tracking file or the DOS directory. Note that the sooner UNDELETE is run after a file is deleted, the better its chances for recovery.

When you delete a file, the first character of the file name is removed. If you undelete using the /DOS switch, you'll be prompted to type in the missing first character. If you use the /ALL switch, each deleted file in the current directory will be restored. DOS will supply the missing first letters with numbers or symbols, such as #, % or &. You can later change these using the RENAME command.

### Syntax

```
UNDELETE [[d:] [path] filename] [/LIST | /ALL] [/DOS | /DT]
```

```
filename Name of the file or files to undelete
```

```
/LIST List the files that may be recovered
```

```
/ALL Recover files without prompting for the first letter
of each filename.
```

```
/DOS Ignore the delete-tracking file
```

```
/DT Use the delete-tracking file
```

Example: To undelete all files in the C:\FILES directory without prompting:

```
UNDELETE C:\FILES\*. * /ALL
```

### UNFORMAT

Before DOS 5.0, a reformatted disk was an erased disk. Now you can reconstruct a disk that was reformatted using the UNFORMAT command. UNFORMAT attempts to recover the disk, using the files created by MIRROR. If no files created by MIRROR exist, UNFORMAT still attempts to reconstruct the disk—although it is slow and may not be as reliable as unformatting a disk that has up-to-date MIRROR files.

UNFORMAT was originally written by Central Point Software for their utility package, PC Tools, and licensed by Microsoft.

### Syntax

To restore hard disk partition tables:

```
UNFORMAT /PARTN [L]
```

To unformat a hard disk or floppy:

```
UNFORMAT d: [/U] [/L] [/TEST] [/P]
```

To verify that the disk contains a mirror file:



UNFORMAT d: /J

- /PARTN Restore copy of disk partition tables
- /L List all file and directory names found, or display current partition table when used with /PARTN
- /J Verify MIRROR files with information on disk
- /P Send output messages to printer on LPT1
- /TEST Do not write changes to disk
- /U Unformat without using MIRROR file

### Working With Text Edit

Finally, DOS offers an easy-to-use, full-screen text editor. EDIT can create text files just as efficiently as your word processor, without the hassle of having to convert the word processing file into ASCII format.

For compatibility reasons, you'll still find EDLIN included with DOS 5.0. But you'll probably choose EDIT for its flexibility and features.

Edit will not run unless QBASIC.EXE is in the current directory, the current path, or the directory in which EDIT.COM is stored. DOSSHELL also can launch EDIT if you select it from the MAIN group using the arrow keys or mouse.

### Syntax

- ```
EDIT [[d:] [path] filename] [/B] [/G] [/H] [/NOHI] [/?]
filename Name of the file to edit
/B Start Editor in black and white
/G Quickly writes to a CGA monitor (may cause "snow" on some monitors)
/H Displays EDIT in the maximum lines that your screen supports (43 for EGA, 50 for VGA)
/NOHI Uses reverse video instead of high-intensity characters (for LCD screens)
/? On-screen help
```

### Memory Configuration EMM386

EMM386 is both a device driver and a DOS command. The device driver enables your 80386SX, 80386 or 80486 computer to convert extended memory into EMS 4.0 expanded memory, as well as control it. EMM386 also can convert some extended memory to reserved memory.

Used as a command, EMM386 turns the memory manager on or off. However, some programs will not run properly with EMM386 active. If you find such a program, be sure to disable EMM386 before running using the command

EMM386 OFF

or you can place it in the AUTO mode with EMM386 AUTO

and enable it again once you've exited the program. However, you can't switch to AUTO or OFF mode if you're currently using expanded memory for an application or if any TSRs or device drivers are loaded in upper memory.

### Syntax

- ```
EMM386 [ON|OFF|AUTO] [W=ON|OFF]
ON Enables EMM386 driver
OFF Disables EMM386 driver
AUTO Enables EMM386 driver only
```

- W=ON Enables Weitek coprocessor support
- W=OFF Disables Weitek coprocessor support

### LOADFIX

When DOS 5.0 is loaded into high-memory on a 286-, 386- or 486-based computer, programs may be loaded and run in the lower 64K of RAM. This may cause problems with some programs and produce a "packed file corrupted" message. LOADFIX avoids this problem by loading the file above the first 64K of RAM memory.

### Syntax

- ```
LOADFIX [d:] [path] filename [parameters]
filename Name of program to run
parameter Parameters passed to the program being loaded
Example: To execute the program SAMPLE.EXE that gave the "packed file corrupted" message, type:
LOADFIX SAMPLE.EXE
```

### LOADHIGH (LH)

On 80386SX, 80386 or 80486 computers, you can assign memory to empty spaces between the top of conventional memory (640K) and where extended memory begins (1MB). LOADHIGH lets DOS relocate programs to the upper-memory area of RAM. Loading a program into the upper memory area frees conventional memory normally used by these programs.

To use LOADHIGH, you first need to install HIMEM.SYS and EMM386 as device drivers in your CONFIG.SYS file. You also need to include DOS=UMB or DOS=HIGH,UMB to load DOS

Table 1  
Sample Display of Memory Allocation Using MEM/C

#### Conventional Memory:

| Name         | Size in Decimal | Size in Hex |
|--------------|-----------------|-------------|
| MSDOS        | 12208 ( 11.9K)  | 2FB0        |
| QEMM386      | 2416 ( 2.4K)    | 970         |
| SETVER       | 400 ( 0.4K)     | 190         |
| HHSCAND      | 8288 ( 8.1K)    | 2060        |
| COMMAND      | 2624 ( 2.6K)    | A40         |
| ENVELOPE     | 2336 ( 2.3K)    | 920         |
| FREE         | 64 ( 0.1K)      | 40          |
| FREE         | 192 ( 0.2K)     | C0          |
| FREE         | 626592 (611.9K) | 98FA0       |
| Total FREE : | 626848 (612.2K) |             |

#### Upper Memory:

| Name         | Size in Decimal | Size in Hex |
|--------------|-----------------|-------------|
| SYSTEM       | 241760 (236.1K) | 3B060       |
| VDEFEND      | 19040 ( 18.6K)  | 4A60        |
| SWAPSH       | 9264 ( 9.0K)    | 2430        |
| FASTBIOS     | 37456 ( 36.6K)  | 9250        |
| EANSI        | 1888 ( 1.8K)    | 760         |
| SMARTDRV     | 14576 ( 14.2k)  | 38F0        |
| FREE         | 23312 ( 22.8K)  | 5B10        |
| FREE         | 25280 ( 24.7K)  | 62C0        |
| FREE         | 8128 ( 7.9K)    | 1FC0        |
| FREE         | 4064 ( 4.0K)    | fe0         |
| Total FREE : | 60784 ( 59.4K)  |             |

Total bytes available to programs (Conventional+Upper) : 687632 (671.5K)  
Largest executable program size : 626368 (611.7K)  
Largest available upper memory block : 25280 ( 24.7K)

3719168 bytes total EMS memory  
1998848 bytes free EMS memory

3407872 bytes total contiguous extended memory  
0 bytes available contiguous extended memory  
1998848 bytes available XMS memory  
MS-DOS resident in High Memory Area

into the high memory area and enable upper memory blocks.

Programs that you'll normally load high are device drivers and TSRs — for example, ANSI.SYS or FASTOPEN.EXE. You can include LOADHIGH in the AUTOEXEC.BAT file so you can load device drivers or TSR programs into upper memory each time you boot your system.

If there isn't enough upper memory remaining to load a TSR high, LOADHIGH will automatically put it in conventional memory. To check where the programs reside, use the MEM/C command.

### Syntax

LOADHIGH [d:] [path] filename [parameters]

filename The name of the program to run

parameters Parameters passed to the program being loaded

Example: To load your mouse driver from the C:\DOS into upper memory, use the following form:

```
LOADHIGH C:\DOS\MOUSE
```

### MEM

MEM displays the amount of used and free memory in your system. By using MEM, you can display statistics for conventional memory, as well as extended and expanded memory if they are available.

If you have an 80386SX, 80386 or 80486 based PC, you can use MEM/C or MEM/P as you begin to load your device drivers and TSRs into reserved memory. With the information you'll receive, you can determine the order that the device drivers and TSRs load into memory so you can make the best use of the available memory space (Table 1).

### Syntax

MEM [/CLASSIFY|/DEBUG|/PROGRAM]

/CLASSIFY Display a list of programs loaded in conventional and upper memory

/DEBUG Display a list of programs and internal device drivers loaded in memory

/PROGRAM Display a list of programs loaded in memory (These switches can be abbreviated /C, /D and /P)

Example: To see what's loaded into upper memory, type:

```
MEM/C
```

### QBASIC

The command QBASIC starts MS-DOS QBasic, a complete environment for programming in the BASIC language. A subset of Microsoft QuickBasic, QBasic replaces BASIC, BASICA and GWBASIC from earlier versions of DOS.

QBasic has extensive online help. If you want more information about using QBasic, press ENTER immediately after starting QBasic, or press F1 while running QBasic (Figure 1).

### Syntax

QBASIC [/B] [/EDITOR] [/G] [/H] [/MBF] [/NOHI] [/RUN] [d:] [path] [filename]

/B Start QBasic in black and white

/EDITOR Start the DOS Editor

/G Performs fast updates on CGA monitors. Don't use this switch if "snow" appears on the screen

/H Displays maximum number of lines

/MBF Converts the built-in functions CVD, CVS, MKS\$ and MKD\$ to CVDMBF, CVSMBF, MKSMBF\$ and MKDMBF\$

/NOHI Suppress display of high-intensity colors

/RUN [d:] Loads filename.ext into memory and starts the

program

filename

Example: To start QBasic and run a program called LABELS, type:

```
QBASIC /RUN LABELS.BAS
```

### Summary

DOS 5.0 represents a major improvement in operating systems for PC compatibles. It's compatible with programs that ran under earlier versions of DOS, but also includes many advanced features that take advantage of the power of today's PCs.

New features of DOS 5.0 often eliminate the need for add-on programs PC users have needed in the past to perform certain functions. For example, to UNDELETE or UNFORMAT formerly required programs such as PC Tools or Norton Utilities.

In Part 3, we'll discuss how to get the most out of your computer's memory in DOS 5.0.

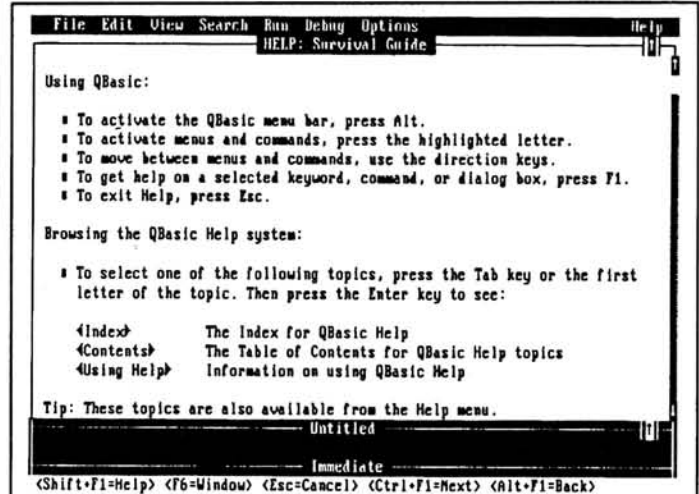
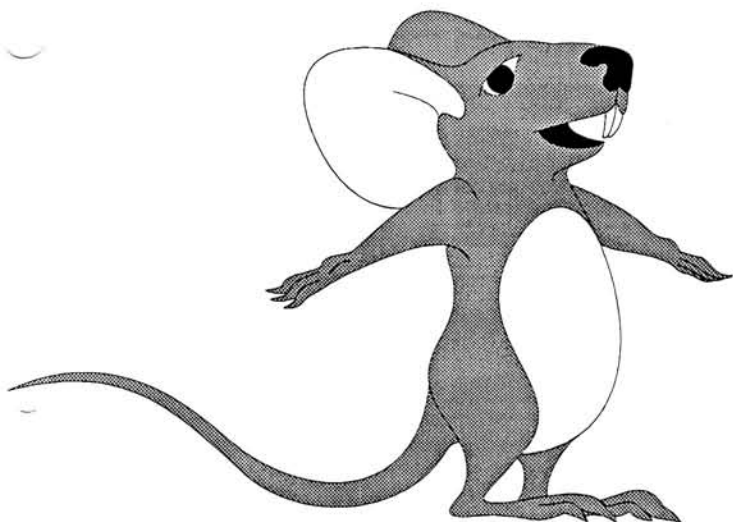


Figure 1  
QBASIC On-Line Help Screen







# Lying to Your Mouse

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

## How to Make a Mouse Think It's a Mouse Systems PC Mouse

Although more and more IBM-Compatible computer users are buying mice, not very much has been written on how to use them properly. For a long time people regarded mice as something having to do with Macintosh computers, but now all that is changing, so the purpose of this article is to help people who have a mouse to use it better, and to help those who haven't purchased one yet, or who have one still in its box, to start using one.

PC-Write 3.04 is an excellent word processor. When it comes to mice, however, PC-Write is sort of a hybrid. Although it does not have a built-in mouse driver, "mice that are distributed without menu software cannot be used with PC-Write" (from the September 1988 PC-Write manual).

Some software comes with its own drivers. This includes multi-tasking programs such as Quarterdeck's DESQview, and games such as Sierra's Leisure Suit Larry series or Railroad Tycoon, the railroad simulator from MicroProse. All of these programs look for the mouse driver that comes with the mouse. If they find it installed, the program can be used with the mouse. Other software is not built for a mouse at all. These programs can be used with a mouse, however, if someone designs a special mouse driver for them. In many cases, the mouse manufacturers design their own program-specific drivers. For instance, Logitech has designed a special menu which allows DOS Edlin to work with a mouse, and another special menu which allows even DOS itself to work with a mouse, without invoking DOSSHLL.

PC-Write is a little different. Although the program itself is not mouse-driven, and there is no setup for PC-Write which asks you about a mouse, PC-Write comes with

mouse software which has been specially written for it. In PC-Write 3.xx, this software includes menus for the three most popular mice, as well as compiled versions of these menus. If you own a Microsoft 6.0 (or later) mouse, you can call up PC-Write by writing MOUSE and then MICSMICE. MOUSE.COM is a file which comes with the Microsoft 6.0 (or later) mouse, and MICSMICE.COM is the compiled version of MICSMICE.DEF, both of which are included with PC-Write 3.xx. If you don't like something about the way the Microsoft mouse works with PC-Write, you can use the Microsoft Mouse Programmer's Reference Guide to edit the MICSMICE.MNU file, and then recompile it into a new MICSMICE.COM. The instructions are slightly different for a Microsoft 5.0 mouse, for which you also need a file called MICSMICE.MNU, which is also provided with PC-Write 3.xx.

If you own a Logitech mouse, as I do, you run MOUSE (the Logitech mouse driver), followed by MENU (the Logitech menu user) LOGIMICE. This loads the LOGIMICE.MNU, which is the compiled version of the LOGIMICE.DEF, provided with PC-Write 3.xx. In other words, type:

```
MOUSE
MENU LOGIMICE
ED (starts PC-Write)
```

If PC-Write does not respond, try again, but this time, before writing ED, write MOUSE ?. The reported baud rate should be 2400. If some previous program has set it at 1200, PC-Write will freeze. Writing MOUSE 2400 should solve the problem. You may also have MOUSE set for the wrong COM port.

If you don't like the way the mouse works, you can edit the LOGIMICE.DEF file in any editor or word processor, and then

recompile it using NEWMENU.EXE, which comes with the Logitech mouse.

To sum up the files mentioned so far:

Included with PC-Write 3.xx:

MICSMICE.DEF — Menu for a Microsoft mouse  
MICSMICE.MNU — Needed for a Microsoft 5.0 mouse

MICSMICE.COM — Compiled version of MICSMICE.DEF

LOGIMICE.DEF — Menu for a Logitech mouse  
LOGIMICE.MNU — Compiled version of LOGIMICE.DEF

Included with the Microsoft mouse:

MOUSE.COM — Driver for the Microsoft mouse  
Included with the Logitech mouse:

MOUSE.COM — Driver for the Logitech mouse  
MENU.COM — Loads Logitech menus  
NEWMENU.EXE — Compiles .DEF files into .MNU files

With PC-Write, the Logitech and Microsoft mice are similar, the only differences being in the fact that the Logitech mouse has three buttons, and the Microsoft mice have two. Using the mouse buttons on either mouse, you can mark text, call the Help screen, use the left button as ESC and the right button as ENTER. You can use the mouse itself to highlight various menu options in any of the four menus shown at the top of the screen, and then choose one with the right mouse button, or simply to move around the text quickly.

Figure 1 shows two of the four PC-Write menus. They can be accessed either with cursor + ENTER, the F keys, or with the mouse + right button.

The first menu is the one showing at all times while editing, and the second is the one accessible by holding the Alt key down. There are also Shift and Ctrl menus.

The problem here is basically: Since PC-Write gives me all these nice menus anyway, what do I need the Logitech mouse

```

Esc:Menu Push Wrap+Se+ R:F 93# 1/99, 1 Read "articles\mouse.lie"
F1:System/help F3.Copy/mark F5.Un-mark F7.Paragraph F9:Find-text
F2:Window/ruler F4.Delete/mark F6.Move/mark F8.Lower/upper F10.Replace
MENU: help, exit, save/unsave, name current file, load file, print, directory

Esc:cancel. F1-F1:help. Alt/Shf/Ctl/Arrow:select. Fn-key/Letter/Enter:action.
aF1:Name/file aF3:Key-record aF5:Conversion aF7:Pagebreaks aF9:To-location
aF2:Spelling aF4:Misc-ops aF6:Formatting aF8.Upper-case aF10:Replace-all
MENU: filename prefix, drive, path; file rename, write, erase; directory

```

Figure 1

for?

Let's pause to examine the situation for people owning a Mouse Systems PC Mouse. To use it with PC-Write 3.xx, all I have to do is run MOUSESYS.COM, which comes with the PC Mouse, and then MSYSMICE.COM, the compiled version of MSYSMICE.MSC, both of which come with PC-Write 3.xx. In other words,

```

MOUSESYS
MSYSMICE
ED

```

PC-Write will now give you the normal menus, accessible with the cursor + ENTER, or the F keys, and will also let you move the cursor around the text. In addition, MSYSMICE offers you six(!) pull-down menus. Unfortunately, although DESQview 2.6 did a wonderful job of capturing the regular menus into this article, I was unable

to get it to capture the pull-down menus. I captured the regular menus by running a second copy of PC-Write within DESQview, and then marking the top of the screen and copying into the first copy of PC-Write. When I tried to do the same thing with the pull-down menus showing, I was unable to call up the DESQview menu! At any rate, the pull-down menus look something like this:

```

MARK
(To Page Menu)

Clear Marks

Mark
Delete
Move
Copy

Mark All
Copy to File
Un-Delete

Exit This Menu

```

The other five menus are the Page Menu, the Edit Menu, the File Menu, the Search Menu and the Jump Menu. The Edit Menu offers the following commands: Push/Overwrite, Reformat, Wrap+/, Para+, Upper/Lower, UPPER ONLY, Center Line, Del Word <, Del Word >, Del Line End, Del All Line. Some of these commands may be a little confusing for those of you unlucky enough not to be using PC-Write, but anyone can easily see how useful it is to have so many commands at your fingertips.

If you don't like the way MSYSMICE.COM operates, you can edit MSYSMICE.MSC and then recompile it with MSC.EXE.

I know that many people get used to a certain piece of software, and then don't want to upgrade. In certain cases the original is better than the upgrade, and in other cases the upgrade is so much more complicated that people simply want to go on using what they are used to. In case you are one of these people, MSYSMICE.COM was not offered with the PC-Write 2.xx diskettes. You have to take the PCWMOUSE.DEF file which came with PC-Write and compile it, using MSC.EXE.

Now, here's the problem faced by many mouse owners. We have Logitech mice or trackballs, but we don't like the way they work with PC-Write. The solution: Lie to your mouse! Tell the Logitech mouse that it is really a Mouse Systems PC Mouse.

Luckily, I had an old Mouse Systems Optical mouse lying around, complete with software, so I was able to do the following.

MOUSE PC — Tells the Logitech mouse to emulate a Mouse Systems mouse

MOUSESYS — The driver for a Mouse Systems mouse

MSYSMICE — The driver for PC-Write ED — PC-Write

For those of you who need a capsule summary, here is a list of the programs mentioned since the last list. Included with PC-Write 3.xx:

MSYSMICE.MSC — Menu for a Mouse Systems mouse

MSYSMICE.COM — Compiled version of MSYSMICE.MSC

Included with the Mouse Systems mouse:

MOUSESYS.COM — Driver for the Mouse Systems mouse

MSC.EXE — Compiles .MSC files into .COM files

**Note:** Do not confuse MSC.EXE with MSC.BAT, which now comes with the Logitech mouse, and is simply a list of commands telling the Logitech mouse to emulate a Mouse Systems mouse. This is the same as writing MOUSE PC or MOUSE 1200 5b.

I recently added a new mouse to my menagerie, the Genius Mouse. The Genius is Mouse Systems compatible by default (it can also be Microsoft compatible), so I thought I would try lying to it as well. I tried

```

MOUSESYS
MSYSMICE
ED

```

and I now use my new Genius mouse with PC-Write, pull-down menus and all!

### Products Discussed

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

Leisure Suit Larry  
Sierra On-Line Inc.  
Coarsegold, CA 93614  
(209) 683-6858

Railroad Tycoon  
MicroProse  
180 Lakefront Drive  
Hunt Valley, MD 21030  
(800) 876-1151

PC-Write  
Quicksoft  
219 First Avenue N #224  
Seattle, WA 98109  
(206) 282-0452

Logitech  
6505 Kaiser Drive  
Fremont, CA 94555  
(415) 795-8500



**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; ZR AM-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

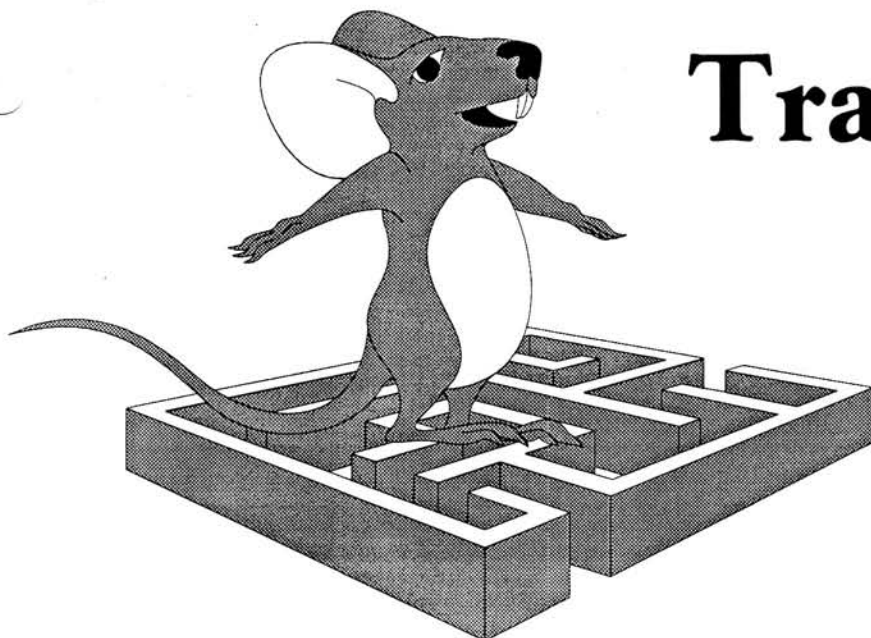
#### Call Or Write For Additional Information

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 TouchTone Selectable





# Trapping that Mouse

David W. Lind  
RR 1, Box 3114  
Bar Harbor, Maine 04609  
Copyright © 1991

## Introduction

The mouse is perhaps the most widely used pointing device on International Business Machines Personal Computers (IBM-PCs) or compatibles. Many of us would have great difficulty operating our computers were it not for the mouse. Recently, I installed version 5.0 of the Microsoft Disk Operating System (MS-DOS) on my Heath 386 computer. The driver for the mouse was not installed when I booted the new operating system for the first time. I spent many agonizing moments fumbling with the keyboard to accomplish the most simple tasks. Finally, I managed to edit the AUTOEXEC.BAT file and reboot the system making the mouse available. At that point, I was able to return to my usual efficient mode of operation with a considerable sense of well-being.

Many of us depend upon the mouse for the efficient operation of applications programs, but never think of incorporating the mouse into programs we write. Perhaps such a task seems formidable, but it is not. This article demonstrates how to program mouse operations. In a departure from the rule I usually follow in articles I write, this article explicitly shows how to program mouse features in high level languages as well as assembly languages. Readers who are not interested in assembly language coding can ignore those sections and find the balance of the discussion useful.

The discussion begins with techniques designed to ensure the mouse driver is present and active. This description is followed by a discussion of the basic mouse operations. Finally, simple assembly language and QuickBASIC programs using these operations are presented to demonstrate the integration of these operations

into an applications program. All the discussion assumes the Microsoft Mouse specifications. Devices manufactured by other companies may differ, but most should operate properly within the scope of this article.

Readers should have no difficulty trapping their own mice (mouses?) in their own application programs. Many game programs can be enhanced by the incorporation of mouse functions. Those readers who have version 5.0 of MS-DOS will find a game file denoted "NIBBLES.BAS." Modifying this file to use a mouse may be an interesting and instructive exercise.

## The Mouse Driver

Any peripheral device requires a "device driver." These drivers are subroutines which handle the communication with the device. Drivers for some devices, e.g. the standard console, keyboard, or printer are installed and/or activated as a part of the booting process when the computer is first turned on, and require little or no setup by the user. Others require explicit loading and control. The mouse driver is an example of the latter.

Usually a mouse driver is installed in one of two ways. The driver can be installed in the CONFIG.SYS file by setting DEVICE=MOUSE.SYS using the SYS file provided with the mouse device. More commonly, the mouse driver is installed using the AUTOEXEC.BAT file by including a call to a command program like MOUSE.COM. If the mouse was purchased in recent years, the latter method is established by a setup program included with the mouse. A "control panel" program is also established to permit adjustment of mouse parameters.

Driver subroutines are often executed

through interrupts which are numbers used by the microprocessor to find specific subroutines. This task is accomplished in MS-DOS through a table located at the very beginning of memory called the "Interrupt Vector Table (IVT)." This table is an array of four-byte addresses ("FAR" calls) which are the entry points to the subroutines. When the microprocessor is told to execute an interrupt, it multiplies the interrupt number by four to find the offset address in the IVT of the far address of the corresponding subroutine. It uses this far address to call the subroutine. This process ensures that interrupt calls are independent of the operating system versions. Personal computers using Operating System 2 (OS/2) have IVTs with eight-byte addresses. OS/2 applications are beyond the scope of this article.

The interrupt number reserved for mouse drivers is interrupt 33h (hexadecimal or base 16) or 51 in the decimal number system. Interrupts are normally expressed in the hexadecimal number system. It is possible for a mouse driver to use some other interrupt, but interrupt 33h was reserved by Microsoft for the mouse driver.

Before any mouse function is called, the user should ensure that the mouse driver is installed. If calls to int 33h are attempted when no driver is present, the call may be returned without action or the computer may become unresponsive, in which case the computer must be reset. Thus, an applications program should establish the presence of the driver before proceeding.

Microsoft recommends the following procedure to establish the presence of the driver. First, get the address of the mouse driver using int 21h function 35. This interrupt returns the segment:offset address of the mouse driver subroutine in registers

es:bx. One should not attempt to read the segment:offset address directly from the IVT because the operating system may intercept a specific interrupt and redirect the call. The use of int 21h function 35 is the only sure way of obtaining the segment:offset address. There is a notable exception to this rule which is described later.

If the segment:offset address of the mouse driver is zero, then the driver is not present. Otherwise, one must examine the data at the given segment:offset address. If the byte at this address is CFh, the driver is not present. Otherwise, one can assume the driver is present. An explanation, seldom given in the literature, is in order. A zero (0000:0000h) address implies a subroutine was not installed for this interrupt because unused portions of the interrupt table are normally initialized to zeros. Any attempt to call this interrupt would likely result in the computer becoming unresponsive. On the other hand, a nonzero address could be vectoring the system to a subroutine that simply returns - a process that produces nothing but keeps the system responsive. In this case, the instruction at the segment:offset address would be "Interrupt Return (IRET)" which is coded as CFh. Therefore, CFh at the segment:offset address implies that the driver is not installed. Any other data at this segment:offset address indicates that a driver is installed. One can assume that this driver is for a mouse because int 33h is reserved for that purpose. However, int 33h can be vectored to any subroutine. Therefore, int 33h may not be vectored to the mouse driver. This situation would represent very poor programming practices for which the guilty programmer should be condemned to some excruciating eternal punishment. **WARNING:** Some Microsoft literature on this subject appears to be in error. To avoid confusion, these errors will not be reflected nor discussed in this article.

### Mouse Driver Functions

After one has determined the presence of the driver, one must call the driver's function 0 to ascertain that the mouse is functioning. The driver might be present, but the mouse may not be physically connected. More about this function in a moment. First, more general discussion of the driver and driver functions is necessary.

The driver is called through int 33h. For assembly language programs and some recent versions of higher level languages, this process is straight forward. Older versions of higher level languages do not permit direct calls to interrupts. In this latter case, intermediate subroutines are supplied with the mouse driver software in a library of object code. Older Basic interpreters require another technique which is described later in this article.

The basic Microsoft Mouse driver has thirty numbered functions, but only twenty-three are documented for use by applications programmers according to the references used for this article. Only a few of these functions are necessary for most applications. Fortunately, these functions have simple calling arguments.

### The Call to the Mouse Driver and Calling Arguments

The driver requires data to determine the functions it must perform. This data is passed to the driver through registers or memory locations (including the stack segment) depending upon the computer language in use. Fortunately, it is not necessary to understand the exact mechanism used by most languages. The mouse manufacturer will normally provide subroutines which handle the transfer of this data to the driver. The user must provide variables to these subroutines in the form of an argument list. In the case of the Microsoft Mouse, this list consists of four integer variables and/or array variables. For example, a call to the driver from a FORTRAN language program could be

```
CALL MOUSES (ITHREE, IBUTTONS,  
            HORIZONTAL, IVERTICAL)
```

where MOUSES is a subroutine in the mouse library provided by the manufacturer. ITHREE points to a memory location containing the integer 3 for function 3. IBUTTONS, HORIZONTAL and IVERTICAL point to memory locations where the status of the mouse buttons and the position of the mouse cursor will be stored upon execution of the subroutine. The programmer must ensure that the mouse library containing MOUSES.OBJ is accessible to the linker when the program is compiled and linked.

Assembly language programs are exceptions to this rule. The driver is called directly by assembly language programs and the data is passed through registers. Most calls to the driver use the four general registers: ax, bx, cx, and dx. Calls to the driver requiring the passage of array data will use other registers as well. This latter type of call is not discussed in this article. The reader will find clear descriptions of these more complex calls in the Microsoft Mouse Programmer's Reference Guide. The FORTRAN example could be coded in assembly language as in Figure 1.

Note that the registers ax, bx, cx, and dx are the arguments corresponding to ITHREE, IBUTTONS, HORIZONTAL, and IVERTICAL respectively in the FORTRAN

program segment.

If one does not have the mouse library routines, one can write an assembly language program to interface a higher level language program with the driver. This interface program would be assembled to an object file which, in turn, would be used in lieu of the MOUSE.LIB programs. Although the program need not be much more complex than the assembly language example just given, the programmer would need a good knowledge of the subroutine interface conventions of the higher level language. A detailed discussion of these conventions is beyond the scope of this article. But, a thorough discussion of these conventions is usually included in the documentation of the higher level language, often under the topic "Mixed Language Programming."

After the program listings and at the end of this article there is a short assembly language procedure and the corresponding object file memory image which will serve as an interface between the QuickBASIC program and the mouse driver for the readers who do not have a mouse library. This procedure can be assembled by a Microsoft assembler. Be sure the object file is named MOUSE.OBJ. Include this file as a second object file when the main program is linked. If the reader needs a MOUSE.OBJ program for another programming language, the listed program can be used as a model; but the conventions of that other language must be satisfied. **WARNING:** The listed interface procedure was only tested with the listed BASIC program. It may require modification if more elaborate calls are used.

If the reader does not have an assembler, the memory image data following the procedure can be used. This data is the exact memory image of the MOUSE.OBJ file. One can use the DEBUG program supplied with the operating system to generate the object file. **DO NOT** attempt to use the DEBUG program to assemble the procedure. One needs an object file. The DEBUG assembler produces a command (.COM) file. Proceed as follows:

- a. Call the DEBUG program.
- b. At the prompt (-), type N MOUSE.OBJ and depress Enter(Return). The DEBUG program does not check the name of the file to ensure it is a command file; so, this filename should be accepted.
- c. Type D and depress Enter(Return). The current segment:offset of the code segment will be displayed followed by data on the same line which is the image of

```
mov     ax,3           ;Put an integer 3 into register ax
int     33h           ;Call mouse driver
mov     ibuttons,bx   ;Store buttons' status
mov     horizontal,cx ;Store cursor's horizontal position
mov     ivertical,dx  ;Store cursor's vertical position
```

Figure 1



the data in the sixteen bytes of memory (called a paragraph) at that address. The next seven paragraphs of memory will be displayed in the same manner. Take note of the first segment:offset address. It will have the form xxxx:0100, for example, 1F55:0100.

- d. Starting at the address noted in step c, use the DEBUG program's Fill (F) command to place the memory image data listed at the end of this article into memory exactly as listed.
- e. When all the data (about 11 paragraphs or 165 bytes) is entered, display the paragraphs with the D command and check that the memory image is correct. Note that only five bytes of data in the eleventh paragraph are entered. The values of the remaining data in this paragraph can be any number, zero is fine.
- f. Using the DEBUG program's Register (R) command, display the BX register. Ensure that this register is 0000. If not, type a 0 after the : and depress Enter(Return). Otherwise, just depress Enter(Return). In the same way, display the CX register. Type A5 after the : and depress Enter(Return).
- g. Save the object file by typing W 0100 and depressing Enter(Return).
- h. Quit the DEBUG program with the Quit (Q) command and check to ensure that a file named MOUSE.OBJ was created with a size of 165 bytes.
- i. If the View file feature of MS-DOS is available, display the file in hexadecimal format or use the DEBUG program again to display the stored file. Ensure that the file is exactly as listed. If it is, the object file is correct and may be used. Otherwise, correct the file before proceeding.

Although the old elementary BASIC Interpreter call is similar to the calls to the mouse driver from the high level languages, there are some marked differences. The most important difference is that the sub-routine interface to the mouse driver is not an external or library program. The CALL command in these older BASIC interpreters indicates an entry point into code defined elsewhere in the program. That entry point into the driver is not the beginning of the code, an important point. Consider the following BASIC interpreter program:

```
10 DEFINT A-Z
20 DEF SEG=0
30 DRVSEG=256*PEEK(207)+PEEK(206)
40 MOUSE=256*PEEK(205)+PEEK(204)
50 IF DRVSEG OR MOUSE THEN 90
60 DEF SEG=DRVSEG
70 IF PEEK(MOUSE)=207 THEN 90
80 PRINT "Mouse driver installed.":END
90 PRINT "Mouse driver not present.":END
```

This program determines whether or not the mouse driver is present. According to the Microsoft Mouse Programmer's Reference Guide, subsequent calls to the driver located by lines 30 and 40 of this program would be made through the state-

ment

```
CALL MOUSE(W,X,Y,Z)
```

where the value of MOUSE is first incremented by 2 to establish the correct BASIC entry point. Unfortunately, all the BASIC interpreters to which I have access permit calls to object code in the same way as the FORTRAN example above. The techniques required to force the interpreter into this older mode of operation are so extreme that they abrogate the test results. Therefore, I cannot verify the Microsoft approach. Also BASIC interpreters vary in how they implement calls. If the reader has one of the older BASIC interpreters, some experimentation may be required to find the correct procedure.

I've verified the simple BASIC program above and a description of this program is instructive. Line 10 types all variables as integer because no other type is used by this example. The next line defines the segment address (as in segment:offset) for use by the subsequent PEEK instructions. Now we come to the exception to the rule about using int 21h function 35 to find the address of the driver. Older BASIC interpreters have no commands which permit direct calls to interrupts. Thus, it is appropriate to read the IVT directly with the PEEK instruction because a change to the IVT could affect the interrupt 21h function 35 vector as much as the interrupt 33h vector. The IVT begins at segment address 0000h which is the reason the working (not code) segment is defined as 0. Now one can read the segment:offset addresses at int 33h in the IVT.

We know that the segment:offset address for the entry point of the mouse driver should be at 0000:0CCh because the IVT begins at 0000:0000h and has a four-byte entry for each interrupt number (4X33h=CCh). The lowest addressed byte of the four-byte entry is the least significant offset address byte. The highest addressed byte is the most significant segment address byte. This arrangement is confusing. For example, assume the following hexadecimal string is found at 0000:00CCh

```
EC 04 A5 0A
```

where ECh is the byte at address 0000:00CCh and 0Ah is the byte at address 0000:00CFh. The least significant offset address byte of the driver is ECh and the most significant offset address byte is 04. Likewise, the most significant segment address byte of the driver is 0Ah and the least significant address byte of the driver is A5h. Thus, the segment:offset address of the driver in this example is 0AA5:04ECh. It is important to understand this arrangement.

Now look at line 30 in the BASIC program example. Here, the segment address of the driver is found. Consider the least significant byte found by the instruction PEEK(206). The offset address 206 was found by multiplying 51 or 33h by 4 and

adding 2. Remember that the least significant byte in the segment address of the driver is the third byte in the four-byte string. Since the first byte is number 0 (4X51+0), one would add 2 to 4X51 to point to the third byte. The fourth byte is the most significant byte of the segment address of the driver and would be found by PEEK(4X51+3)=PEEK(207). DRVSEG is a word length variable and the PEEK function returns only bytes. Therefore, it is necessary to multiply the high order byte by its place value which is 16<sup>2</sup>=256 before adding it to the least significant byte. The reader should now understand the logic behind the coding of line 30 and should apply this logic to understand the coding of line 40.

In line 50 of the BASIC program, the values of DRVSEG and MOUSE are logically ORed. The only time that the result of an OR operation can be zero is if both numbers upon which the operation is performed are zero. If the result is zero, the segment:offset address is the beginning of the IVT (0000:0000h) and the driver is clearly not installed. A message is printed to this effect. If the result is not zero, then either the driver is installed or the subroutine is a simple IRET. Before one can say that the driver is installed, the possibility of a simple IRET must be eliminated. One proceeds in this task by redefining the working segment address as the driver segment address found in line 30 by the statement

```
60 DEF SEG=DRVSEG
```

Then one can determine if the byte at the driver segment:offset address is the processor instruction IRET which is CFh or 207 in the decimal system. If this processor instruction is found by the code in line 70 of the example program, then the mouse driver is not present and a message is displayed to that effect. Otherwise, a message is displayed indicating that the mouse driver is present.

Recent operating systems and some BASIC compilers or interpreters relocate data and code in memory in an unpredictable manner. The BASIC program just described should work on the vast majority of systems, but may fail on some systems. If this program fails to work, check the code carefully and ensure that the memory location information is used immediately, that is, before an operation which could relocate data or code. I've no experience with systems that relocate code in a manner that makes calls to drivers and address information tenuous, but I understand such systems exist.

At this point, the reader should have a grasp of what is required to call the mouse driver from any language and what steps are necessary to ensure the mouse driver is present. However, the fact that a mouse driver is installed does not mean that the



hardware is properly installed. The mouse driver should be instructed to check the status of the hardware and reset mouse circuits before the mouse is used. These procedures are accomplished through mouse driver function 0.

### Function 0 — Reset and Status

Mouse driver function 0 resets the mouse circuits to default values or configuration and returns the status of the mouse. The only input argument in the call is the first argument which is zero. The first argument is always the function number. The output is returned through the first two arguments in the call. The last two arguments in the call are dummy arguments and can have any value.

For the purpose of this and subsequent function descriptions; ARG1, ARG2, ARG3, and ARG4 are the first, second, third, and fourth arguments in the call. These arguments are registers ax, bx, cx, and dx respectively in assembly language programs. Even though an argument may have no purpose, it must be included in the call. Arguments not used as input should be set to zero. Thus, a call to function 0 might look like

CALL MOUSE (ARG1, ARG2, ARG3, ARG4)

where ARG1=0. The input values of the other arguments have no affect on the function, but should be set to zero. However, when the driver returns control to the calling program, ARG1 and ARG2 will contain data. If the mouse is connected and the hardware is working properly, ARG1=-1. Otherwise, a zero is returned. ARG2 will contain the number of buttons the mouse has if ARG1=-1. The Microsoft Mouse has two buttons. The other arguments are not used, but they must be present.

Even if this function indicates that the hardware is functioning, the mouse may still not be functioning. The driver performs only limited checks.

### Function 1 — Display Cursor and Function 2 — Hide Cursor

Many folks are confused by the cursor display when they first begin to program for a mouse. The screen cursor, the one we regularly see on the console display, is not linked to the mouse cursor. It can be linked, but the programmer must write code to do so. By default, the mouse cursor is NOT shown, making the use of the mouse difficult. Mouse function 1 sets a flag which enables the display of the mouse cursor. The mouse cursor will remain displayed until the mouse is reset or the cursor is disabled by mouse function 2, Hide Cursor. Both mouse functions 1 and 2 are called with ARG1 set to the function number. The other arguments are dummy arguments and no data is returned. These functions have no effect on the screen cursor.

### Function 3 Buttons Status and Cursor Position

Now that the mouse cursor is displayed, one can start using the mouse. If one moves the mouse, the cursor moves as well. This cursor movement is produced by the mouse driver when the mouse senses movement. In order to use this process, the position of the cursor at any given instant in time must be determined. Function 3 is the primary means of obtaining this information.

The only input to this function is the function number in ARG1. The button status, at the instant this function is called, is returned in ARG2. If the left button is depressed, bit 0 of ARG2 is set (equal to 1). Otherwise, this bit is 0. If the right button is depressed, bit 1 is set. Otherwise this bit is zero. Thus, ARG2 can have the values 0 (no button depressed), 1 (left button depressed), or 2 (right button depressed). Normally, this function call is in a loop where it is continually called and no action is taken unless ARG2 is nonzero.

ARG3 returns the horizontal cursor position and ARG4 returns the vertical cursor position. These values depend upon the video adapter or the adapter it emulates. In all cases, the values are the addresses of a pixel (picture element or the smallest resolvable dot) on the screen. The pixel in the upper left hand corner of the screen has the horizontal value 0 and vertical value 0 for a coordinate of (0,0). The largest values are in the lower right hand corner. The first coordinate is the horizontal value and the second is the vertical value.

If one is using a Color Graphics Adapter (CGA) or a video mode supported by the CGA, the screen has 640 X 200 pixels. The pixel in the lower right hand corner has the coordinate (640,200). This scheme is true regardless of the CGA mode used. For example, some CGA modes have a screen with 320 X 200 pixels, but the mouse software will still see 640 X 200 pixels. Enhanced Graphics Adapter (EGA) modes use screens with either 640 X 200 pixels or 640 X 350 pixels. The EGA can operate in the CGA modes, sixteen-color modes with 640 X 200 pixels, or 640 X 350 pixel modes. The mouse software will recognize 640 X 350 pixels in the EGA 640 X 350 pixel modes, but only 640 X 200 pixels in the other EGA or CGA modes. In most monochrome systems, the mouse software/hardware detects each pixel. Video Graphics Array (VGA) adapters normally emulate the EGA and have additional modes with resolutions of 640 X 480 pixels or greater. There are such a wide variety of VGA adapters on the market that it is not practical to discuss the characteristics of each. The reader can adapt the programs at the end of this article to determine the characteristics of specific adapters.

When the video circuits are operating in a text mode, the mouse cursor will move horizontally and vertically in increments the size of the character box, that is, the number of horizontal and vertical pixels required to display a character. For example, if the size of the character box is 8 X 16 (vertical X horizontal), each time the mouse cursor moves one unit horizontally it will move 16 pixels. Likewise, each unit of vertical motion spans 8 pixels. The default address of the cursor is the address of the pixel in the upper-left corner of the character box. For example, assume the video mode is a text mode with 80 X 40 characters. The character box in this example is 8 X 8. The address of a mouse cursor in the third line (line 2, lines and columns are zero-based, that is, numbered beginning at 0) and fourth column (column 3) is (24,16). Thus, the horizontal address of a mouse cursor in a text mode is the product of the column number (zero-based) and the horizontal size of the character box (usually 8, 14 or 16). Likewise, the vertical address is the product of the line number (zero-based) and the vertical size of the character box (usually 8 or 9).

In graphics video modes, the mouse cursor moves from pixel to pixel for each unit of motion. The address of the cursor is the address of the pixel in the upper-left corner of the character box.

The default text cursor is a reverse video character box. The default graphics cursor is an arrow terminating at the upper-left corner of the character box and whose tip is the pixel which has the address of the cursor. Other cursor forms are possible, but their creation is beyond the scope of this article.

### Function 4 Specify Cursor Position

In addition to finding the position of a mouse cursor, one can set the mouse cursor to a specific position on the screen. Mouse driver function 4 is used for this purpose. ARG1 is set to 4. ARG2 is not used. ARG3 is set to the desired horizontal coordinate (if in text mode, be sure this number is a multiple of the horizontal size of the character box). ARG4 is set to the desired vertical coordinate (if in text mode, be sure this number is a multiple of the vertical size of the character box). No values are returned. The default position of the mouse cursor is in the center of the screen.

### Function 7 Specify Limits for Horizontal Cursor Position

One can limit the mouse cursor position to only one part of the screen. Many application programs, particularly Windows type programs, will use this feature. Function 7 of the mouse driver is used to limit

the horizontal position. ARG1 is set to 7. ARG2 is not used. ARG3 is set to the desired minimum horizontal coordinate. ARG4 is set to the desired maximum horizontal coordinate.

**WARNING:** If maximum and/or minimum positions are set outside the range of the video display, the cursor can exceed the video display limits and disappear.

## Function 8

### Specify Limits for Vertical Cursor Position

Function 8 sets the limits on the vertical coordinates of the cursor. ARG1 must be set to 8. ARG2 is not used. ARG3 is set to the minimum vertical coordinate. ARG4 is set to the maximum vertical coordinate. Ensure that these values are within the range of the display or the cursor may disappear.

## Other Functions

The other mouse driver functions allow the programmer to create special cursors, set the characteristics of the mouse driver and hardware, emulate the light pen, and obtain mouse data in different ways. The smallest unit of distance that the mouse can move and sense is about 1/200 of an inch. This unit is called a "Mickey" after Mickey Mouse, I assume.

The number of mickeys of mouse motion required to move the cursor one pixel on the screen can be set by mouse driver function 15. Function 19 determines how fast the cursor moves across the screen. These functions are just two examples of the kind of control one can exert through the mouse driver. Details may be found in the Microsoft Mouse Programmers Reference Guide.

## The Demonstration Programs

The assembly and QuickBASIC programs at the end of this article demonstrate the functions discussed in this article. They are largely the same. However, the assembly language program handles some operations which are performed by the BASIC compiler. The BASIC program requires the MOUSE.LIB or an object program or subroutine to interface with the mouse driver.

Both programs check to ensure the driver is present. If the user has a system which is not responding properly to this section of code, this code can be eliminated. However, the computer system may become unresponsive and require resetting the first time the driver is called if the driver is not present. The first call to the driver is to ensure that the hardware is operating and to reset the mouse system to the default values.

If the mouse system is working properly, the mouse cursor is activated. Remember that the default condition for the mouse cursor is off. Then limits of the

cursor movement are set. Both programs only allow the mouse to move in a box defined by the coordinates (8,16) and (512,184). Normally and by default, these limits would be (0,0) and (640,200). Note that the BASIC program assumes a SCREEN 0 video display with 80 X 24 characters. The assembly language program is explicitly set to video mode 3 after the original video mode is saved. The user may wish to vary both the video modes and cursor limits to observe the differences.

Mouse driver function 3 is called to obtain mouse data. Even before function 3 executes, mouse motion will move the cursor on the screen. The driver controls this motion independently of the applications program. Some folks are reluctant to program their mouse for fear they will be required to program the mouse/cursor interface, which they are not. Note that function 3 is in a loop which repeats as long as ARG2 is zero. Depressing either mouse button generates a nonzero ARG2 and the loop is discontinued. If ARG2 (or register bx) is 1, the left mouse button was depressed and the coordinates of the cursor are displayed. Otherwise, the right button was depressed and the program terminates.

There is a keyboard input required after the mouse buttons are depressed. This input has only one function which is to prevent multiple displays caused by the fact that most computers will process the data faster than a person can depress and release the mouse button. Remember that data is intercepted and processed by function 3 for as long as the mouse buttons are depressed unless one provides a means of delaying the data loop until the user can release the button.

Finally, the programs reset the mouse system before the program terminates. This action is a good practice to prevent the retention of the mouse cursor into other programs or subroutines. The remarks or comments in the programs explain other program details.

## Conclusion

The reader should now understand how to incorporate basic mouse functions in application programs. A mouse makes the use of a computer a much less tedious task. In addition, the reader should see that incorporating mouse functions into an applications program is a relatively simple task that can pay substantial dividends. Even complex programs like computer aided design packages should seem less formidable to write. With a little experience, virtually anyone who operates a personal computer can quite easily and routinely write programs incorporating mouse operations. Good luck trapping your mouse.

## References

1. Microsoft Corporation, Microsoft Macro Assembler 5.0 Mixed-Language Programming Guide, 1987, Microsoft Corporation, Redmond, Washington.
2. Microsoft Corporation, Microsoft Macro Assembler 5.0 Programmer's Guide, 1987, Microsoft Corporation, Redmond, Washington.
3. Microsoft Corporation, Microsoft Mouse Programmer's Reference Guide, 1986, Microsoft Corporation, Redmond, Washington.
4. Microsoft Corporation, Microsoft QuickBASIC 4.0 BASIC Language Reference, 1987, Microsoft Corporation, Redmond, Washington.
5. Microsoft Corporation, Microsoft QuickBASIC 4.0 Programming in BASIC Selected Topics, 1987, Microsoft Corporation, Redmond, Washington.
6. Que Corporation, DOS and BIOS Functions Quick Reference, 1989, Que Corporation, Carmel, Indiana.

## Trademarks

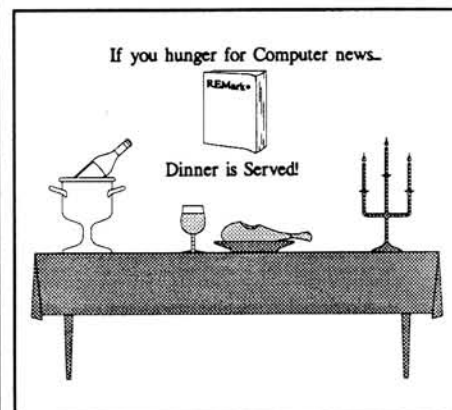
IBM, IBM PC, CGA, EGA, and VGA are trademarks of the International Business Machines Corporation.

Mickey Mouse is a trademark of Walt Disney Productions.

Microsoft, Microsoft Mouse, MS-DOS, OS/2 and QuickBASIC are trademarks of the Microsoft Corporation.

The following programs are intended for instructional use only. No expressed or implied warranties are made for these programs with respect to merchantability or fitness for any other purpose. The user assumes all responsibility for any damages resulting from the use of these programs.

The BASIC program is written in QuickBASIC interpreter/compiler and must be linked with MOUSE.LIB or a separate object program or subroutine written to access the mouse driver. The assembly language program calls the driver directly — no library or access subroutine is required.



**BASIC Program**

```

REM Copyright (C) 1991 by David W. Lind
REM Define all variables as integers and declare the subroutine.
10 DEFINT A-Z
20 DECLARE SUB MOUSE (ARG1, ARG2, ARG3, ARG4)
REM If the user wishes to select a video mode different from the default;
REM the statement SCREEN n, where n is the video mode, and WIDTH c,1, where
REM c is the number of columns and 1 is the number of lines, should be
REM inserted here.
REM Determine if the mouse driver is installed. A direct read
REM of the interrupt vector table is used here to demonstrate this
REM method. A call to interrupt 21h, function 35, Get Interrupt Address
REM should be used by compilers/interpreters which have interrupt call
REM features.
30 DEF SEG = 0
40 DRVSEG = 256 * PEEK(207) + PEEK(206)
50 MOUSEOFF = 256 * PEEK(205) + PEEK(204)
60 IF (DRVSEG OR MOUSEOFF) = 0 THEN 80
70 GOTO 90
80 PRINT " Mouse driver is NOT present, program terminated.": END
90 DEF SEG = DRVSEG
100 IF PEEK(MOUSEOFF) = 207 THEN 80
REM Determine if the mouse hardware is present and reset using mouse driver
REM function 0.
110 ARG1 = 0
120 ARG2 = 0
130 ARG3 = 0
140 ARG4 = 0
150 CALL MOUSE(ARG1, ARG2, ARG3, ARG4)
160 IF ARG1 = 0 THEN 180
170 GOTO 190
180 PRINT " Mouse hardware is not connected, program terminated.": END
190 PRINT " Mouse operational and reset to default values."
REM Display mouse cursor.
200 ARG1 = 1
210 CALL MOUSE(ARG1, ARG2, ARG3, ARG4)
REM Set cursor limits to a box defined by (8,16) and (512,184).
220 ARG1 = 7
230 ARG3 = 8
240 ARG4 = 512
250 CALL MOUSE(ARG1, ARG2, ARG3, ARG4)
260 ARG1 = 8
270 ARG3 = 16
280 ARG4 = 184
290 CALL MOUSE(ARG1, ARG2, ARG3, ARG4)
REM The following block of code reads the position of the mouse cursor
REM when the left mouse button is depressed and displays the mouse cursor
REM coordinates. When the right mouse button is depressed, the program
REM terminates. Ensure that the arguments from the previous call not
REM used as input to the next call are zeroed.
300 ARG1 = 3: ARG2 = 0: ARG3 = 0: ARG4 = 0
310 CALL MOUSE(ARG1, ARG2, ARG3, ARG4)
320 IF ARG2 = 0 THEN 300

```

```

330 IF ARG2 = 2 THEN 390
340 PRINT " Current mouse cursor coordinates are: (":ARG3;":":ARG4;":)"
350 PRINT " Depress any KEYBOARD key to continue followed by ";
360 PRINT "right mouse button to stop."
REM The following call to the keyboard ensures only one set of coordinates
REM are processed. The loop will process coordinates for as long as the
REM left mouse button is depressed. On a fast computer, like an 80386,
REM even the quickest finger in the west will produce multiple messages
REM with the default mouse settings. An adventurous user could eliminate
REM the following step and move the mouse around while depressing the
REM left mouse button. A large number of coordinates will be processed.
370 INPUT: " ", DUMMY$
380 GOTO 300
REM Reset mouse and end.
390 ARG1 = 0: ARG2 = 0: ARG3 = 0: ARG4 = 0
400 CALL MOUSE(ARG1, ARG2, ARG3, ARG4)
410 END

```

**Assembly Language Program**

COMMENT \* Copyright (C) 1991 by David W. Lind. \*

```

data
videomode DB ?
hund DW 100
mess1 DB ' Mouse driver is NOT present, program terminated.', '$'
mess2 DB ' Mouse hardware is not connected, program terminated.', '$'
mess3 DB ' Mouse operational and reset to default values.', '10,13,' '$'
mess4 DB ' Current mouse cursor coordinates are: (', '$', '$'
mess5 DB '10,13,' Depress any KEYBOARD key to continue followed by'
        DB ' right mouse button to stop.', '10,13,' '$'
        ENDS
code
SEGMENT
ASSUME cs:code,ds:data ;Begin code segment
                                ;Tell assembler the segment labels
start:
mov ax,data ;Put data segment address into ax
mov ds,ax ;Transfer to ds register
COMMENT * Save video mode and set to 80 X 24 character text mode. *
mov ax,0F00h ;Put function number into reg. ax
int 10h ;Get video mode
mov videomode,al ;Save video mode
mov ax,0003h ;Put function # and mode into ax
int 10h ;Establish video mode 3
COMMENT * Determine if a mouse driver (software) is installed. *
mov ax,3533h ;Put function number and interrupt
                                ;number into register ax
int 21h ;Call interrupt address
mov ax,es ;Move es register data to ax reg.
or ax,bx ;Logical or ax and bx registers
jnz driver1 ;If not zero, driver may be present
jmp nodriver ;Else, driver is not present
driver1:
mov al,es:[bx] ;Put byte at es:bx into reg. al
cmp al,0CFh ;Compare to IRET code
jne driver2 ;If not IRET, driver is present

```



```

driver2:
jmp     nodriver
       ;If IRET, no driver

COMMENT * Determine if the mouse hardware is present and reset. *
mov     ax,0
int     33h
       ;Put function number in reg. ax
cmp     ax,-1
       ;Call status and reset function
je      hardwareok
       ;Check status
jmp     nohardware
       ;If *, hardware is present
       ;Else, terminate program

hardwareok:
mov     ax,900h
lea     dx,mess3
int     21h
       ;Put function number into reg. ax
       ;Put offset of mess3 into reg. dx
       ;Display message

COMMENT * Display mouse cursor. *
mov     ax,1
int     33h
       ;Put function number into reg. ax
       ;Call display mouse cursor

COMMENT * Set cursor limits to a box defined by (8,16) and (512,184).
mov     ax,7
mov     cx,8
mov     dx,512
int     33h
       ;Put function number into reg. ax
       ;Put minimum coordinate into cx
       ;Put maximum coordinate into dx
       ;Call set horizontal limits

mov     ax,8
mov     cx,16
mov     dx,184
int     33h
       ;Put function number into reg. ax
       ;Put minimum coordinate into cx
       ;Put maximum coordinate into dx
       ;Call set vertical limits

COMMENT * The following block of code reads the position of the
mouse cursor when the left mouse button is depressed
and displays the mouse cursor coordinates. When the
right mouse button is depressed, the program terminates. *

loop1:
mov     ax,3
int     33h
       ;Put mouse function number into ax
       ;Get button status and cursor
       ;position
cmp     bx,0
je      loop1
       ;Compare button status to zero
       ;If zero, call function again
cmp     bx,1
je      displaycoord
       ;Is left button depressed?
jmp     loop1end
       ;If yes, display coordinates
       ;Else, end process

displaycoord:
dx
mov     ax,900h
lea     dx,mess4
int     21h
       ;Save register dx (vertical coord)
       ;Put function number into reg. ax
       ;Put offset of mess4 into reg. dx
       ;Display message
xor     dx,dx
mov     ax,cx
div     hund
add     al,48
push    dx
mov     dl,al
int     21h
       ;Convert hundreds digit to ASCII
       ;Save register dx (remainder)
pop     ax
mov     ax,200h
int     21h
       ;Put ASCII number into register dl
       ;Display byte
       ;Pop remainder into ax
       ;Convert to BCD
       ;Convert to ASCII
       ;Save register ax
mov     dl,ah
mov     ax,200h
       ;Put ASCII number into reg. ax
       ;Put function number into reg. ax

nohardware:
mov     ax,900h
lea     dx,mess2
int     21h
       ;Put offset of mess2 into reg. dx
       ;Display message
xit:
xit:

```

```

21h
ax
:Display byte
:Restore ax register
:Put ASCII number in dl
:Put function number into reg. ax
:Display byte
:Put ", " into register dl
:Display byte
:Restore dx (vertical coordinate)
:Move vertical coordinate to ax
:Zero register dx
:Divide by 100
:Convert hundreds digit to ASCII
:Save register dx (remainder)
:Put ASCII number into register dl
:Put function number into reg. ax
:Display byte
:Pop remainder into ax
:Convert to BCD
:Convert to ASCII
:Save register ax
:Put ASCII number in dl
:Put function number into reg. ax
:Display byte
:Restore ax register
:Put ASCII number in dl
:Put function number into reg. ax
:Display byte
:Put ", " into register dl
:Display byte
:Put function number into reg. dx
:Put offset of message into reg. dx
:Call display message

COMMENT * The following call to the keyboard ensures only one set
of coordinates are processed. Loop1 will process
coordinates for as long as the left mouse button is
depressed. On a fast computer, like an 80386, even the
quickest finger in the west will produce multiple messages
with default mouse settings. An adventurous user could
eliminate the following two steps and move the mouse around
while depressing the left button. A large number of
coordinates will be processed. *

ax,0
int     16h
jmp     loop1
       ;Put function number into reg. ax
       ;Read keyboard
       ;Continue loop

loop1end:
COMMENT * Reset mouse. *
mov     ax,0
int     33h
jmp     xit
       ;Put function number into reg. ax
       ;Call reset mouse
       ;Exit program

nodriver:
mov     ax,900h
lea     dx,mess1
int     21h
jmp     xit
       ;Put function number into reg. ax
       ;Put offset of mess1 into reg. dx
       ;Display message
       ;Exit program

nohardware:
mov     ax,900h
lea     dx,mess2
int     21h
       ;Put function number into reg. ax
       ;Put offset of mess2 into reg. dx
       ;Display message
xit:
xit:

```

```

mov ds:[si],cx
mov si,[bp+6]
mov ds:[si],dx
pop si
pop bp
pop 8
ret ENDP
mouse ENDP
END

```

MOUSE.OBJ Memory Image

```

80 0B 00 09 4D 4F 55 53 45 2E 41 53 4D D4 96 24
00 00 06 44 47 52 4F 55 50 04 44 41 54 41 04 43
4F 44 45 0A 4D 4F 55 53 45 5F 54 45 58 54 05 5F
44 41 54 41 7D 98 07 00 48 33 00 05 04 01 DC 98
07 00 48 00 00 06 03 01 0F 9A 04 00 02 FF 02 5F
90 0C 00 00 01 05 4D 4F 55 53 45 00 00 00 D5 88
04 00 00 A2 01 D1 A0 37 00 01 00 00 55 8B EC 56
8B 76 0C 8B 04 8B 76 0A 8B 1C 8B 76 08 8B 0C 8B
76 06 8B 14 CD 33 8B 76 0C 89 04 8B 76 0A 89 1C
8B 76 08 89 0C 8B 76 06 89 14 5E 5D CA 08 00 59
8A 02 00 00 74

```



Zenith Data Systems  
Power Supplies  
Complete Board Assemblies

# SURPLUS COMPUTER PARTS

20 Meg. Hard Drives  
From \$0.00

- Largest Surplus Electronics Dealer in Western Michigan
- In Business Over 40 Years
- Over 12,500 square foot Store and Warehouse

Examples of Zenith Data Systems Salvage Products:

|                                          |            |
|------------------------------------------|------------|
| 386-16 1 Meg. Memory Board               | \$190.00   |
| Z-100 Power Supplies                     | From 15.00 |
| Laptop Carrying Cases — 5 Sizes          | From 8.00  |
| Z-449 Video Boards (CGA/VGA)             | 75.00      |
| AT Dual Controller Board                 | 40.00      |
| Laptop Keyboards                         | From 2.00  |
| 20 Meg. New & Used External Tape Backups | From 50.00 |
| 8087 & 80287-6 Co-Processors             | 50.00      |
| Numeric Keypads for Laptops              | 6.00       |
| Laptop Modems                            | From 5.00  |
| PC & AT 84-key Keyboards - Used          | From 5.00  |
| Z-148 Power Supplies                     | From 15.00 |
| Zenith Data Systems 101-key Keyboards    | From 10.00 |
| 286 Laptop Motherboards                  | 500.00     |
| Power Supply Fans                        | From 5.00  |
| 40 Meg. Hard Drives for 286 SuperPort    | 300.00     |
| Laptop Replacement Batteries             | From 6.00  |

Many Other Zenith Data Systems Salvage Parts Available  
All Surplus/Salvage Sold AS IS - NO GUARANTEE -  
No Catalog - Please Call for Quotes

## Surplus Trading Corp.

THE HOUSE OF EVERYTHING "ALMOST"  
2700 N. M-63, P.O. BOX 1082  
BENTON HARBOR, MI 49022

CALL: (616) 849-1800 FAX - (616) 849-2995 (Orders Only)  
We Ship U.P.S.-C.O.D. (ONLY) Sorry - No Foreign Shipments

COMMENT \* Pause for two seconds to permit message display before video display reset. \*

```

mov ah,2Ch ;Put function number into reg. ah
int 21h ;Call time
mov bh,dh ;Store seconds in bh

xit1:
push bx ;Save register bx
mov ah,2Ch ;Put function number into reg. ah
int 21h ;Call time
pop bx ;Restore bx
cmp dh,bh ;Compare new and old time
js xit2 ;If negative, adjust for turnover
sub dh,bh ;Else, find difference
cmp dh,2 ;Has two seconds passed?
ja xit3 ;If so, finished
jmp xit1 ;Else, continue sample

xit2:
add dh,60 ;Add 60 seconds to current time
sub bh ;Now find difference
cmp dh,2 ;Has two seconds passed?
ja xit3 ;If so, finished
jmp xit1 ;Else, continue sample

xit3:

```

COMMENT \* Restore original video mode (int 10h, function 0) and terminate program. \*

```

xor ax,ax ;Zero register ax
mov al,videomode ;Put video mode into register al
int 10h ;Set video mode
mov ah,4Ch ;Put function number in ah
int 21h ;Call terminate program
ENDS ;End code segment

segment stack
DW 64 DUP (?) ;Begin stack segment
ENDS ;Establish a 128-byte stack
;End stack segment

END ;End of program and define start

```

Interface Procedure (in lieu of MOUSE.LIB)

```

.MODEL MEDIUM
.CODE
PUBLIC mouse
PROC
push bp
mov bp,sp
push si
mov si,[bp+12]
ax,ds:[si]
mov si,[bp+10]
mov bx,ds:[si]
mov si,[bp+8]
mov cx,ds:[si]
mov si,[bp+6]
mov dx,ds:[si]
int 33h
mov si,[bp+12]
mov ds:[si],ax
mov si,[bp+10]
mov ds:[si],bx
mov si,[bp+8]

```

---

---

# Add Seven Hard Drives to Your System!

**Frederick O. Smetana**  
**5425 Parkwood Drive**  
**Raleigh, NC 27612**

Have you ever needed additional hard drive storage beyond that provided by the two drives already present in your machine? I did. I wanted to have all the files — both text and graphics — associated with a 900-page textbook I was working on available on line. These totaled about 50 megabytes. I only had about half that much free space available on the existing two drives. What to do?

If you are like me you certainly don't want to discard two perfectly functional drives and replace them with larger ones simply to get some additional storage space. REMARK has had several articles about adding additional floppy drives to a system via a CompatiCard. I've used that route myself. But I've not seen an article about adding additional hard drives. Through some phone calls I learned that DOS only supports two hard drives (the DOS 5 manual says that DOS 5 will support more drives but doesn't tell you how to do it) and that any additional drives must be added via a suitable controller and an appropriate device driver. I was looking for an additional 80 megabytes or so and the most reasonable prices at the time were for a Seagate ST-296N SCSI drive. SCSI host adapters range in price from \$50 to \$700 or more. The least expensive host adapters only permit one to attach two SCSI drives to a system and just replace the standard disk controller. The more expensive ones provide very fast data access via large RAM caches and the ability to talk to other SCSI devices.

A few months before I began this project a couple of articles in BYTE discussed a new software interface for SCSI host adapters developed by Adaptec. The

host adapter mentioned in the article was the non-caching, bus-mastering 1542B. Bus mastering was something I had not used before. I knew that IBM had provided the capability to operate one bus mastering device as part of the PC/AT system design. (The EISA design specification provides for multiple bus masters.) The bus master signals the CPU that it wants control of the system bus by putting the CPU into a hold state. It then causes the data transfer to be managed by a DMA chip. When the transfer is complete, the bus master returns control of the system bus to the CPU. This method is supposed to be about twice as fast as the PIO (Programmed Input-Output) method which most lower-cost host adapters use. I was also interested in installing a SCSI host adapter in the system at some point because I was running out of expansion slots and had thought that some day I would like to add things like a tape drive and CD ROM to my system. Most such devices, I've noticed, talk to the machine via a SCSI interface. I now had the excuse I needed to order the 1542B kit which includes the drive, a cable, and four device drivers.

The Adaptec 1542B is a 2/3 length card with AT bus type edge connectors. All the chips except the BIOS and the micro-code for the onboard 8085 CPU are surface mounted. A 34-pin ribbon connector for floppy drives is provided as are a 50-pin ribbon connector for internal SCSI devices and a 50-contact female Centronix connector on the tie-down bracket for external devices. Board configuration is set by jumpers on six multi-position jumper blocks. The factory settings can be used without change in systems having only SCSI drives.

I examined the 1542B installation manual carefully and noted that the board is accessed via I/O port 330H, although that can be changed via jumpers to one of five other addresses. (A standard AT hard drive controller is accessed through port address 1F0H, drive #1 at 320H, and drive #2 at 322H.) One of six IRQ lines can be used to signal the CPU with IRQ 11 being the default. One can use either DMA channel 0, channel 5, channel 6, or channel 7. DMA 5 is the default. The onboard BIOS is located in the CPU's memory space at one of four 16K locations between C800H and DFFFH. The default is DC00H. In order to use the onboard BIOS, the board must be accessed at 330H. The onboard BIOS overwrites part of the standard system BIOS so that the first SCSI drive will appear to the system as C: if no other hard drive is present or as D: if one other hard drive is present. In either of these cases no additional software drivers are needed if the system is running DOS.

From the number of available DMA channels it is evident that up to four 1542B's can be installed in a machine at the same time. However, only one board can have its floppy controller enabled and only one board can have its BIOS enabled since the BIOS will not work when the board address is other than 330H. If you are going to use the board only for the third or additional drives, the BIOS should be disabled, as I discovered, after the drives are low-level formatted (this routine is located in the BIOS). Having it enabled interferes with Zenith's boot from any drive capability. It assumes that the floppy boot drive is always A:. When it loads and sees that there are already two hard drives in the system it



attempts to return control to drive A:. If you are actually booting from drive B: the system naturally crashes. In my machine drive A: is a 1.2 meg 5.25" floppy and drive B: is a 1.44 meg 3.5" floppy.

Once the jumpers are correctly set, the board installed in the machine, and the cable connected it's time to add the lines

```
DEVICE = ASPI4DOS.SYS  
DEVICE = ASPIDISK.SYS
```

to CONFIG.SYS. The first driver provides the software interface between the host adapter and the system. The second provides the interface between the host adapter and the disk drive or drives. An additional driver is required for each type of device added to the SCSI bus. The /L switch in ASPI4DOS permits one to partition a drive into several logical units. The /D switch causes the information the driver has found about the drive to be displayed on the screen. The /W and /V switches cause a 64K buffer to be created in extended memory. No additional drivers are required if you wish to add additional SCSI disk drives (up to a total of seven) to the SCSI bus. You must make sure, however, that each drive has a unique SCSI identification number. Most SCSI drives have hardware to enable the user to set these ID numbers. My drive is ID 0 and the host adapter is ID 7. Also be careful to follow the manual's instructions about proper SCSI bus termination.

I first tried to install the host adapter in a Zenith ZW-241 (a 6-MHz no-wait-state AT clone) but I could not format the Seagate 296N from the onboard BIOS. I was advised by Adaptec technical support that this machine probably did not support bus mastering to the extent necessary. This seemed like a credible explanation since I had been given a similar reason as to why an AOX Master 386 upgrade would not work in this machine. (AOX therefore designed a different upgrade card for Zenith machines, one that replaced the Z-241's CPU card. The Z-241 is a backplane machine with the basic system hardware on two plug-in cards.) The gentleman I spoke with at Adaptec said I might try the less-expensive 1522 controller (which operates differently) instead of the 1542B if I wanted a SCSI interface in that machine.

I then tried the 1542B in my Zenith Z-386/16. This was Zenith's first 386 machine and it also used a backplane architecture instead of a system board. The video card in the machine is an OEM VGA card made by Paradise for CompuAdd. After much experimentation, I was able to format the drive. The manual was of little help here since it did not indicate how long one should wait for the process to complete. (The seller of the drive, Hard Drives International of Phoenix AZ, advised that the drive was not low-level formatted. They claim to provide all necessary low-level formatting

software but the ON-Track Disk Manager they sent was for IDE drives. I also called Seagate. They said that all SCSI drives leave the factory low-level formatted but apparently only for Seagate host adapters.) I once permitted the machine to tell me it was formatting for seven hours when actually nothing happened. The drive access light was on but it did not format. I could only tell what message the machine would write to screen during formatting by looking at the BIOS with DEBUG; there certainly was nothing in the manual to tell one what to look for. I tried again and this time I could hear the drive making some noises. This time the access light seemed not to be operating except perhaps sporadically.

When it finished I found I could run AFDISK, the Adaptec program that writes the partition information on the SCSI drive. AFDISK cannot be used if the drive is either C: or D:. In that case you must run FDISK from DOS.

SCSI drives require some time to initialize from a cold boot. The ASPI4DOS driver will not load properly if this initialization has not been completed beforehand. Thus, the power to the SCSI drive should be applied about 10 seconds or so before the computer attempts to load the operating system. In my system the SCSI drive is mounted externally in an old Z-148 computer case. This case has space and sufficient power for a second SCSI drive. It sits atop the active computer, allowing the use of a short connecting cable.

I found also that in order to write the partition information on the SCSI drive, it was necessary that I not load the onboard BIOS from my CompatiCard IV secondary floppy controller. The "Please Wait" message in AFDISK would flash and the machine would lock if I did not disable the CompatiCard BIOS. Once the drive was high-level formatted, I tried reenabling the CompatiCard BIOS but there seemed to be problems accessing data on the SCSI drive. There is no address conflict in the base addresses of the two cards (362H for the CompatiCard and 330H for the 1542B) and no conflict in DMA channels or interrupts (CompatiCard uses DMA 2 and IRQ 6); however, if the CompatiCard BIOS is loaded, INT 68 is enabled at address 9FC1H and the available memory is given as 639K bytes. If the CompatiCard BIOS is loaded from a device driver instead, it is loaded in low memory and the total DOS memory is reported as 640K bytes. Using the device driver takes about 7K of conventional memory, however. The CompatiCard BIOS has seven possible locations but the Z-386/16 copies the system BIOS and video BIOS to RAM memory segments F000H and E000H respectively which seems to prevent the CompatiCard from using either of the two possible addresses in the E000H segment. It's a little difficult to tell how

much space the 1542B BIOS requires so I did not try the CompatiCard at the DE00H address. (I had wanted to try loading the CompatiCard BIOS after the 1542B BIOS. Previously, I had used the CompatiCard BIOS at address CE00H.) I suspect that part of the problem is that both cards do something to INT 13.

The machine is running Zenith's version of MS-DOS 4.01 and has 2 meg of 32-bit memory and an additional 2 meg of 16-bit memory. ASPI4DOS.SYS is loaded with the /L /D /V /W switches. I don't know whether the /V and /W switches are necessary but some software seemed to run a little better with them enabled. The /L switch seemed to be necessary even though the drive has only one partition.

The Zenith BIOS (my machine has version 2.7E dated 1-30-89) permits one to boot from any drive connected to the MFM controller. Zenith also provided a dual boot utility so that DOS and OS/2 could reside in the same partition. When I was running the machine with DOS 3.3 (and without the 1542B) I also had OS/2 1.1 on the C: drive and could boot either operating system depending upon whether I held down the "ALT" key or not during boot. During the upgrade to DOS 4.01, I deleted all the OS/2 files. After installation of the 1542B I tried to boot OS/2 from a floppy disk on drive B: as I had often done previously. The operation got about halfway through the boot process when the machine locked because of an error at location #0210:0797. I recognize that the ASW-1420 driver, the one for OS/2 — in contrast to the ASW-1410 driver, the one for DOS — will not attach the SCSI drive to the system if there are also MFM drives installed; however, I did not think the installation of the 1542B controller would prevent the system from even booting OS/2!! (By the way, the Zenith will boot IBM OS/2 — at least it would previously — but my PS/2 model 70-A21 at the office will not boot Zenith OS/2; the boot process just continues to cycle.)

I attended COMDEX East in 1990 and was therefore able to get a copy of Windows 3.0 from Microsoft at a very attractive price. However, I haven't been able to get it to run long enough in any mode to bother with it. I've also had some interesting experiences with QEMM. The Adaptec drivers do not run satisfactorily when installed after QEMM. They will work correctly if installed before QEMM in CONFIG.SYS.

The technical support from the dealer, Smart Ideas, Inc., in Boston was of no help at all. They really do not understand the product or how it is supposed to work. They did return my calls promptly and suggested a number of possibilities but it was obvious they knew no more than I did. Adaptec's technical support representative took my plea for help and promised

someone would call. The gentleman who later called could only spare a short time but after I described the symptoms, he had an immediate and helpful response (see above). He knew exactly how the board should work! A very pleasant surprise.

I left the controller installed in the machine because my primary need at that time was for a lot of drive space. (I have in addition two 40 meg MFM drives on the primary controller.) I have done without OS/2 since because I use it primarily to compile and run an occasional large Fortran program. I now do those programs on my office machine which boots OS/2 version 1.3.

I have been generally pleased with the operation of drive F: (It's drive F: because I have one of the 40 meg MFM drives partitioned into two logical units.) It has served my extra storage needs quite effectively. I have found, interestingly, that file transfers, particularly those of large files (most often

when copying to the floppy drives but occasionally when copying to the Seagate ST-251-1, an add-on to the original system) contain one or two random two-byte errors (two adjacent characters are corrupt). I have not been able to pin down the cause of these errors which appear to result from a failure to transfer all the bits in a character. There are perhaps some slight bus timing incompatibilities or problems getting the data out of the buffer in extended memory. Looking at the file on the SCSI drive with the TYPE command or with a word processor shows that the file is stored correctly. What diagnostic software I have doesn't work well with SCSI drives, particularly when the drive is not C: or D:.

Since one never has enough disk storage I am now looking at a technique some of the bulletin board people around here are using to increase their disk storage inexpensively: adding a bridge controller as one device on the SCSI bus. This control-

ler is a circuit board about the dimensions of a hard drive planform that doesn't plug into the system bus (but receives its power using a Y-splitter from the drive power leads). It looks to the host adapter like a SCSI device and the BIOS on the 1542B writes the information the two need to communicate to a ROM on the bridge controller. The bridge controller in turn can run two MFM or RLL drives (two different models). If all the SCSI ID's were assigned to bridge controllers you could connect 14 additional drives to your machine. The local BB's get a bunch of old, inexpensive MFM or RLL drives which they then use to store masses of data and talk to them in this fashion.

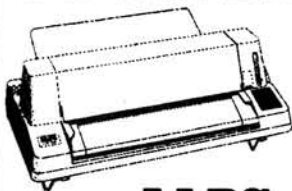
Now if we could get a suitable device driver to permit multiple SCSI drives to run under OS/2 version 2.0 we won't have to spend a bundle on bigger drives to hold both the operating system and our old files. At least, that's my dream. ✨

# W S Electronics

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

1106 State Route 380, Xenia, Ohio 45385

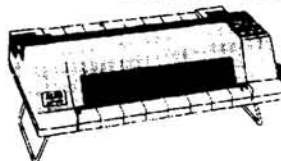
## YOU AND YOUR BIG IDEAS.



- \* 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.

# QUIKDATA - 16 YEARS OF H/Z SUPPORT!

YOUR H/Z STORAGE & ENHANCEMENT EXPERTS!

## ACCELERATE YOUR PC/XT/AT!

For your H/Z241, 248 and 386/16 we have the direct replacement **WESTERN IMAGING Z-33 ZUNI MOTHERBOARD** that's just loaded with features. Super fast 33Mhz 80386 main board with ten expansion slots and built in are two serial ports, one parallel port, floppy disk controller and an IDE disk controller! That gives you 10 slots. Add video (and your old MFM controller if you desire) and you still have plenty of expansion left! Support to 32 meg on-board RAM using SIMMs, and 32 additional megs with memory expansion.

**WIZ-33** - \$795 with 0K RAM **80387-33** - \$195 Coprocessor  
**SIM1X9-8** \$52 1 meg X 1 80ns SIMM DRAM  
**SIM256-8** - \$13.50 256K X 1 80ns SIMM DRAM

From Sota Technologies, Inc., the fastest and most proven way to give new life to your H/Z PC/XT computer, giving it -AT compatible speeds! Turn your turtle into a -286 rabbit with a 12.5 Mhz 80286 or 80386 16Mhz SX accelerator board. Complete with 16K on-board CACHE.

The EXP12 286i is the effective solution, making your H/Z150/160/150/158/159 series of computers, or any general PC/XT computer faster, in many cases, than a standard IBM AT type computer! **You won't believe your stop watch!**

**EXP-12** - \$249  
**EXP386** - \$395 Much faster 16Mhz 80386 SX version

## MEMORY UPGRADES

Note: All memory upgrades come without memory chips. 150ns 256K DRAMs are \$1.59 as of this printing.

**Z150MP** - \$17 Will allow you to upgrade your H/Z150/160 to up to 704K on the main memory board, using up to 18 256K DRAM chips.

**EAZYRAM** - \$89 Upgrades EaZy PC from 512 to 640K

**ZMF100** - \$55 Will allow you to upgrade your H/Z110/120 (old motherboards; with p/n less than 181-4918) to 768K system RAM. Requires 27 256K DRAM chips.

**Z100MP** - \$55 Similar to ZMF100 above, but for new motherboards with p/n 181-4918 or greater.

**3MB RAM BOARD for Z241/248 computers** is an excellent memory card. Will backfill your 512 to 640K, and provide both extended and expanded RAM; all can coexist. Uses 100ns M256-10 RAM chips, 36 per megabyte desired. Minimum of 18 DRAM chips required (\$1.79 each).

**EVATRD** - \$109

**Z248/12, Z286LP RAM UPGRADE Z605-1** consists of 2MB SIMM 80ns RAM kits to upgrade your H/Z systems.  
**Z605-1** - \$115

**Z386/20, Z386/25, Z386/33, Z386 EISA 2MB SIMM 80ns UPGRADE** to add increments of 2MB to these systems. Two required.  
**ZA3600ME** - \$79

**ZA3800MK** - \$269 4 megabyte SIMM upgrade for above. Must have 4-1 meg SIMMs installed first.

We also have memory upgrades for just about every H/Z desktop and laptop unit except the 171, 181, 183, and 184 series. Tell us what you need and we can quote you a price.

## WINCHESTER UPGRADE KITS

**PCW40** - \$319 Complete MFM winchester setup for a H/Z150, 148, 158, 159, 160, PC etc. Includes 42 meg formatted half-height Segate ST-251 28ms drive, controller, cable set, doc.

**ST-251-1** - \$269 Bare drive only

**PCW80** - \$459 80 meg with Segate ST-4096 full size drive.

**ST-4096** - \$395 Bare drive only

**DTCON** - \$59 PC/XT hard drive controller board

**WDATCONF** - \$95 1:1 interleave HD/floppy controller for AT's

**IDECON** - \$69 AT bus IDE/floppy controller for placing an IDE hard drive in any AT compatible.

## FLOPPY DRIVE SAMPLE

**MF501** - \$71 5" 360K DS/DD drive  
**MF504** - \$75 96 TPI 1.2 meg AT/Z100 drive  
**MF353** - \$71 720K 3.5" drive in 5" frame  
**MF355** - \$75 1.4 meg 3.5" AT drive in 5" frame  
**TM100-2R** - \$65 40tk DS refurb (H8/89/Z100 PC type)

## ANY TYPE FLOPPY IN YOUR PC/XT/AT

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

**CCARD2** - \$79 **CCARD** - \$99 **CCARD4** - \$119

**OR, add a floppy to any laptop or PC with a parallel port** easily and inexpensively. With **Backpack**, you simply connect the external unit to your parallel printer port (do not lose printer function), install software, and away you go! No expansion boxes needed for laptops, and no slots required. Want a 2.8mb/1.4mb/720K floppy on your MinisPort? Want to add a 1.2 meg to your laptop? Want to add an additional drive to your desktop? Plug it in and go. 2.8MB 3.5" version will read and write 2.8 meg, 1.4 meg, and 720K format.

**BPACK2.8** - \$279 **BPACK1.4** - \$229  
**BPACK1.2** - \$229 **BPACK360** - \$229

**ADD AN EXTERNAL 80MB TAPE BACKUP DRIVE** to any laptop or PC with the Microsolutions Backpack QIC-80 tape backup system. Plugs into a parallel port to give you an affordable way to easily backup your hard drive and/or transfer data. Uses DC2000 tapes. Fast! **BPACKT8** - \$445

**ADD AN EXTERNAL HARD DRIVE** - Just like above, but these units are completely portable hard drives. Think of the uses, including the security! 40 and 100MB units available by Microsolutions.

**BPHD40** - \$429 **BPHD100** - \$575

## 8-BIT/Z100

**We carry a full line of replacement boards, parts and power supplies for the H/Z89/90 and Z100. We also have some H8 boards available. We continue to fully support and carry a full line of hardware and software products for the H8/H89/90 and Z100 computers.** Of course we carry much more. Find out!

## OTHER STUFF

Quikdata also stocks ROM upgrades and batteries for most H/Z PC/XT/AT computers, spike protection filters, backup power supplies, tape backup units, modems, printers, cables and ribbons, disk drives and diskettes of all types, external hard drive and floppy drive enclosures, cables and connectors, video monitors and cards, memory chips and cards, ICs, joysticks, accessory cards, mice, software and much more! **Need a PC/AT computer?** Tell us what you want and we will quote you a price on one of our custom assembled QD computers made up to meet your demands.

Call or write in to place your order, inquire about any products, or request our free no obligation catalog. VISA and Master Card accepted, pick up 2% S&H. We also ship UPS COD and accept purchase orders to rated firms (add 5% to all items for POs). All orders add \$5 S&H. Phone hours: 9AM-4:30PM Mon-Thu, 9AM-3PM Friday. Visit our **bulletin board**: (414) 452-4345. **FAX**: (414) 452-4344.

**QUIKDATA, INC.**  
2618 PENN CIRCLE  
SHEBOYGAN, WI 53081-4250  
(414) 452-4172



## Getting Started With ...

# Q&A

Alan Neibauer  
11138 Hendrix Street  
Philadelphia, PA 19116

Most software can be easily categorized as either word processing, database, spreadsheet, graphics, or some specific application such as accounting. Of course, there are also integrated programs, such as Microsoft Works and Lotus Symphony. But for the most part, the individual parts of an integrated program are less powerful than stand-alone programs. The spreadsheet part of an integrated package, for instance, can not compete with Excel or Lotus 1-2-3.

Q&A, however, is not as easy to categorize. Q&A is usually classified as a database management program. Yet it also includes a moderately robust word processing program. But in addition, the database and word processing elements of Q&A are fully integrated so you can share data between them to create form documents and database reports of all types.

While Q&A doesn't include its own spreadsheet or graphics modules, it can use spreadsheets and graphic files created with most other popular programs. So, for instance, you can integrate Q&A with a program such as Lotus 1-2-3 to get the best of all worlds. You can even use Q&A with databases created by programs such as dBASE and Oracle, and with documents created by WordPerfect and other popular programs. In fact, because of Q&A's Intelligent Assistant modules, which you'll learn about shortly, Q&A is a perfect "front-end" for dBASE files that would require complex programming in dBASE itself.

In this article, we'll look at each of Q&A's major parts to see what a remarkable program this really is.

### Database Management

Q&A File is Q&A's database manager system. Unlike other systems, it is "form based." This means that you design on the screen what would appear to be a replica of a paper form. However, you can have a maximum of 2045 fields on each form (or record) spread over 9 pages, or screens.

Each record can be a maximum of 65,536 characters and there is a maximum of 534,266 records per database.

The form concept allows you to visualize your data in a very realistic way, particularly if you maintain paper copies of the forms. This allows for very easy transition from a paper to an electronic system. Figure 1, for example, shows the first page of a database form for tracking client data. The form automatically becomes the means to enter, retrieve, sort, and print data.

Also, you can use the same field label more than once if it makes your form easier to understand. Unlike most database systems, duplicate field labels won't confuse Q&A. In most database programs, each field label must be unique. That is, you could not have two fields with the same label.

This makes it difficult when you have a form that contains two similar items, such

as two persons. You'd have to have one field called something like Last-Name1, another called Last-Name2. That's because most database systems use the field label, which is the prompt that appears on the form or screen, as the field name. The field name is how a database system distinguishes one field from the other. So if you have two field names that are same, such as Address, the database system wouldn't know which address you want when printing reports or form letters.

In a paper form, however, there may be labels that are not unique. You might have two fields called Phone, but each in a different section of the form that clearly distinguishes between them. Q&A separates field labels from field names. If a field label is unique, Q&A assigns the field the same name as the label. So the only field with the label Client Code is given the name Client Code. When field labels are

| Customer Record           |                    |               |
|---------------------------|--------------------|---------------|
| Customer Code: [REDACTED] |                    |               |
| Name:                     |                    |               |
| Address:                  |                    |               |
| City:                     | State:             | Zip Code:     |
| Primary Contact           |                    |               |
| Last Name:                | First Name:        | MI:           |
| Title:                    | Phone: ( ) -       |               |
| Order Information         |                    |               |
| Last Order Date:          | Amount:            | Total Orders: |
| Invoice No.:              | Credit Limit:      |               |
| Products Purchased:       | Taxable:           |               |
| Shipping Zone:            | Preferred Carrier: |               |

CLIENTS.DTF    New Record 1    of 1    Total Records: 10    Page 1 of 2  
Esc-Exit    F1-Help    F3-Delete form    F7-Search    F8-Calc    F10-Continue

Figure 1

not unique, Q&A assigns each field its own internal field name. For example, it would assign the first field labeled Name the name Name, the second Name field Name1, and the third Name field Name2.

However, your on-screen form can appear just like a paper form and you might never have to use the internal field names yourself unless you are performing some advanced programming functions.

Q&A provides seven field types: text, numeric, money, date, hours, yes/no, and keyword.

Text fields may contain any character you can enter from the keyboard, so they are perfect for names and addresses, and fields that contain numbers as well as punctuation characters, such as phone numbers, social security numbers, and nine digit zip codes.

You only designate a field as numeric if you plan to use it in calculations. The money type is a special numeric field that Q&A automatically formats into dollars and cents, with two decimal places. For instance, when you enter 922.56 into a money field, Q&A displays \$922.56. When you enter whole numbers, such as 99, Q&A adds the decimal places for you — \$99.00. You can create formulas using any combination of money fields, numeric fields, and numbers.

The date field is useful because Q&A formats the dates for you automatically. You can enter a date in any of 20 different formats, such as 11/16/92, 16.11.1992, or even 92-11-16. Q&A will automatically convert the date and display it as Nov 16, 1992. You can also use date fields in calculations, such as calculating the number of days or years between two dates to print aging reports of unpaid invoices. Likewise, you can use the hours field to record the start and stop times of activities, keeping track of your time for billing purposes.

When you want to record a simple Yes or No answer (like a logical true or false), use the Yes/No field. This field represents only two possible values — true or false — and can be entered Yes, y, True, t, or 1 to represent a true condition; No, n, False, f, or 0 for a false condition.

Q&A also allows a keyword field. You can use these fields as well to generate reports. One problem of designing a database is handling a repeating field. These are fields that contain the same type of information. For example, suppose you want to record the activities that all of your organization's members enjoy. You could create a number of similar repeating fields such as

Activity1  
Activity2  
Activity3  
Activity4  
Activity5

But since you don't know the maxi-

mum number of activities a member can have, how many times would you repeat the field? Would five fields be enough? Five? Ten? With most database software, you can only avoid this problem by creating multiple databases, and linking them together with complex commands or programming statements. Q&A, however, provides the keyword field just for this purpose.

The keyword field can contain a series of entries separated by semicolons. So you can have one field called Activities and enter information such as

Activities: Baseball; bowling; stamp collecting

You can then print reports listing which members enjoy a specific activity because Q&A can separate the values in the field into individual items. Q&A will even match keywords regardless of capitalization, as long as they are spelled the same. If your keyword field contains the names of sports, Q&A will consider baseball, Baseball, and BASEBALL the same.

### "Limitations"

Technically, Q&A is not a relational database. A relational database is one which can combine data from more than one database, or file. With a relational database, for instance, you can create invoices by combining order information from one file, with customer information and price data from other files. But while Q&A is not relational, you can perform these same functions using Q&A's Report and Mail Merge functions, and other special commands. In fact, most relational-type tasks can be performed quickly and easily using Q&A.

In addition, Q&A does not include a full-fledged programming language like that found in dBASE IV. In reviews and articles, Q&A is always classified as a "non-programmable" database.

Database programming languages are often used to extol a software's power and versatility. But in many respects, programming languages are provided to make up for deficiencies in the program itself. The software's designers are making you responsible to develop features that they failed to provide.

Most database languages are traditional in terms of computer programming. You write a program that is separate from the database. The program must include explicit statements for every action. You have to tell it the name of the database, the fields to use, and each step you want the program to perform. It's no wonder that even some simple tasks need programs that are hundreds, if not thousands, of lines long.

Now Q&A really does have a programming language, but not in the traditional sense. You write a Q&A program directly

on the database form. Each programming statement automatically relates to the fields and data with which you're dealing. The programming commands work in conjunction with all other Q&A functions, not as a separate program that must interact with the database independently and under its own control. Often, one programming statement in Q&A is the equivalent of hundreds of lines in some other database's language.

To be fair, if you need to develop a large and complex custom database application, then you may need a database program that has a more traditional language. But if you want to be a database user, and not a programmer, you can build many sophisticated applications using Q&A.

In my book *Mastering Q&A* (Sybex), I give an example of setting up an invoicing system using Q&A. The entire system requires less than 50 actual lines of "program code." Yet, it allows you to enter invoices as they are received, and automatically update inventory and client data through a processing posting from one database to another.

Most of the functions, including lookup tables and professional-looking reports, are performed automatically in Q&A. Even the more advanced and custom functions are performed by built-in commands, functions, and your own macros from within your database file, something Q&A calls programming the form. The same functionality would require hundreds, if not thousands, of lines in a dBASE program, and quite a bit of programming knowledge.

Now what about those complex functions that Q&A's non-programmable structure doesn't allow? With imagination, a great deal of programming can still be accomplished. For example, Figure 2 shows part of a Q&A form that can be used to check and update stock on hand while completing an invoice. Notice that this "non-programmable" program includes familiar structures such as IF-THEN and subroutine calls using BASIC-like Gosub and Return commands.

This routine, which is written directly on fields on a form, checks to see if this is the first invoice for a session. If it is, it reads the values from the stock database and inserts them in ten fields on this page of the form. It then uses these fields to check stock on hand as each invoice is completed. If this isn't the first invoice for the session, it gets the on-hand values from the second page of the previous invoice.

Of course, it seems quite an overhead to maintain the program and stock quantities directly on the invoice form, but Q&A's posting process takes care of this. The forms used to record the invoice are part of a temporary transaction database. When the session is complete and the invoices printed, a custom macro posts the pertinent information to the actual invoice data-

```

P1: <#85: if #101<>"" then goto #38; #86 = 101; IF
@XLOOKUP(@FILENAME, #8-1,"Invoice","101") ="" THEN gosub #87 else
gosub #89; return

P2: #86

P3: <#87: if #86<=110 then {GOSUB #88; #86 = #86 + 1; goto #87};
return

P4: <#88: @(#86)=@XLU("STOCK", #86,"STOCK NUMBER","ON HAND");
return

P5: <#89: if #86<=110 then {GOSUB #90; #86 = #86 + 1; goto #89};
return

P6: <#90: @(#86)=@XLU(@FILENAME,#8-1,"Invoice", #86); return

101: #101          106: #106
102: #102          107: #107
103: #103          108: #108
104: #104          109: #109
105: #105          110: #110

```

Figure 2

base, and automatically updates the stock and client databases. The transaction file is then archived and the temporary invoices deleted.

This is quite a process for a non-programmable database!

### Printing Reports

Q&A provides a printing function from within its database module for printing individual and groups of records. So, for instance, you can print your records neatly arranged without leaving the database file or writing even one line of code.

What if you want a report in columns? Or a statistical cross tabulation? How about using your printer's fonts and special effects? You can generate these quickly and easily in the Report module.

Columnar reports can even include column breaks, or control breaks. These occur when the contents of a field changes, and it signals Q&A to perform some special processing. For example, look at the report shown in Figure 3. There is a column break on the first column, with subtotals being calculated for each employee. But there is another column break in the second column for the Style field. The break in that column occurs when the style changes from Condo to Single. Subtotals are calculated on both control breaks — for each style and for each agent.

If the field is a date field, you can also specify the column break to occur when the day, month, or year changes. So you could get a printout of your daily invoices or monthly bills.

In addition, Q&A allows derived and invisible columns. A derived column in one calculated from information in other fields. You could print a column on sales commissions, for example, even if a database does not include a Commission field. The information in that column can be calculate, or

derived, from values in other columns. You can even designate an invisible column. That is a column that will not print but includes values that are used for generating a derived column or column break.

You can also automatically generate crosstab reports, as shown in Figure 4. This report shows the relationship between the days of attendance and the average grade by class. Looking at the report, you can see that as attendance decreases, so does the average grade. The report also breaks down the analysis by class. If you look at the bottom row of the report, it appears that the average grade increases somewhat with class. The other rows of the table, however, reflect the general relationship between attendance and grades, although well-attending freshman seem to have a slightly higher average grade (95) than well-attending se-

niors (93).

The Report function also allows you to gather information from multiple data files, performing relational-like tasks.

### Word Processing

Q&A includes a word processor that can be used to create documents of all types and can be your primary word processing software. While it lacks some features found on more powerful programs, such as footnotes and index generation, it includes a spelling checker and thesaurus. It even lets you merge documents, graphics and drawings created by other programs.

However, the word processor's main power lies in its integration with your data files. Not only can you use it to create form letters and database reports, you can merge data from multiple data files and perform calculations to prepare invoices and or-

| Agent  | Style  | Price          | Commissions  |
|--------|--------|----------------|--------------|
| Chesin | Condo  | \$79,050.00    | \$3,952.50   |
|        | Total: | \$79,050.00    | \$3,952.50   |
|        | Single | \$147,650.00   | \$7,382.50   |
|        |        | \$165,650.00   | \$8,282.50   |
|        |        | \$158,150.00   | \$7,907.50   |
| Hall   |        | \$374,100.00   | \$18,705.00  |
|        |        | \$143,250.00   | \$7,162.50   |
|        | Total: | \$988,800.00   | \$49,440.00  |
|        | Total: | \$1,067,850.00 | \$53,392.50  |
|        | Condo  | \$142,250.00   | \$7,112.50   |
| Walker |        | \$100,025.00   | \$5,001.25   |
|        | Total: | \$242,275.00   | \$12,113.75  |
|        | Single | \$99,250.00    | \$4,962.50   |
|        |        | \$37,000.00    | \$1,850.00   |
|        |        | \$157,375.00   | \$7,868.75   |
| Walker |        | \$129,275.00   | \$6,463.75   |
|        |        | \$427,400.00   | \$21,370.00  |
|        |        | \$18,275.00    | \$913.75     |
|        | Total: | \$868,575.00   | \$43,428.75  |
|        | Total: | \$1,110,850.00 | \$55,542.50  |
| Walker | Condo  | \$62,475.00    | \$3,123.75   |
|        |        | \$184,425.00   | \$9,221.25   |
|        |        | \$98,075.00    | \$4,903.75   |
|        | Total: | \$344,975.00   | \$17,248.75  |
|        | Single | \$727,250.00   | \$36,362.50  |
| Walker |        | \$147,500.00   | \$7,375.00   |
|        |        | \$283,800.00   | \$14,190.00  |
|        |        | \$76,030.00    | \$3,801.50   |
|        |        | \$141,500.00   | \$7,075.00   |
|        |        | \$138,700.00   | \$6,935.00   |
| Walker | Total: | \$1,514,780.00 | \$75,739.00  |
|        | Total: | \$1,859,755.00 | \$92,987.75  |
| Total: |        | \$4,038,455.00 | \$201,922.75 |

Figure 3



| Grade         | Year  |       |       |       | Average Grade |
|---------------|-------|-------|-------|-------|---------------|
|               | 1     | 2     | 3     | 4     |               |
| >=90          | 95.00 | 91.00 | 95.00 | 94.80 | 94.58         |
| >=80..<=89    | 85.00 | 89.00 | 86.00 | 85.00 | 85.86         |
| >=75..<=79    | 75.00 | 75.00 | 75.00 | 76.00 | 75.25         |
| >=70..<=74    | 73.00 | 74.00 | 74.00 | 73.00 | 73.50         |
| <=69          | 55.00 | 56.00 | 46.00 | 65.00 | 56.50         |
| Average Grade | 76.60 | 77.00 | 84.20 | 85.73 | 82.35         |

Figure 4

ders, catalogs, contracts, and proposals, agreements and leases. This capability can replace thousands of lines of programming code necessary in other database products.

Mailing labels are even easier to create. You can select from 50 pre-defined label formats, or create your own format for non-standard labels. To design a custom label, you just designate the number of labels across, space between labels, and lines per label sheet.

All modules of Q&A let you use your printers fonts, including softfonts, and character formats.

### Intelligent Assistant

One of the most dramatic benefits of Q&A is its ability to serve as a front end to your data. After all, a database is only useful if you can utilize its information to make timely and accurate decisions. You have to be able to quickly and easily display or print the information that you need to know.

With most database programs, you can only get specific information by learning complex syntax or generating reports. But generating a report or learning specific syntax are highly structured ways of getting information. You have to translate your question into the framework required of database. Instead of simply asking "Who is?", you have to know which fields you want to print, the syntax for selecting records, the codes for calculating results, and how the data should be sorted.

In the real world, we ask questions in a much less structured way. We question intuitively, often spontaneously, with the answer from one question often leading to others. Who are our best clients? Where are they located? What products do they buy? When was their last order?

If you had to create a report for every question you had, you'd soon run out of room on your disk, and you'd have little time for anything else.

Q&A includes Intelligent Assistant, a system that allows you to get information in this more natural way. You might start with the simple question, *How many clients owe over \$500?*

You can then immediately follow up with, *Give me their names, address and phone numbers.*

Then, after thinking about it you de-

cide to ask, *Which haven't ordered since 12/1/92?*

Notice you do not have to list specific fields or worry about syntax, but ask questions just as you would to a live assistant. Also notice that Q&A can maintain the thread of the questions. Only those clients identified by the first question are used to formulate the response to the second. For example, Q&A understands that the request for "their names" refers to the clients who owe over \$500.

IA can correctly recognize regular plurals of nouns, that is plurals created by adding s, es, or ies. So you can ask, *What cities?* when the database field is actually called City.

As shipped from Q&A, IA understands about 600 words. To ask questions in a totally natural way, you have to "teach" IA about your database and how you want to interact with it. Teaching IA is particularly useful in a home or office where persons not skilled in Q&A have to occasionally access the database. Instead of training them how to follow the structure of Q&A, teach IA how to understand a more natural language.

For instance, the client database has fields called credit limit and amount. If you want to be able to ask, *Who has exceeded their credit?* you can teach IA that "exceeded their credit" means "where amount is greater than credit limit."

You can teach IA on an ad hoc basis — teach it new words as you find it necessary — or by using a series of lessons that Q&A will take you through step-by-step. Each lesson is designed to teach IA another set of words that you'll use to communicate with it. Most words that you can teach IA using the Lessons, can also be taught to it ad hoc. But the lessons, while they do take some time to complete, let you define a complete set or words and grammar, allowing you and others to communicate with IA in a much more natural and less structured way.

Through teaching IA, you can use synonyms for fields, such as telephone or phone number for phone. You can even define synonyms for expressions or collections of fields. As you go through each lesson in this chapter, you'll learn tips on how to phrase your requests and questions to IA.

It is after the teaching process that you can ask a series of questions to further refine the data received:

- How do I reach Chesin?
- Where does he live?
- Who is his secretary?
- How much does he owe?
- What does he order?

In addition to retrieving data, IA allows you to create and change records. For example, the command *Increase Chesin's credit limit by 500 will add 500 to the value in the Credit field for Chesin's record.*

With imagination, you can easily teach Q&A another language. You can teach it synonyms for fields, such as the French le nom for name. You can define *calculez* as a synonym for calculate, so you could tell IA in French to *calculez (65 + 76 + 97)/3*. Create the synonyms *Donde* for where, *estan* for are, and *ellas* for they, so you can display locations by asking the Spanish *Donde estan ellas?*

### Query Guide

You can also get answers you need using Query Guide. Query Guide is a menu-driven assistant that guides you step-by-step through the process of generating reports, displaying records, and calculating statistics. You don't have to know how the report functions works or even the structure of your database. Just follow the prompts and menus that Query Guide displays on the screen.

For example, if you want to ask, *Which clients purchased more than \$1000?* Query Guide will help you develop the request.

Produce a report showing the name and the total orders for records where total orders is greater than \$1000.

Query Guide develops requests in fragments. It begins the process by letting you select the first fragment, or the first words of your request to Query Guide. These can be

- Find and show...
- Produce a report showing the...
- Count...
- Summarize the data by...
- Run...
- Cross-tabulate...

Think of these fragments as the start of a sentence, such as

Find and show records where total orders is greater than \$1000

or

Count the records where City is Margate and credit limit is greater than \$750

Query Guide then guides you through the process of refining and completing your request by helping you designate the fields you want to display, the statistic you want to calculate, the retrieve criteria, and the sort order.

As you select options from the screen,

Continued on Page 35

# MULTI-EDIT

## VERSION 5.01P

Dan Jerome  
801 E. 132nd Street  
Burnsville, MN 55337-3870

Multi-Edit is an ASCII editor/word processor that is MORE than user-friendly: it is a real "sweetheart;" easy to learn and easy to use! It has many of the features the more famous word processors have, but Multi-Edit's features are easier, simpler, and faster to access. Plus, Multi-Edit has some nifty features of its own. The latest version, 5.01P, is programmed to run under MS-DOS 5.0. If you are running MS-DOS 4.01 or earlier, you can order Version 5.0.

Multi-Edit is mouse-enabled, and may be operated in two modes: [1] in conjunction with Microsoft Windows (TM) or [2] as a first-rate stand-alone editor, without Windows. Procedural options are to use the pull-down menus, issue commands from the edit screen, or a combination of both.

Multi-Edit is unique among editors in that it was created to serve a dual purpose:

1. To provide a delightful, full-bodied facility for editing or writing manuscripts
2. To provide the programmer with "total convenience" by allowing him to work inside the editor during the entire program writing cycle.

Major features to enhance program writing include writing the source code, debugging it, and compiling it. There is a library of built-in features and commands to simplify these programming tasks. The "Professional Version" includes facilities for programming languages such as BASIC, FORTRAN, PASCAL, dBASE, MULTI-EDIT MACRO, and MODULA2. [NOTE: The "Standard" and "Shareware" versions allow you to work with the C and assembly (ASM) languages only.]

Undoubtedly, Multi-Edit's most significant feature is the use of **virtual memory**. This feature allows you to write or edit huge files. For large and extremely large files, Multi-Edit works most efficiently with expanded memory, but works just fine for normal-sized files in conventional memory.

The term 'virtual memory' applies to an editor that has been programmed in a unique fashion. Two **scratch files** are cre-

ated to support the current file that you are writing. As you write your document at a certain point the computer buffer becomes filled. At this time the frontmost scratch file comes into play and absorbs the extra text that you create. The second scratch file absorbs the rearmost text, as you move toward the beginning of the current file. When you save the current file or send it to disk, all files combine. The entire process is transparent to the user. With such an editor, a writer can produce huge files!

To give you an idea of how powerful this editor is, it can edit 100 files in 100 windows simultaneously. It can also edit files up to 32 megabytes in size (depending on the size of your available memory). Another plus from my point of view is that it is very flexible and can be easily melded to suit my needs. The manual is fairly well written and lies flat on your work table. This makes it easier to check on a new feature when you encounter one.

### SCOPE OF THIS REVIEW

The scope of this review is to concentrate on the text editing features from the writer's standpoint, with just a touch on programming features. For information on the programming features (of version 2.0) refer to "Multi-Ed in Review," an article written by Thomas Lasanti III. This article appears in REMark Magazine, October 1988.

### OPERATING MULTI-EDIT

Multi-Edit is called to your screen with the "ME" command. When the program appears, the status lines at the top of the screen indicate line number, column number, filename, and other data. The first thing you notice on the bottom line of the screen is a series of ten highlighted "function key labels," each label positioned above a function key. Each of these function key labels contains a command that may be executed. If you press ALT, CTRL, or SHIFT, the function key labels are switched to an

entirely new set. This highly-effective technique provides 40 quick commands that are instantly available to the user. Cursor movement is adjustable and can be extremely rapid, and scroll bars used with a mouse provide the writer quick text access. In addition, context-sensitive HYPERTEXT help screens are available at any stage of writing a document. The help screens serve as a pretty decent quick manual for the new user.

Refer to Appendix 1 for the "Multi-Edit Dynamite Quick Command List." This list does not include all possible commands, just the commands that I use the most.

The program provides 12 pull-down menus, as follows:

|        |        |         |
|--------|--------|---------|
| File   | Search | Macro   |
| Window | Text   | Install |
| Block  | Layout | Other   |
| Cursor | Print  | Quit    |

To help you to understand the power of this ASCII editor, lets summarize chief features of all the pull-down menus and closely examine two of the most-used. NOTE: When the menu option is followed by three periods, this indicates additional data from you is required before it can carry out the command.

### The INSTALL Menu

Installing Multi-Edit is like nothing I ever experienced. It creates the ME, Help, and SRC directories. The SRC files are macro source codes. If you purchased the Professional Editor, it installs the special features in a modular fashion. For example, included in this is a special disk for "Document/Spell Check," "Languages Keymaps/Other," "Communications," and "Macro Source/DeBugger." This makes it convenient to install the additional modules you initially decided not to purchase.

The INSTALL menu provides most of the default settings that you will want to set before beginning to write. Most of these are in the form of dialogue boxes that give

```

+-----FILE-----+
| Edit New File ...
| Load File Into Current Window ... <ShftF3>
| Save File in Current Window
| Save File As ... <F3>
| Information About Current File...
+-----+
| Merge File From Disk ...
| Save Block to Disk ... <CtrlF3>
+-----+
| DOS Directory Shell <CtrlF4>
| eXit Multi-Edit <AltX>
+-----Cancel<ESC>-----+

```

Figure 1. File Menu.

you option choices.

### The FILE Menu

This menu sees the most use. The best advice I could give you is to ALWAYS use a fully-qualified filename whenever the program calls for it. If you don't, the program may become confused and cause unpredictable effects. See Figure 1 for the contents of the FILE Menu.

### Edit New File

This option lets you start a new file while you are in the editor. (Of course, you can always start a file the usual way, from

```

+-----PRINT-----+
| Print Current File
| Print Marked Block
+-----+
| Printer Setup ...
| Printer Type ... Epson
| Printer Device/File ... PRN
+-----+
| Copies to Print ...1
| Line Numbering Off
| Print Margin ...1
| Eject Page
+-----+

```

Figure 2. The Print Menu.

DOS, but if you are already using the editor, this technique is more convenient.) You are presented with a dialogue box into which you type the fully-qualified new filename and extension. When you hit <ENTER>, the dialogue box clears and the editing screen reappears. The filename and path which you designated appear in the "File Status Line," about four lines down from the top of the editing window. Then you are ready to begin editing.

### Load File Into Current Window

This option lets you edit an old file while in the editor. Select it, and you are presented with a dialogue box. The "magic shortcut" is to press the TAB key. This brings up a two-column list of files from disk. To select the desired file, just move the highlight bar over the desired file using the arrow keys, and hit <ENTER>. The dialogue

box clears and the editing screen reappears. Then you are ready to begin editing.

### Save File in Current Window

You can either enable automatic file saving or you can save files manually, or both. When done writing your file, the easiest way to tuck it away on disk is to first save it by using this menu option. To clear the screen so that you can begin typing a new file, press CTRL-

W. If you opt to leave the program, just type ALT-X. Automatic file-saving is available from the INSTALL menu. CAUTION: Be sure to save your file before clearing the screen. If you don't there are built-in checks, but if you're not careful it will go to "never-never" land.

### Save File As

If you want to save a file under a different filename, select this option. A dialogue box will appear and you will be expected to type in a fully-qualified path, filename, and extension.

### Merge File From Disk

This option joins a disk file to the current file at the cursor position. When you call it up, a dialogue box appears and you select the file you want to merge by using the highlight bar, then tapping <ENTER>.

### Save Block to Disk

Save a block to disk by first marking the block, and then calling up this option. You will be asked for a new filename and extension for

the block.

### DOS Directory Shell

This option provides you with a number of on-the-spot utilities available while still in the editor. When you enter it you can do things like copy or delete files, sort directories, go to DOS and issue a command, change the directory to select and edit another file, and type "Exit" to return.

### OTHER MENUS

The WINDOW Menu provides many options that you can select to manipulate windows, including a Window List. Multi-Edit can provide up to 100 windows.

The BLOCK Menu lets you copy, move, and delete text, but it does it royally. For example, you can block a line (or lines) of text, streams of text, or columns of text. It also provides a "Pick and Put" buffer if you

want to repeat certain lines of text without retyping them.

The CURSOR Menu gives you 10 Bookmarks, and a GOTO Line Number.

The SEARCH Menu Finds Text, Finds and Replaces Text, and provides you Multiple File Search and more. You have access to global forward or backward in the file.

The TEXT Menu provides UNDO, REDO, Time and Date Stamp, Justify or Unjustify a Paragraph, or Reformat A Paragraph.

The LAYOUT Menu provides Word Wrap, Indent Style [Manual Indent, Auto-Indent, or SmartAuto-Indent]. In addition, it gives you Print Formatter, Auto-Setup, and Tabs. The Print Formatter option allows you to modify the old codes and add new printer codes to work under the PRINT menu. It also provides such niceties as generating a Table of Contents or an Index.

### The PRINT Menu

This menu is designed to print the files generated with Multi-Edit. It is another menu that will be often-used. Once you get used to using it you will miss it when editing in other programs.

**Print Current File** prints the current file. But wait, first set the defaults you want the file to have. See "Printer Setup" for the list of defaults. (Figure 2)

**Print Marked Block** lets you mark a block of text and print it.

**Printer Setup** (Figure 3) selections are as follows. To select one or more, just

```

+--SELECT A PRINTER CODE--+
| 10 cpi
| 12 cpi
| 17 cpi
| 6 lpi
| 8 lpi
| Draft
| NLQ
| [Skip the Perf]
| Underline Begin
| Underline End
| Emphasized Begin
| Emphasized End
| Italics Begin
| Italics End
| Double Wide Begin
| Double Wide End
| Bold
| Bold Off
| [Reset Printer]
+-----+

```

Figure 3. Printer Setup.

arrow the highlight bar to the proper selection and hit <ENTER>. The program offers 18 printer defaults, and you can change the defaults to suit. For example, the two codes included in square brackets are two printer codes I added to tailor the printer function for my personal needs.

The printer needs to be "RESET" every time you change CPI or LPI settings. (The norm is to combine 6 LPI with 10 and 12 CPI



and combine 8 LPI with 17 CPI or higher.) "SKIP THE PERF" is a means of printing forms with multiple pages without setting formfeeds. This topic is covered under "Formfeeds or No Formfeeds" in the "Multi-Edit Tricks" section of this review.

**Printer Type** gives you an option as to which printer you wish to use. A list of printers comes up and you move the highlight bar over the printer you are using and then hit <ENTER>. This is an important selection to make, since the codes that run "Printer Setup" depend upon which printer you select.

In case you don't have any of the selected printers there are provisions for adding your printer's name here by typing over an existing printer and for adding your printer's codes in the INSTALL menu under "Printer Setup."

**Print Margin** lets you set your left-hand print margin by selecting this option. Since this program has no defined left margin, you need to center your file on the printer paper. For instance, when I use a right margin of 76, I set this option to 10 and it comes out centered.

**Eject Page** is provided in case you neglected to type a formfeed (CTRL-L) at the end of your file, you can still get the printer to formfeed to the next perforation on the paper by executing this command.

The **MACRO** menu gives you access to creating your own keystroke macros, and lets you Run A Macro, Load A Macro, Debug A Macro, List All Macros, or List All Globals. Multi-Edit has a unique macro programming language of its own. Once you get the hang of it you will find it is relatively easy to learn and use.

The **OTHER** menu has some neat options: Line Draw, and Spell Checker. Both features work smoothly and are enjoyable to use. In fact, I prefer them over the ones included with either of my "flagship" word processors.

In case there are any programmers reading this, it also has some goodies that they may be interested in. These include an ASCII Conversion Table, Programmer's Calculator, Execute Compiler or Program, Find Next Compiler Error, and Build Template.

Be careful with the **QUIT** menu. If you select it, you are suddenly in DOS.

## MULTI-EDIT TRICKS

### Setups Hint

Whenever you encounter a dialogue box that request a pathname or especially if you are within the FILE menu telling Multi-Edit to begin a new file, be certain to provide a FULLY-QUALIFIED pathname. For example: C:\ME\DANUSER\Filename.Ext. A partially qualified pathname will get you into trouble every time.

### Left Margins

Concerning a left margin, Multi-Edit has no specific setting such as most editors/word processors have. One just begins typing at the desired margin using the "AUTO" indent cursor style. The auto-indent style will attempt to hold the margin where you set it. But if you establish a left margin of say, 5, and attempt to type your document, after a wordwrap the type will sometimes fly back to column one, ignoring the margin you set.

There is a provision for entering beginning defaults into a startup macro, but I found the easiest way is to use the "Print Margin" option from the PRINT menu. Another way is to "Create A Left Margin" described below if you really want to get fancy.

### To Create A Left Margin

You can always "create" a left margin when your document is complete. To do so, hit CTRL-HOME and move to the first line of the file. Then enter the LAYOUT menu and turn "Word-Wrap" off. Then go to "Fixed Tab Spacing" and set it to 5. (This assumes you want the text to be shifted 5 columns to the right.) Finally, change the "Indent Style" to off. Mark the text you want shifted, using F7 to begin marking; END, and DOWN ARROW to identify the text. Mark an entire page at a time if you like. Again press F7 to stop the marking. To execute the shift process, hit ALT-I. The entire field that you marked will be shifted to the right by 5 columns. To turn off marking, press CTRL-F9. [NOTE: The number of columns the text is shifted depends upon the setting you choose for "Fixed Tab Spacing." If you chose 5, the text will shift 5 columns. If you chose 10, the text will shift 10 columns.]

### Formfeeds or No Formfeeds

**Method #1: Formfeeds.** After you complete your multi-paged manuscript, you will want paginate it and add headers. Multi-Edit has its own method for doing this: type a formfeed (CTRL-L) at the end of each page. I think this is an awkward method of doing things. (I rather suspect that Multi-Edit will allow you to automatically set formfeeds, but I was unable to dig that out of the manual). Besides, I'm prejudiced: I prefer the second method.

**CAUTION:** If you decide to use formfeeds instead, and type a formfeed with the command CTRL-L, the cursor must be at the exact left margin. If it isn't, the printing will go wrong, and it will be difficult to determine the reason without a bit of experimentation.

### Method #2: Surefire Manual

**Formfeeds.** The procedure I prefer is to break up the text MANUALLY. Check the "Table of Line Numbers, Appendix 2 for details. To break up text, move the cursor to one line number at a time. Then arrow the cursor up three lines and hit <ENTER> three times to move the text down and create 3 blank lines at the end of the preceding page. Insure that the cursor is now resting on one of the lines called out for "headers," and then for the first time type your header and page number. For example, according to the Table of Line Numbers, line number 61 would be page 2. It should have three blank lines above it. An example of a typical header and line number is shown in Figure 4.

The next step is to place the header/page number text lines into the "Pick and Put Buffer." To accomplish this, put the cursor on the first header/page number line of text and tap F7. The entire line is marked. Now use the DOWN ARROW key to move to the second or subsequent lines in order to mark the pertinent text. To place the text into the "Pick and Put Buffer," just tap the keypad PLUS key. The legend on the status line (third line from the top of the screen) should say: "Text copied to buffer."

When all the pertinent text is marked, stop marking by pressing F7. Press CTRL-F9 to delete the marking.

At this time go down to the next page line number according to the Table of Line Numbers and repeat the process described for page 2. When the cursor is on the line number indicated in the table, hit keypad ENTER and the header/page text will drop into place. Now press CTRL-F9 to delete the marking. Continue to repeat this procedure until the document is paginated. By using the Table of Line Numbers you can tell which page numbers to create on which line numbers. NOTE: Despite this rather long description, I can promise you that once you have tried it you will find the process quite easy to use.

After the pages have been paginated, you should check them. Use ALT-F8 and the "GOTO" feature, checking successively line numbers 61, 121, 181, etc., and make sure that the page numbers are consecutive and that they are on the correct line number. Here is where I use my macros "DOWNGO" and "UPGO." It only takes two keystrokes to move the cursor down 60 lines from one page number to the other. Refer to Appendix 3 for the "DOWNGO" macro source code listing. Using such a macro only takes two keystrokes.

Finally, to compensate for not having a formfeed, certain steps must be taken to

modify the PRINTER menu. Install a "SKIP OVER PERF" control code that makes the printer skip over 6 blank lines, including the paper perf line. For an Epson or Panasonic printer the printer codes to use are: "SKIP OVER THE PERF" |27|78|6 and "RESET PRINTER" |27|64. Note that Multi-Edit uses the vertical line key as a delimiter. To accomplish this, go to the "PRINT" pull-down menu and choose the Epson printer and hit <ENTER>. Select F3 and a dialogue box will appear. Move down the list of printer options until you come to a blank line and insert both the function (i.e., SKIP OVER THE PERF" one line and then across the gutter add the code, |27|78|6. While you are there, you might as well add the "PRINTER RESET" function and its code, |27|64. CAUTION: DO NOT TYPE ESC, AS YOU WILL BE EJECTED FROM THE PROGRAM POST-HASTE! Just a simple |27 will do. Then exit the option and return to your screen. To check whether your work has taken, obtain the PRINT pull-down menu again and move the select bar over "PRINTER SETUP" and hit <ENTER>. You should see the new function(s) added to the list.

#### Preventing Text from Shifting Right When Moving or Copying A Block

Go to the INSTALL pull-down menu and select "EDIT SETTINGS." Tab down to "Tab Expand" and change it to "spaces." Tab down to "Column Block" and change it to "Leave Space."

The problem is that without making these settings if you plunk down a block of text in some other spot, not having left sufficient room for the new text, the old text jumps to the right beyond the right margin. From the writer's point of view the old text is missing. You can only spot it if you move over to the right; or if you recreate the old text and then print out the file. Of course, you could move the cursor out beyond 85 columns and scan it, but normally this will not occur to you. The settings indicated will prevent all these ugly things from happening to you.

#### To Make Text Stop Jumping Around

To prevent text from jumping around when you add data to the left of where previous text has been typed, enter the LAYOUT menu and turn "Word Wrap" off. Then go to "Fixed Tab Spacing" and turn it

off. Finally, change the "Indent Style" to off. These settings will return to the default that you set in the proper LAYOUT dialogue box.

#### BOTTOM LINE

I have used Multi-Edit for about three months, and really enjoy using it. It was just what I wanted in a full-featured ASCII editor. Take it from me, it is a FANTASTIC program! Here, at last, is a fine editor that is very functional, flexible, and FUN to use!

#### SOFTWARE SOURCE

This product is available from:  
American Cybernetics, Inc.  
1830 West University, Suite 112  
Tempe, AZ 85251  
(602)968-1945

Multi-Edit is available in three flavors:  
(1) The Professional Version -about \$200,  
(2) The Standard Version (missing the Spell Check module and some other modules) - about \$100, and (3) the Shareware Version. NOTE: The shareware version is fully functional except for not having Spell Check and a few other modules.

### APPENDIX I Multi-Edit 5.0: Dynamite Quick Command List

#### CURSOR

HOME ..... To Left Margin  
END ..... To End of Line  
CTRL-C ..... To End of Block  
CTRL-HOME ..... To Top of File [TOF]  
CTRL-END ..... To Bottom of File [EOF]  
CTRL-PgUp ..... To Last Page Break  
CTRL-PgDn ..... To Next Page Break

#### DELETE

ALT-D ..... Delete from Cursor to EOL  
SHIFT-F8 ..... Delete Line  
CTRL-F10 ..... Delete Marked Block  
CTRL-W ..... Delete Current Window

#### BLOCKING

F7 ..... Mark A Block of Lines  
SHIFT-F7 ..... Mark A Block of Columns  
ALT-F7 ..... Mark A Stream of Text  
F9 ..... Copy Marked Block  
F10 ..... Move Marked Block  
CTRL-F10 ..... Delete Block  
CTRL-F9 ..... Turn Block Marking OFF  
CTRL-F3 ..... Save BLOCK to Disk

#### PICK AND PUT TEXT

[1] Block text first!  
[2] GREY+ ..... Clears Buffer, "Picks" Marked Text  
[3] CTRL-GREY+ ..... Appends Text to Buffer  
[4] GREY<ENTER> ..... "Puts" Block into Text at Cursor  
[5] CTRL-F9 ..... Turns Block Marking OFF

#### WINDOW MANIPULATING

[Window Menu]->Split .. Creates A Subsequent Window  
ALT-F1 ..... Window List  
ALT-W ..... Toggles Between Windows  
CTRL-F5 ..... Zoom Window  
CTRL-B ..... Send Cursor to Bottom of Window  
CTRL-T ..... Send Cursor to Top of Window  
CTRL-D ..... Scroll Window Down  
CTRL-U ..... Scroll Window Up  
CTRL-W ..... Delete Current Window  
SHIFT-F9 ..... Inter-window Copy Block  
SHIFT-F10 ..... Inter-window Move Block

#### SEARCH

F6 ..... Search for Text  
SHIFT-F6 ..... Search and Replace  
CTRL-F6 ..... Repeat Previous Search

#### EDITING COMMANDS

SHIFT-F3 ..... Load A File into Current Window  
ALT-J ..... Justify Paragraph  
ALT-R ..... Reformat Paragraph  
F5 ..... Make Bookmark  
SHIFT-F5 ..... Move to Bookmark  
ALT-F8 ..... Go To Line n  
F8 ..... Run a Macro  
CTRL-F7 ..... UNDO  
CTRL-DEL ..... REDO  
SHIFT-F2 ..... Stamp Date and Time  
ALT-L ..... Line Draw  
CTRL-L ..... Page Break  
F11 ..... Toggle Case  
F12 ..... Center One Line

**APPENDIX 2**  
Table of Line Numbers

| TABLE OF LINE NUMBERS |      |        |      |
|-----------------------|------|--------|------|
| HEADER                | PAGE | HEADER | PAGE |
| 1                     | 1    | 2401   | 41   |
| 61                    | 2    | 2461   | 42   |
| 121                   | 3    | 2521   | 43   |
| 181                   | 4    | 2581   | 44   |
| 241                   | 5    | 2641   | 45   |
| 301                   | 6    | 2701   | 46   |
| 361                   | 7    | 2761   | 47   |
| 421                   | 8    | 2821   | 48   |
| 481                   | 9    | 2881   | 49   |
| 541                   | 10   | 2941   | 50   |
| 601                   | 11   | 3001   | 51   |
| 661                   | 12   | 3061   | 52   |
| 721                   | 13   | 3121   | 53   |
| 781                   | 14   | 3181   | 54   |
| 841                   | 15   | 3241   | 55   |
| 901                   | 16   | 3301   | 56   |
| 961                   | 17   | 3361   | 57   |
| 1021                  | 18   | 3421   | 58   |
| 1081                  | 19   | 3481   | 59   |
| 1141                  | 20   | 3541   | 60   |
| 1201                  | 21   | 3601   | 61   |
| 1261                  | 22   | 3661   | 62   |
| 1321                  | 23   | 3721   | 63   |
| 1381                  | 24   | 3781   | 64   |
| 1441                  | 25   | 3841   | 65   |
| 1501                  | 26   | 3901   | 66   |
| 1561                  | 27   | 3961   | 67   |
| 1621                  | 28   | 4001   | 68   |
| 1681                  | 29   | 4061   | 69   |
| 1741                  | 30   | 4121   | 70   |
| 1801                  | 31   | 4181   | 71   |
| 1861                  | 32   | 4241   | 72   |
| 1921                  | 33   | 4301   | 73   |
| 1981                  | 34   | 4361   | 74   |
| 2041                  | 35   | 4421   | 75   |
| 2101                  | 36   | 4481   | 76   |
| 2161                  | 37   | 4541   | 77   |
| 2221                  | 38   | 4601   | 78   |
| 2281                  | 39   | 4661   | 79   |
| 2341                  | 40   | 4721   | 80   |

**APPENDIX 3**  
DOWNGO in Macro Source Code Format

```

$macro downgo;
{*****MULTI-EDIT MACRO*****}
Name: Downgo
Description: When creating files using "LINENOS.DOC" (i.e., no
formfeeds)it is convenient to be able to double-check header lines
and page number sequences. This macro will perform this function
by sending the cursor down 60 lines with each F8/ENTER.
by Dan Jerome/Toney Robinson
*****}
GOTO_LINE(C_LINE + 60); {Sends cursor up 60 lines}
REDRAW; {Redraws the screen}
END_MACRO;

NOTE: To convert this macro source code to a macro, send this file
to the ME directory. Then from DOS in the ME directory, type MEMAC
DOWNGO<ENTER>and it will be converted into a macro. This note should
not appear in your source code.

```

Continued from Page 6

and carries a one year warranty.

Zenith Data Systems provides the following hardware upgrades and options for the computer (it is recommended that you take the computer into an authorized dealer in order to have these installed, although an enterprising and knowledgeable user can probably perform the installations without any problem):

- 525M SCSI tape unit
- 338M SCSI hard disk drive
- 8M SIMMs
- 2M SIMMs
- 128K cache upgrade
- high-performance disk array interface
- diagnostic software
- optional manual packages
- Weitek 4167 coprocessor
- TIGA™ video card
- 1.2M, 5.25-inch floppy disk drive
- 1.4M, 3.5-inch floppy disk drive

In addition to the upgrades listed above, since the computer can use standard SCSI equipment, there's a ton of SCSI peripherals on the market that I wouldn't even begin to describe in the limited space I have here.

With a high clock speed, an 80486 microprocessor, and a plethora of upgrades currently available (and more planned for the future), the Z-486/33ET is set for a long, useful life as a network server. Or if you could afford it, it could be the ultimate stand-alone workhorse. And (since I already stepped into the world of fantasy with my last sentence) it would make a dream machine for computer adventure-game enthusiasts (imagine state-of-the-art graphics combined with blinding processor speed and massive on-line data throughput). ✨

Continued from Page 30

Query Guide adds additional fragments onto the request. When your request is complete, you tell Query Guide to execute, or process it.

Query Guide and Intelligent Assistant are exciting approaches to database management. Intelligent Assistant particularly offers immense potential for personal satisfaction. Not only can it be used for getting information out of a database, but it provides a real learning experience for the user as well. From an educational standpoint, Intelligent Assistant could easily be used in classrooms for teaching the learning process. It can give students a better understanding of the process and problems of communications. ✨



OK.. give me the story  
one more time,  
you're reading a  
borrowed REMark?



## 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?

|                            |                             | Renew                          | New                            |
|----------------------------|-----------------------------|--------------------------------|--------------------------------|
| 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<br>Surface Overseas | <input type="checkbox"/> 32.95 | <input type="checkbox"/> 37.95 |
| Daytime Phone #: ( ) _____ | Air Printed Overseas        | <input type="checkbox"/> 52.95 | <input type="checkbox"/> 57.95 |

# Sorting, Searching and KWIC Indexes

## Part 1

John Day

59, rue Nationale — Immeuble Montreal  
75013 Paris, France

When I first joined Control Data, in 1967, KWIC indexes were starting to be popular. "KWIC" means "Keyword In Context"; this sort of index works with document titles, or anything similar, as long as it gives a clear idea of the contents. It lists documents in alphabetical order of all the significant words. If you take, for example, parts of two recent issues of REMark, you could get what's in Figure 1.

Words like "and", "to", "with", and so on are in a list of "not keywords". Any word in the title that is not a keyword is used for indexing. Long titles, like "Adding Laserjet Soft Fonts...", give several entries, whilst "Powering Up" is listed under "Powering"

only. Titles like "Impressive New Features" don't really get you anywhere, unless you add a rider whilst typing them in - so this technique only works well when the titles are clear and explicit.

The first KWIC programs I wrote were on big mainframe computers, a Control Data Cyber and an IBM 4331. Making the index file requires a lot of searching and sorting, but with these big brutes you just snap your fingers and say "Sort!", and the machine sorts - megabytes and megabytes. PCs can sort up to 64 kilobytes with no problems, which would be less than 25 pages of print, so when I moved over to desktops, I left KWIC indexing behind. I still

need to program efficient searching and sorting in RAM, and how to sort a disk file without loading all of it into memory. The second article will discuss how to speed up sorting a hundredfold, and will then give an example of a sorting and searching program that will index all your favorite articles in a few minutes.

### Searching

The straight sequential search algorithm goes through a sorted array, checking each entry:

- is it still within the array, otherwise stop without finding;
- is the entry the one you're looking for, in which case stop;
- is the entry less than the one you're looking for, in which case you've overshoot so stop without finding;
- otherwise, step to the next entry.

When implementing this algorithm, you can speed it up by moving tests outside the loop. Replacing one instruction inside the loop with three instructions outside makes your program run faster whenever you loop four or more times. For a program doing intensive sorting or searching and running for five minutes or more, tightening the loops pays off in minutes saved. The obvious improvement is to quit the loop once the value you're looking for is not less than an array entry - you've either found it or overshoot - and test whether you hit or missed outside the loop. Less obvious is to cut out the test for end-of-array, which can only be done by trickery. Keep a free entry at the end of the array, and store the value that you're looking for in it. That way, the loop must stop, on an equal compare, when it gets to the end. Outside the loop, you can work out whether you got a real

|                 |                                               |        |
|-----------------|-----------------------------------------------|--------|
| to WordPerfect  | 5.X With Bitstream's Facelift Adding          | Feb 91 |
|                 | Adding Laserjet Soft Fonts to WordPerf        | Feb 91 |
| rfect 5.X With  | Bitstream's Facelift Adding Laserjet          | Feb 91 |
|                 | Database Publishing with dBASE IV             | Mar 91 |
|                 | dBASE III [Part 10]                           | Feb 91 |
|                 | dBASE III [Part 8]                            | Feb 91 |
| ublishing with  | dBASE IV Database P                           | Mar 91 |
| th Bitstream's  | Facelift Adding Laserjet Soft Fonts           | Feb 91 |
|                 | Impressive New Features                       | Mar 91 |
| Laserjet Soft   | Fonts to WordPerfect 5.X With Bitstrea        | Feb 91 |
|                 | dBASE III [Part 10]                           | Feb 91 |
|                 | dBASE III [Part 8]                            | Feb 91 |
|                 | Impressive New Features                       | Mar 91 |
| ing with dBASE  | IV Database Publish                           | Mar 91 |
|                 | Adding Laserjet Soft Fonts to WordPerfect wit | Feb 91 |
|                 | Powering Up [Volume 2]                        | Feb 91 |
|                 | Database Publishing with dBASE IV             | Mar 91 |
| adding Laserjet | Soft Fonts to WordPerfect 5.X With Bit        | Feb 91 |
|                 | The World of WP50 and Its Wonders             | Mar 91 |
| The World of    | WP50 and Its Wonders                          | Mar 91 |
| f WP50 and Its  | Wonders The World o                           | Mar 91 |
| Soft Fonts to   | WordPerfect 5.X With Bitstream's Facel        | Feb 91 |

Figure 1

hit, a dummy hit, or a miss — tests that will only be made once for the entire search operation. The following example looks for a value K in a sorted array T, with M the last entry in the array and M+1 free for the stop entry:

```
I = 0: T(M+1) = K
WHILE K < T(I)
  I = I + 1
WEND
IF K = T(I) AND K <= M THEN
  Found = -1
ELSE
  Found = 0
END IF
```

You can't get a working loop much tighter than that! Using a sorted array doesn't let you find things any faster; sorted or not, on average you'll search through half the array each time. Its only advantage is letting you call the search off once you're beyond the place the item should be — again, on average, letting you declare a miss (no match) after going through half the array only. If you have very few misses and if your hits (finding a match) are concentrated on a small proportion of the array entries, it makes sense not to sort the array, but to arrange it with the most sought-for entries at the beginning, where they can be picked up after very few compares. When you should do this is your choice, based on your knowledge of how you're using the array. With an unsorted array, the loop control statement becomes "WHILE K < T(I)". Many misses slow down the unsorted algorithm, and many hits at the beginning of the array speed it up.

Sequential searching is rather like finding a name in a 'phone book by starting on the first page, then reading all the pages in order until you come to the one you want. Another technique, binary searching, does it like you would: look in the middle, choose the half that's got the name you want, then split it in half again to narrow down to a quarter, an eighth... and in no time at all, you've found the name you're looking for. The good news is that this way, you can find any value in a 64K RAM array in sixteen

| Array size | 20 | 40 | 60 | 80  | 100 | 200 | 400 | 1000 |
|------------|----|----|----|-----|-----|-----|-----|------|
| Sequential | 32 | 57 | 81 | 106 | 130 | 253 | 498 | 1233 |
| Binary     | 36 | 42 | 46 | 50  | 52  | 60  | 67  | 76   |

Figure 2

steps or less. The bad news is that this only works for sorted tables, and takes a lot more code. For small arrays of integers binary searching can be slower than sequential techniques. An article in the September 1990 REMark by Alan Grattan gives a complete explanation of the standard binary search algorithm, with examples. I'll give the basic procedure again; you use three variables, H, L and P as indexes to the array T whilst searching for K:

- set L to the low bound, H to the high bound of the array;

- start the loop: if you're done, quit;
- calculate a compare point P half-way between L and H;
- if T(P) = K, quit;
- if T(P) > K, move L up to P and loop
- otherwise, T(P) must be < K, so move H down to P and loop.

"If you're done" is a condition which depends on how the initial loop bounds are set up. Here is a variant on the standard algorithm, designed to give better handling for the "no match" case; as before, array entries go from T(0) to T(M):

```
L = -1: H = M + 1
Found = 0
WHILE L <> H-1 AND NOT Found
  P = (L+H)\2
  IF K < T(P) THEN
    H = P
  ELSEIF K = T(P) THEN
    Found = -1
  ELSE
    L = P
  END IF
WEND
```

This code is less efficient than the standard code, but... the calculations are simpler, and it makes up for extra passes by looping faster. When there is a matching entry — a "hit" — it is as fast as the standard algorithm. The "no match" case — a "miss" — is always awkward with these routines. You should remember that sorted sequential searching is just as efficient for a hit or

| Array size  | 25         | 50         | 75          | 100         |
|-------------|------------|------------|-------------|-------------|
| Integer key | 38 [38]    | 45 [69]    | 49 [99]     | 52 [130]    |
| Decimal key | 763 [1538] | 917 [2900] | 1005 [4262] | 1077 [5625] |
| String key  | 198 [373]  | 225 [621]  | 269 [868]   | 281 [1120]  |

Figure 3

for a miss, because the code stops as soon it finds an array entry which is not greater than the search key. For a hit, it stops on the matching entry; for a miss, it stops at the point where the matching entry would have been. A binary search, on the other hand, is looking at isolated array entries. It can stop on a match, but to signal a "miss" it must continue until it finds two adjacent

array entries, of which one is less than the search key and the other is greater. The code above was optimized for this case, and runs faster than the standard code if you are searching for non-existent keys. The most useful comparison figure between binary and sequential searching is the break-even point, where both searches run equally fast. For integer keys, binary and sequential searches run at just the same speed for array sizes of:

|            | Standard | My Variant |
|------------|----------|------------|
| All hits   | 25       | 25         |
| All misses | 50       | 30         |

Shorter tables are handled faster with a sequential search, and for longer tables

you're best off with a binary search. For an array of 25 values, the standard algorithm is half as fast on misses as on hits! These relative timings should apply to any machine. The following average timing figures, in microseconds and for 100% hits (when both algorithms run at the same speed), were obtained on a Zenith 386-SX at 20 Mhz, still with integer keys; other machines will give different absolute figures, but the ratios should stand (Figure 2).

As you can see, for arrays up to about 50 entries binary searching isn't that much faster, and it may not be worth the extra code — particularly if you have a low hit rate. I have also calculated the array sizes where a binary search runs at 90%, 80% and 50% of the sequential search time — this one is for the worst case, 100% misses, and using the Basic math emulator package:

|             | [110] | 90 | 80 | 50 |
|-------------|-------|----|----|----|
| Integer key | 19    | 36 | 45 | 85 |
| Decimal key | -     | 6  | 10 | 26 |
| String key  | -     | -  | -  | 30 |

The first column indicates that sequential searching is 10% faster than binary searching with 19 integer entries in the array. For decimal or string keys, sequential searching is never much faster.

The last figures you need to choose your access method are the absolute timings, in microseconds (sequential search

in square brackets). I was still using the Basic math emulator package (See Figure 3).

The obvious first lesson is "don't search decimal arrays unless you have a math coprocessor". Decimal arrays take 20 times as long to search, because of emulator overhead. If your key values exceed 32, 767, use long integers if your compiler supports them, or use two arrays, TH for the high-order two bytes of your value and TL for the low-order two bytes:

```
IF K < TH(P) THEN
  H = P
ELSEIF K = TH(P) THEN
  IF K < TL(P) THEN
    H = P
  ELSEIF K = TL(P) THEN
    Found = -1
  ELSE
    L = P
  END IF
ELSE
  L = P
END IF
```

The loop is now about twice as long as for a single array, but will still run much faster than a decimal compare. String searches run slower than integer searches, but not that much. For two ASCII characters, you will get faster results with two



integer arrays containing the decimal ASCII codes instead of the characters. For three ASCII characters, you could improve results slightly by running a triple search on three decimal arrays. For longer character strings, stick to ordinary searching - it's still four times faster than decimal searches. If you use multiple arrays for the search keys, it's worth while declaring a user function to compare two keys across several arrays - see FNKeyComp in the sample program, which compares across four arrays and returns +1 for "greater", -1 for "less", and 0 for "equals". FNKeyComp supposes that both the keys are in the same array, but you can adapt it to other needs.

The second lesson is that for integer arrays less than 35 entries, binary searching is not worth the bother. For a program which does a fair bit of searching, use binary algorithms from 45 (integer) or 10 (decimal) entries; binary searching seems always faster when using strings. If you are doing occasional searches only, use binary searches from 85 (integer), 25 (decimal) or 30 (string) entries; you will halve the search time, but the overall effect on your program won't be that great.

The final lesson to remember is that just because it's better, it's not necessarily faster. Most improvements to algorithms require extra code, and increased efficiency will be offset by increased overhead. When in doubt, try it. The Basic TIMER function lets you check how long code takes to run, and was used to give the above figures - but beware! The timer ticks every 55 msec, so to get 1% accuracy you must time over at least 10 seconds. The algorithms above were timed for 10,000 to 100,000 searches, using various search keys on a standard array, to get usable results.

All the timings given above use integers for indexing the arrays; you should always use them wherever you can in loops, as integer arithmetic is always faster than decimals. However, if you're running binary searches in integer arrays of over 16 000 entries, you will have to modify the calculation step " $P = (L+H)\backslash 2$ ". For a value near the end of the array, L and H will both point to entries near the upper limit, and the intermediate value in the operation will overflow. You may get an error message, but the search is more likely to just hang up. Convert the indexes to decimal or long integers for the calculation, then convert the result back to an integer.

### Sorting

The previous examples require sorted arrays - which don't grow on trees. If you want to search efficiently, you must sort first. Don't try and write a superb general purpose disk sort program, you'll be at it for some years. Disk sorting is for experts, and uses special techniques such as the "tournament" algorithm, which runs compari-

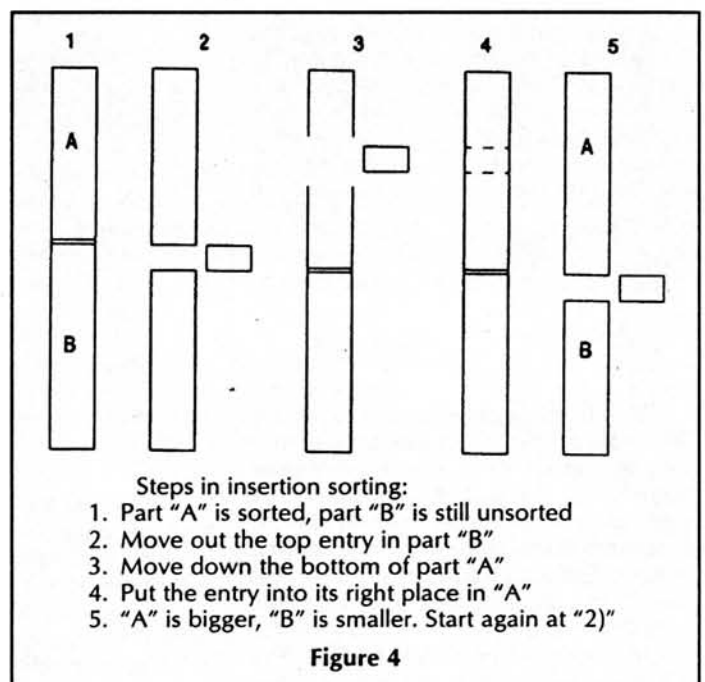
sons like a tennis tournament with finals, semi-finals, quarter finals and so on. Stick to core sorting, where Basic will sort up to 64 Kb in less time than it takes to drink a coffee, using fairly simple algorithms. And be careful with your code (unless you've got a huge coffee cup). Sorting is very compute-intensive. Simple sorting is an  $N^2$  operation, which means that unless you're downright careless, sort time is multiplied by 4 when you double the size of the array. There is a lower theoretical limit for sorting of  $N \log N$ , which means that when you double the size of the array, the sort time is multiplied by 1.386. Unless you're a genius, your performance will lie somewhere between these two; a factor of 2.83 is a good target. A good, useful sort technique is insertion sorting. It is up to three times faster than the well-known bubble sort, and takes less than twice as much code - 17 lines of code against 11. For small tables, you can even code it in fewer lines than a bubble sort. Let's explain it by imagining a left-to-right sort of a hand of cards. For a bubble sort, you look at the first two cards, and if they are out of order you exchange them. Then you look at cards two and three, and do the same... then cards three and four, and you go on like this to the last two cards in your hand. It's fairly obvious that once you reach the highest card, it will always be exchanged with the card on its right, and will end up to the right of your hand. The rest of the hand will be better sorted than before, but won't be in order yet. You then repeat the process, up to the last card you swapped before; it can be proved that the hand is always correctly sorted to the right of the last swap. You go on running through the hand like this until you make one pass without swapping any cards; the hand is now sorted. This is an easy algorithm to code on a computer, always examining two adjacent values; this must explain its popularity.

For insertion sorting of your hand, you withdraw the second card and place it correctly with respect to the first; you then withdraw the third card, and place it correctly with respect to the first two; then place the fourth card relative to the first three, and so on to the end of your hand. After one pass, your hand is

sorted. Think about the place in your hand where you have withdrawn a card. To the left, your hand is sorted; to the right, it is still untouched. As you proceed, the sorted part gets longer and the unsorted part gets shorter. To insert the card you've withdrawn, you need an extra place in your hand - but you've got it, because withdrawing the card frees up one place to let you move the high part of the sorted hand to the right the one place you need. Here's the code to sort array T, containing entries from 0 to TMax:

```
FOR I = 1 TO TMax
  Temp = T(I): L = -1: H = TMax + 1
  WHILE L <> H - 1
    P = (L+H)\2
    IF Temp <= T(P) THEN
      H = P
    ELSE
      L = P
    END IF
  WEND
  FOR J = I - 1 TO H STEP -1
    T(J+1) = T(J)
  NEXT J
  T(H) = Temp
NEXT I
```

You will recognize a simplified version of the binary search. The array is always sorted up to the entry before I - we start with I = 1, when there is only T(0) to the left of I, and an array with only one entry is sorted by definition. "Temp = T(I)" withdraws the next entry from the unsorted part of the array, and frees up a slot. The binary search runs until H points to the first entry greater than Temp. A loop then moves everything from this point to the end of the sorted part down one place - using the slot we just freed up. Finally, Temp is moved into place between the section of the array that moved and the section that didn't. If there are duplicate keys in the array, they will be kept in the same order, since the



search algorithm skips over "equals" to the first "greater than". Sometimes this is important, and sometimes it isn't - so let's do it anyway.

A diagram will help you to see how each entry is moved out, shifted up to its proper place and put back in again (see Figure 4).

If you have several satellite arrays linked to the key array - a record structure - then you must move all the *I*th array entries to temporary values, shift all the arrays in lockstep, and put all the temporary values into the *H*th entry in their corresponding array.

Note that whereas bubble sorting requires you to have the whole array available before starting, insertion sorting only requires the sorted part plus one record. This makes insertion sorting friendly to use when you are reading and sorting from a disk file to memory. Instead of reading the whole file into memory and sorting it, you can insert each record into the right place as you read it. The "B" part of the table doesn't exist; you take each new record,

match it to the proper place in the "A" part of the table, and slot it in. The "A" part gets bigger and bigger as you add the records, until it reaches full size at the end of the sort. This method will be used in both the examples given in the sample program in the next article. The binary search algorithm given previously works well for this, as it runs properly on an empty array, returning the first entry as the sort insertion point; this gives you a self-starting sort. Other binary search algorithms may require you to store at least one value into the array before beginning to sort.

For a short integer array of, say, less than about 50 values, you can replace the binary search with a straight search. If you combine searching with moving, by searching backwards from the end of the array instead of forwards from beginning, you get about the shortest possible serious sort program:

```
FOR I = 1 TO TMax
  Temp = T(I): J = I-1
  WHILE J > 0 AND T(J) > Temp
    T(J+1) = T(J)
    J = J-1
  WEND
  T(J+1) = Temp
NEXT I
```

This code frees up a slot as before, then moves down 1 all values greater than the one to insert. As soon as it finds a value that is not greater (or beginning-of-array), it stops and puts the value to insert into the last slot freed up. Note: with some versions of Basic (QuickBasic in particular), you will have to sort the array from 1 to *n*, and not from 0 to *n*-1. Loop 1 from 2 to the end of the array, with "WHILE J > 0 AND...". This is because when J = -1, all the Basic Com-

pilers I've tried will find the first expression false and not evaluate further, or may (QB versions 4.5 and 7.0 at least) evaluate both halves and bomb out on T(-1). Sorting the array from T(1) gives a dummy value in T(0) which can be evaluated (uselessly) by QuickBasic. Whichever way you do it, for an integer array with less than 30 values this method is unbeatable. For any size array, it is shorter and faster than a bubble sort.

Address table sorting avoids moving keys and satellite record entries around in memory - at the price of an auxiliary integer array. Let's call this array "AT"; it will be the same size as the array we are sorting. At the start, each entry in AT will point to the corresponding entry in the table to be

To print T in its sorted order, you must use the values in TA to index:

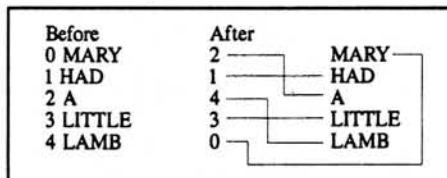
```
FOR I = 0 TO TMax
  PRINT T(TA(I))
NEXT I
```

Seems a lot of trouble for nothing? If you're just sorting a single array of numbers, yes... but now look at the timing figures for a few arrays, in seconds this time. For the direct sort of a string array, the instruction "T\$(J+1) = T\$(J)" was replaced with "SWAPT\$(J+1), T\$(J)"; because of the way Basic handles strings, this change gives a big reduction in execution time. The only arrays that will go up to 32,000 entries are integers, and I had to use long values for indexing. I ran out of memory for strings

|                              | 250 | 500 | 1000 | 2000 | 4000 | 8000 | 16000 | 32000 |
|------------------------------|-----|-----|------|------|------|------|-------|-------|
| <b>Direct array sorting</b>  |     |     |      |      |      |      |       |       |
| Integer keys                 | ,1  | ,3  | 1    | 4    | 15   | 59   | 231   | 940   |
| Decimal keys                 | ,3  | ,8  | 2,3  | 7,4  | 25,6 | 92,5 | 363   |       |
| String keys                  | ,2  | ,7  | 2,6  | 10,1 | 41,5 |      |       |       |
| <b>Address table sorting</b> |     |     |      |      |      |      |       |       |
| Decimal keys                 | ,3  | ,7  | 1,9  | 5,8  | 19,4 | 68,5 | 253   |       |
| String keys                  | ,1  | ,3  | 1,1  | 4,1  | 15,4 |      |       |       |

Figure 5

sorted: it will just contain "0, 1, 2, 3..." and so on. It doesn't have to be initialized first, as we can build it as we go. Instead of moving the main table entries around in memory, we just move the address table entries. Example:



The address table entries mean "read the main table in the order 2, 1, 4, 3 and 0... that is, A HAD LAMB LITTLE MARY. Here's the code:

```
AT(0) = 0
FOR I = 1 TO TMax
  L = -1: H = TMax + 1
  WHILE L <> H-1
    P = (L+H)\2
    IF T(I) <= T(AT(P)) THEN
      H = P
    ELSE
      L = P
    END IF
  WEND
  FOR J = I-1 TO H STEP -1
    AT(J+1) = AT(J)
  NEXT J
  AT(H) = I
NEXT I
```

Note that the contents of the array T (and any other arrays in the record structure) are no longer moved around; at the end of the sort, T is in exactly the same order as when you started. Only the contents of TA are moved, and the sorted part of T is always indexed indirectly using TA.

when trying to sort over 4,000 entries (see Figure 5).

For single integer keys, address table sorting (not given) is no faster than straight sorting. For other types of keys, however, it can run more than twice as fast. Address table sorting only moves entries within one integer array in core, whereas other sorts have to move decimal values and - worse - strings. If you have your key spread over several arrays, or if you have extra information in satellite arrays to move in lockstep, address table sorting becomes more and more interesting.

An extension of the address table is the tag sort. If you want to sort a random disk file, you can manage it without moving anything around on the disk. Read the keys into a memory array, putting each key in the array entry which corresponds to its record number - with Basic, this means starting your array with record 1 in T(1). Run an address sort of the memory table. The final address table lets you read the array of keys in sorted order. Since the keys haven't moved and are still in the same order as the records in the disk file, you can read your file in sorted order by using the address table as a list of record numbers! Depending on what you are doing, you can either use the file as it is, or copy the file to a new disk file, following the sorted sequence. The new file will be sorted. I'll be showing how to use this technique in the sample program, where the sort module will sort pointers to the title file and not the magazine titles themselves. During printing, the titles can be read in from disk as fast as the printer can take them.

## Collating Sequences

The sort sequence for integer or decimal keys is straightforward. Sorting is by increasing arithmetic value unless you invert all the tests, replacing "less" with "greater", in which case your array will be sorted in decreasing order.

With strings, life is less simple. Direct string comparisons use the ASCII values. Capital letters are coded 65 through 90, while lower case letters are coded 97 through 122; so any capital letter will sort before any lowercase letter. In 1967 this wasn't a problem, there just weren't any lowercase letters, because the 64 character BCD code covered capitals, digits, and a few punctuation signs only. These days, we have to be more thorough. Unless you tell it otherwise, Basic will sort "Getting Started" before "Getting into Word Processing", because "S" is coded 83 and "i" is 105. You want both "S" and "s" in 19th position, and "l" and "i" in 9th place. In the same vein, "dBASE" ("d" = 100) looks better before "Z 100" ("Z" = 90). For us Europeans, it's even worse: in 9th place, we need

l i i i i i

The simplest way to handle this is to build a new key, replacing each ASCII character with its position in our own sort sequence. "A", "a" and nine other variations from the international set all become 1, "B" and "b" will be 2, and so on through the alphabet. Now is the time to say "Aha!". We've already seen that sorting strings is slow. Here, we're replacing our string with a series of integers, and we can sort them really fast. All we have to do is arrange to sort on a fixed number of characters, instead of the indefinite length of a Basic string variable. I chose 20, for reasons which will shortly become apparent. Not many keywords will go over 20 letters, and the extra will let us do a short sort on the start of the text following the key: sort "dBASE IV" after "dBASE III", for example, although in both cases the keyword is "dBASE". Now, the extra benefit: this gets us off the hook with Basic string capacity. You are limited to 64K of string data if your compiler supports far strings like PDS (stored outside the Basic data segment), and a lot less if it doesn't, like QuickBasic. If you use strings of about 20 characters, that gives you a sort capacity of just 3,200 entries. 10 digits plus 26 letters (36 distinct values to encode) fit very comfortably in 6 bits, which can encode values from 0 to 63 and you can get 5 6-bit values (30 bits) into a 32-bit long integer whilst keeping well clear of the sign bit: one long-integer key can be used for five string characters. Four long integer arrays (4 x 5 = 20) at their maximum size of 16,384 values each will take 256 Kb of DOS RAM, which is usually available. If

you're using QuickBasic 3 or previous, you won't have long integers, so you can:

- put two 6-bit values in a short integer
- put five values spread across two integers (tricky, but it works; see the comments on how to do it after the discussion of sample program)
- drop coding for numerics, and just code for 26 letters in 5 bits, three letters per integer.

Tests I ran on the titles used for RE-Mark articles show that it's better to in-

clude the ten digits in a collating scheme, to handle keywords like "286LP" or "386SL", which otherwise would sort to "LP" and

"SL". On other types of data, you may get away with not including them. The integer keys should be built in steps. First, you must convert each character to its ASCII equivalent, then use this to look up a table. The CSeq table in the sample program covers the 256 characters in the international character set; if you only use 7-bit ASCII, the first 128 entries are enough. Digits 0 through 9 are equivalenced to "1" through "10", and the alphabet goes from "11" to "36". Anything to be ignored

```
K1&(I) = 16777216*CSeq(ASC(LEFT$(T$(I),1))) + 262144*CSeq(ASC(MID$(T$(I),2,1))) + 4096*CSeq(ASC(MID$(T$(I),3,1))) + 64*CSeq(ASC(MID$(T$(I),4,1))) + CSeq(ASC(MID$(T$(I),5,1)))
```

Figure 6

clude the ten digits in a collating scheme, to handle keywords like "286LP" or "386SL", which otherwise would sort to "LP" and

is coded "0"; this way, "I.B.M." can be sequenced as "19 12 23", not taking the

Continued on Page 48

## Listing 1

```
DEF FNKeyComp (A, B)
FNKeyComp = +1
IF K1&(A) > K1&(B) THEN
EXIT DEF ' ... it is
ELSEIF K1&(A) = K1&(B) THEN
IF K2&(A) > K2&(B) THEN
EXIT DEF ' first array is a tie, so
' ... let second decide
ELSEIF K2&(A) = K2&(B) THEN
IF K3&(A) > K3&(B) THEN
EXIT DEF
ELSEIF K3&(A) = K3&(B) THEN
IF K4&(A) > K4&(B) THEN
EXIT DEF ' A > B on last compare
ELSEIF A4& = B4& THEN
FNKeyComp = 0 ' all four tests compare equal
EXIT DEF
ENDIF
ENDIF
ENDIF
ENDIF
FNKeyComp = -1 ' one test was not greater, not
equal
END DEF

DIM CSeq(255) ' collating sequence table
DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -2, 0, 0, -2, 0, 0
DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
DATA -1, -1, 0, 0, 0, 0, 0, -1, -1, -1, 0, 0, -1, 0, 0, 0
'
' ( )
DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -1, -1, 0, 0, 0, -1
'
' : ; ?
DATA 0, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
' a c e i n o
DATA 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, -4, 0, -3, 0, 0
'
' u y
DATA 0, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25
DATA 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, -1, 0, -1, 0, 0
DATA 13, 31, 15, 11, 11, 11, 11, 13, 15, 15, 15, 19, 19, 19, 11, 11
DATA 15, 11, 11, 25, 25, 25, 31, 31, 35, 25, 15, 0, 0, 0, 0, 0
DATA 11, 19, 25, 31, 24, 24, 11, 25, 0, 0, 0, 0, 0, 0, 0, 0
DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
DATA 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
FOR I = 0 TO 255
READ CSeq(I) ' load collating sequence from
' data statements
NEXT I
```



# Words of a Feather



James Hoffer  
1509 W. Glenlord Road  
St. Joseph, MI 49085

"Les spécialités françaises sont vraiment délicieuses." "Möchten Sie mit mir einkaufen gehen?" "¿Dónde se encuentra la estación?"

Neat, huh? When words of a feather need to flock together, how do you make accented letters, upside-down question marks, and other special effects on your PC? If you do a bit of writing in foreign languages as I do, you will need to know.

There are four basic ways to accomplish these special effects, and some of them include not only accented letters, but symbols such as one-half (½), bullet (·) and section (§), and are limited only by the type of printer you are using. They are as follows, and should work with most word processors (if you are using WordPerfect 5.1 as I am, make sure that your GRAPHICS QUALITY under SHIFT-F7 is set at least to "medium"):

**ALT keys.** Hold down the ALT key and type one to three numbers on the numeric keypad. This is the easiest method, and is described in the DOS manual appendix. The complete keyboard for the default English code-page table 437 is shown on page 632 of the DOS 5.0 manual. (Page A.10 of the Zenith Data Systems MS-DOS Version 5.0 REference. -Ed.) Depending on your setup, some keys may appear on the screen as little squares (·) but will print correctly on paper.

**Character sets.** WordPerfect 5.1 allows the user to type a wide variety of letters, signs and graphics characters through the "compose" function, as described on pages 80-83 of the manual, and Appendix P, including Japanese! I understand that WordStar has a similar function. By typing CTRL-V and a series of two characters or numbers, the new character is formed.

**Overstrike.** This is the least desirable method, because some characters do not

come out as nicely formed, but may be a last resort if you cannot make the particular letter any other way. A slashed zero is a good example. In WordPerfect, the commands would be as follows: Format (Shift-F8), 4 for Other, 5 for Overstrike, 1 for Create, the two characters you desire to merge, Enter, and F7 for Exit. Whew! But the result is a nice ø. Of course, you could design a macro to accomplish it more quickly.

**Setting up a new code page.** For some, this may be the most satisfactory answer, as it causes accent marks to automatically overstrike in combination with certain letters, depending on the code page number. Available, for example, are acute (´), grave (`), umlaut (¨), tilde (~), circumflex (^) and cedilla (¸). For most writers of foreign languages, code page 850 will be the obvious choice. The procedure is a bit complicated, and is described in chapter 13 of the DOS 5.0 manual (and ZDS MS-DOS 5.0 Reference. -Ed.), so I have simplified it for you here.

First you will need to add the following lines to your "config.sys" file:

```
COUNTRY=055,,C:\DOS\COUNTRY.SYS
[ I use 055 for Brazil ]
DEVICE=C:\DOS\DISPLAY.SYS CON=(EGA,437,1)
[ for your monitor ]
DEVICE=C:\DOS\PRINTER.SYS LPT1=(4208,437,1)
[ for your printer ]
```

Then, write a small batch file to activate these functions. Mine, which I call BR.BAT, looks like this:

```
NLSFUNC
[ to activate code page switching for
all devices ]
MODE CON CP PREP=((850)C:\DOS\EGA.CPI)
[ for your monitor ]
MODE LPT1 CP PREP=((850)C:\DOS\4208.CPI)
[ for your printer ]
KEYB BR,,C:\DOS\KEYBOARD.SYS
[ the BR selects Brazilian ]
CHCP 850
[ to turn on code page 850 ]
```

In all of the above, the C:\DOS portion of each command may be different in your case, depending on the subdirectory where you have placed your DOS files.

The commands you have added to CONFIG.SYS will in no way affect your computer until you activate them by running the batch file. So when you wish to type in a foreign language using this method, run the batch file *with your printer on* (it will delightfully squeal as you tickle it with this batch file), and then load your word processor. Now watch as you make accented letters, first by typing the accent mark and then the letter. For example, the c-cedilla is formed by first typing a single apostrophe (´) and then the letter (c) to make ç.

This function can be disabled at any time by pressing CTRL+ALT+F1 and reenabled by pressing CTRL+ALT+F2.

Writers of Greek and Hebrew may be interested to know about a WordPerfect add-on known as ScriptureFonts, published by Zondervan and available from most Christian book stores. Beautifully accented Greek and pointed Hebrew are easily produced by this excellent program:

Οὗτος γάρ ἠγάπησεν ὁ θεὸς τὸν κόσμον  
:בְּרִאשִׁית בָּרָא אֱלֹהִים אֶת הַשָּׁמַיִם וְאֶת הָאָרֶץ.

Of course, these same characters can be printed using the more cumbersome CTRL-V method. In order to see just what your configuration is capable of doing, retrieve the CHARACTER.DOC file that comes with your WordPerfect, and print it out, especially pages 24-29 that contain the Greek and Hebrew.

You may have to experiment a bit to get all these things to work, but you will end up with beautiful, professional results. Good luck! ✨

# Textbooks for the 1990's



Frederick O. Smetana  
5425 Parkwood Drive  
Raleigh, NC 27612

When you've taught at the University level for 40 years as I have you begin to think that perhaps you have acquired some unique knowledge or some unique approach to your subject that might be worth passing on to future generations of students in the form of a textbook. In some fields writing textbooks is regarded as part of the professors's professional duties. In Engineering, as the Dean has so often reminded me, you do it at your own risk. If it's successful, you get to keep the royalties. If it's not, you've put in a lot of time boosting your ego and may have actually hurt your career unless, of course, you become prolific in the book writing business. The writing of a book has no effect on the amount of research dollars you're expected to bring in or the number of graduate students you are supposed to support and supervise. Despite all else that is said, these are really the criteria on which your salary increases and promotions are based.

In the Spring of 1985 I had a visit from an editor from McGraw-Hill who mentioned they were looking for a replacement for a comprehensive dynamics and automatic controls text. I thought about it and liked the idea with a few modifications. Although McGraw-Hill eventually turned down my outline and Preface, I pursued the project with other publishers. One was interested in getting into this area of the engineering textbook market and encouraged me to proceed.

I began in the usual way for technical publication of that time — writing out the manuscript in longhand and making quick sketches of the figures. My department helped by preparing a typescript from my

manuscript using an old A.B. Dick word processor. That machine stored completed results on 8" single-sided, 32 sector, hard-sectored CPM diskettes.

Late in 1986 sample chapters were ready for review. The reviews came back with a broad spectrum of opinions. No two criticized the same thing. I had expected as much since there is no universal opinion on what should be in the course(s). Another evidence of that diversity is that there are at least two dozen competing books. The editor I had been working with then told me that his company had decided to trim the number of titles they were going to

publish and mine would not be one. He personally liked mine and would like to publish it through a small desktop publishing company run by his wife out of their house. He sent me a publishing contract in March 1987 and I thought that once I completed the typescript all I would have to do is proofread the galleys and/or page proofs.

I completed the writing task around the end of the year. When the editor's wife began to look seriously at what had to be done she said there was no way they could handle the typesetting of that number of mathematical expressions. It was either

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{k_1}{m_1} & \frac{k_1}{m_1} & 0 & -\frac{c_1}{m_1} & \frac{c_1}{m_1} & 0 \\ \frac{k_1}{m_2} & -\left(\frac{k_1}{m_1} + \frac{k_2}{m_2}\right) & \frac{k_2}{m_2} & -\frac{c_1}{m_1} & -\left(\frac{c_1}{m_1} + \frac{c_2}{m_2}\right) & \frac{c_2}{m_2} \\ 0 & \frac{k_2}{m_3} & -\frac{k_2}{m_3} & 0 & \frac{c_2}{m_3} & -\frac{c_2}{m_3} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \frac{1}{m_1} [F] \quad (52)$$

$y = x_3$

Figure 1

Figure 2

```

\documentstyle[11pt]{article}
\topmargin 0.25in
\oddsidemargin 1.0in
\textwidth 6.75in
\textheight 6.75in
\parindent 2em
\newcommand{\sfrac}[2]{\displaystyle{\frac{#1}{#2}}}
\def\insertplot#1#2#3{\par
  \hbox{%
    \hskip #3
    \vbox to #2{\vfill
      \special{ps: plotfile #1}
    }}
}
\newcommand{\cf}{\it cf.}
\newlength{\ssp}
\setlength{\ssp}{-1.6ex}
\renewcommand{\baselinestretch}{1.0}
\large
\normalsize

\begin{document}
\small
$$
\left[ \begin{matrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \\ \dot{x}_6 \end{matrix} \right]
= \left[ \begin{matrix} 0 & 0 & 0 & -1 & 0 & 0 \\ -k_1 & -k_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ -k_1 & -k_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ -k_1 & -k_1 & 0 & 0 & 0 & 0 \end{matrix} \right]
\begin{matrix} \frac{k_1}{c_1} \\ \frac{k_1}{c_1} \\ \frac{k_2}{c_2} \\ \frac{k_2}{c_2} \\ \frac{k_2}{c_2} \\ \frac{k_2}{c_2} \end{matrix}
\left[ \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} \right]

```

```

-----\cr
x_4\cr
-----\cr
-----\cr
-----\cr
x_5\cr
-----\cr
-----\cr
-----\cr
x_6\cr
-----\cr
+ \left[ \begin{matrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{matrix} \right]
\left[ \begin{matrix} \frac{1}{m_1} \\ \frac{1}{m_1} \\ \frac{1}{m_1} \\ \frac{1}{m_1} \\ \frac{1}{m_1} \\ \frac{1}{m_1} \end{matrix} \right]
\left[ \begin{matrix} F \\ F \end{matrix} \right]
\eqno{(52)}
$$
\vspace*{0.15in}
y ----- x_3
$$
\vfill
\end{document}

```

translate the text into a form they could work with on a Macintosh or rekey it. We tried all over the country to find someone who could read and translate those old A.B. Dick diskettes without success. That meant the entire text had to be rekeyed, not the simplest of jobs when one is talking about a 900-page book. I even tried scanning the typescript and using OCR software. This worked, sort of, but the number of corrections that had to be made took about the same time to fix as retyping. Then there was the task of deciding what form to do the book in. The editor and I considered Ventura Publisher and Pagemaker but finally settled on LaTeX because of its unexcelled treatment of mathematical expressions.

TeX is a typesetting program written by Dr. Donald Knuth of Stanford University with the support of the American Mathematical Society. It is available in commercial versions for the PC and MAC and in public domain versions for most minicomputers and mainframes. LaTeX is a set of macros for TeX which were created by Leslie LaPorte to ease the use of TeX. LaTeX uses only ASCII characters for all text and formatting instructions. The backslash tells the program that what follows is a formatting instruction or symbol. The \$ sign is used to signify a mathematical expression which is usually written in italics. The .TEX file is processed by LaTeX into a binary file



## Chapter 6

# INTRODUCTION TO STATE SPACE ANALYSIS OF DYNAMIC SYSTEMS

### 6.1 Introduction

In order to analyze the behavior of complex systems (machinery, structures, or apparatus with many degrees of freedom), including those which contain time delays and non-linear elements, one is virtually forced to seek methods which rely upon the speed and power of the computer. Fortunately, one of the problems of modern numerical analysis which has been attacked most successfully is the forward integration of very, very large systems of simultaneous, first-order differential equations. It is not surprising, therefore, that systems analysis techniques tailored to take advantage of such integration techniques have become quite popular. In these techniques, the system under analysis is considered to be modeled at any point in time by the current values of a sufficiently large number of "states". The change in each state from its value at one point in time to its value at some later time is obtained from the solution of a first-order differential equation of the form

$$\dot{x} = \{A\}x + \{B\}u$$

where  $\dot{x}$  is the rate of change of a particular state,  $x$  is a vector of all the states used to describe the system,  $A$  is a row matrix which provides the proportionality between each state in the system and the rate of change of the particular state,  $B$  is another row matrix which gives the proportionality between the rate of change of a particular state and all the control inputs to the system, and  $u$  is the vector of all control inputs. The control inputs are

Figure 3

with the extension .DVI. DVI stands for device independent. DVI files are read by printer or screen drivers and rendered at the resolution of that particular device. For our work the printer driver most often creates a 300 dpi postscript file. Professor Knuth designed special fonts for TeX called computer modern. A set usually consists of 75 or so bit maps defining the fonts at various point sizes in increments varying by a factor of 1.2. The computer modern fonts do an especially good job at kerning, ligatures, and other special typesetting effects. Because they are different from those normally resident on laser printers they must be downloaded at the beginning of the print session. The TeX program is sufficiently flexible that standard laser printer fonts can be substituted for some computer modern fonts, if desired.

LaTeX has a fairly steep learning curve and one must use it fairly frequently in order to be able to remember the commands needed to create specific mathematical symbols. The program was such a

hurdle that I could no longer ask the departmental secretaries to do the "typing" for me. I had to learn it myself. I also hired a couple of students to do the longest chapter in the book for me. Although this rekeying in LaTeX took me almost two years, I was pleased with the result. Figures 1 and 2 show typical pages with the .TEX original and the postscript result for two equations. Figures 3 and 4 show the .TEX file and results for typical text.

The editor also wanted all the figures in machine-readable form. About 150 figures represented the results of computer computations and had to be drawn accurately. Another 325 were line drawings. Originally he had offered to provide the line art. (Eventually, he found even that to be too expensive in light of the anticipated annual sales.) All the figures would be in .EPS format

so that they could either be read into the proper place on the page by the printer driver or printed as separate postscript files and pasted in the hard copy originals for the offset masters. TeX, for example, has a special command that permits one to write a macro for importing graphics files. Not all versions of LaTeX employ such a macro but the one from ArborText, Inc. seems to work well. The ArborText postscript driver also allows one to rotate a file from portrait to landscape with a simple command line instruction.

For the figures which required accurately-drawn curves we chose to do them on a full page using a program called ProPlot from a small firm in the Palo Alto area called Cogent Software, Inc. This program automatically scales the data (up to 1000 points per curve and several curves per picture) to the specified plot dimension, allows the user to place legends at desired locations, uses selectable symbols for the curves, and easily places figure titles and axis names on the figure. The instructions to the figure-creating program are contained in a small file with a .PLT extension while the data are read from one or more files with .DAT extensions. Figure 5 shows such a command file and Figure 6 shows the results.

For the line art one could choose sketching programs like AutoSketch but I found that too difficult to use in the manner I desired. The graphics artist at Engineering

I desired. The graphics artist at Engineering

```
\begin{document}

\chapter{INTRODUCTION TO STATE SPACE ANALYSIS OF DYNAMIC SYSTEMS}
\section{Introduction}
\markboth{Chapter 6}{Introduction}

In order to analyze the behavior of complex systems (machinery, structures,
or apparatus with many degrees of freedom), including those which contain
time delays and non-linear elements, one is virtually forced to seek methods
which rely upon the speed and power of the computer. Fortunately, one of the
problems of modern numerical analysis which has been attacked most successfully
is the forward integration of very, very large systems of simultaneous,
first-order differential equations. It is not surprising, therefore, that
systems analysis techniques tailored to take advantage of such integration
techniques have become quite popular. In these techniques, the system under
analysis is considered to be modeled at any point in time by the current
values of a sufficiently large number of "states". The change in each state
from its value at one point in time to its value at some later time is
obtained from the solution of a first-order differential equation of the form
$$
\dot{x} = \{bf \left\{ A \right\}\} \vec{x} + \{bf \left\{ B \right\}\} \vec{u}
$$
where  $\dot{x}$  is the rate of change of a particular state,  $\vec{x}$  is a
vector of all the states used to describe the system,  $\{bf A\}$  is a row matrix
which provides the proportionality between each state in the system and the
rate of change of the particular state,  $\{bf B\}$  is another row matrix which
gives the proportionality between the rate of change of a particular state
and all the control inputs to the system, and  $\vec{u}$  is the vector of all
```

Figure 4

```
(Pitch.PLT)
title bottom 'TIME, seconds'
(CURVE 1 :)
title left 'AMPLITUDE'
insert pitch.dat
```

Figure 5

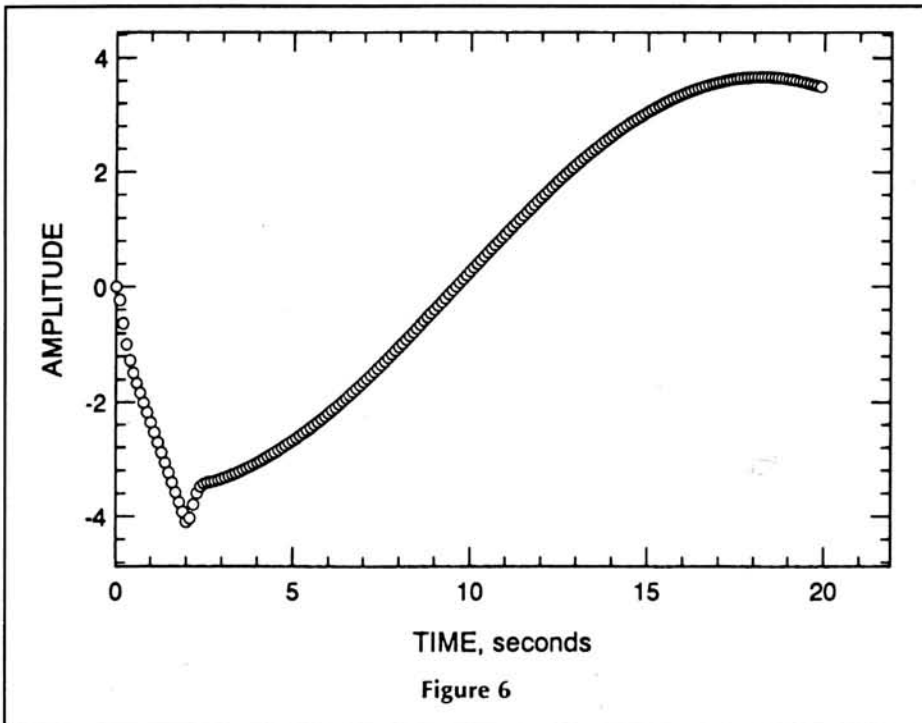


Figure 6

Publications prefers old-fashioned pen and ink. The principal disadvantage of that method is in getting machine-readable versions of his work. Scanned copy looked too rough. I decided to write most line art files directly in postscript. For straight lines and alpha characters this is relatively easy. However, I have discovered that the arrows in the symbol font vary in shape and position from implementation to implementation making it very time-consuming to place an arrow head correctly at the end of a line. I did not have any information, for example, as to where with respect to the reference the shaft of the arrow begins or how long it is. This made it a cut and try operation where often differences as small as 0.5 point were noticeable. Figure 7 shows the

postscript instructions and Figure 8 shows the result as rendered on a dot matrix printer. One of the most useful features of this method of graphics generation is that the file is easily scaled—it takes

```

1.0 0.8 scale
200 425 moveto
200 300 lineto stroke
175 325 moveto
425 325 lineto stroke
200 325 moveto
275 400 lineto stroke
275 400 moveto
350 325 lineto stroke
\Helvetica findfont 24 scalefont setfont
150 385 moveto
(f(t)) show
193 275 moveto
(0) show
272 275 moveto
(T) show
338 275 moveto
(2T) show
150 240 moveto
(FIGURE 5-47 Time History of) show
150 215 moveto
(output of Slexer Data Hold) show
showpage
^D

```

Figure 7

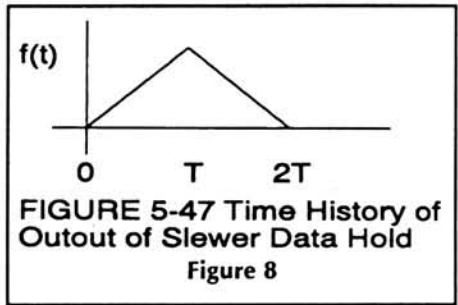


FIGURE 5-47 Time History of Output of Slexer Data Hold  
Figure 8

erated by a laser so curves are a little less smooth than can be printed with a laser printer. GoScript has a preview capability (using an EGA or VGA display) which is much faster but coarser than the printed version. This feature is very helpful when debugging the files you have written. The low-priced version of GoScript comes with 13 scalable fonts. I have purchased an additional 16 but there are literally dozens of other fonts available. In fact, I don't have all the standard fonts resident on a LaserWriter.

About a third of the book is now complete with the line art still to be done for the other two thirds. Text and equations for the entire book, however, exist as postscript files. I expect to use the completed

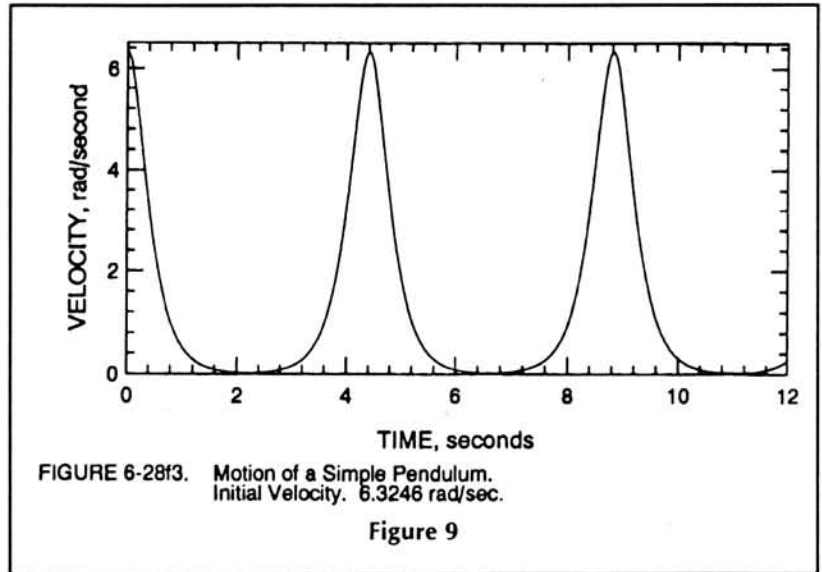


FIGURE 6-2813. Motion of a Simple Pendulum. Initial Velocity. 6.3246 rad/sec.

Figure 9

only one line — to fit the space available. The two axes may even be scaled differently.

Since I do not have a laser printer at home I used a software postscript interpreter from LaserGo, Inc. called GoScript with my Panasonic KXP-1124 dot matrix printer. The tradeoffs here are time — three pages of text and equations take about an hour to process and print — and the fact that the end of a print wire in a dot matrix printer is much larger than a spot gen-

portion as the textbook for a senior-graduate level course I teach. Students can purchase the 300 or so pages reproduced on demand by a local copy shop in a soft binding for about \$15.00. An equivalent hard cover book from a major publisher would be about \$75.00. I hope to figure out some way of getting help to finish the remaining 200 or so line drawings in a reasonable time.

My now former editor pointed out that the major cost in producing a new book traditionally has been the cost of typesetting. More and more small publishers are

Continued on Page 48

# ZUG New Products

ZUG p/n 885-6017

PS's '92 Collection ..... \$20.00

**Introduction:** This disk contains a collection of new utilities and updated older utilities from Pat Swayne. There is a wide variety of utilities for your screen, printer, disks, and others.

**Requirements:** All of these utilities require MS-DOS version 2 or above, and any PC-compatible computer. Many of these programs are compatible with the MS-DOS 5 help system, so that after proper setup you could type, for example, HELP LPT2COM, and get a screen of information.

**Author:** Patrick Swayne, ZUG Software Engineer

Here is a description of the programs on this disk.

**LPT2COM.COM** — This program is designed for users of MS-DOS version 5 who have a serial printer. It takes the place of the CONFIGUR program that was provided with earlier releases of MS-DOS from Zenith Data Systems. With it, you can map an LPT parallel port to a COM serial port, with the exact type of handshaking required by your printer. LPT2COM supports RTS/CTS, DTR/DSR, XON-XOFF (DC1-DC3), ETX-ACK and IBM "Compatibility" handshaking.

**TIMEOUT.COM** — This program replaces another CONFIGUR feature that is missing in MS-DOS version 5. It allows you to alter the "timeout" value of any LPT or COM port. It can be used instead of the MODE LPT1,,P command when you run programs that require that MODE command. It is better than MODE for this because it does not leave anything resident in memory.

**XTABS.COM** — This is a memory resident program that automatically expands tabs to spaces when data is sent to your printer. If you have a printer that cannot handle tab characters, and you print files by COPYING them to PRN, this program will allow you to print files containing tab characters.

**FASTTEXT.COM** — This is a program that makes your computer display data on the screen faster when that data is put there by MS-DOS or BIOS functions. It has no effect on screen speed in the graphic modes or in text modes when the data is put on the screen by direct memory access. It is especially helpful if you have disabled video "slushware" or ROM shadowing, as when

you use the UNSLUSHV device driver. For those of you who are familiar with my FASTSCRN program, this program is even faster — much faster with some video cards.

**VTEST\_T.COM** — This program lets you test how fast your video system is in the text mode. You can use it for "before and after" tests with FASTTEXT.

**VTEST\_G.COM** — This program tests how fast your EGA or VGA video system is in graphic modes. You can use it to compare different systems or different brands of video cards.

**NSS.COM** — The New Screen Saver. This screen saver works on CGA, MDA, Hercules, VGA and some EGA systems. It can be set to blank your screen after a user-defined interval of inactivity, or you can blank the screen by typing a hot key combination. It blanks the screen by commanding your video card to turn off video, which results in a smaller program than when other methods are used. However, this method is not compatible with Microsoft Windows. The program can be temporarily disabled when you run Windows, if necessary.

**NSSCK.COM** — The New Screen Saver combined with a screen clock that can display the time in the upper right corner of your screen, even while you are running other programs.

**GBANNER.COM** — This program is a translation of a program called GOTHIC.ABS that appeared way back in the old HDOS days. It makes big-letter banners on your printer using "gothic" or old English style characters. It uses ordinary text characters to form the characters, so it will work on any printer. It is normally designed for use on wide paper, but will work on narrow paper if the printer is set for "compressed" text and tighter line spacing. There are a lot of banner programs around these days, but in my opinion, this one still makes the best looking old English style characters. GBANNER is actually a "filter" which takes its input from the DOS Std Input device, and outputs to the Std Output device. Therefore it can take input from the keyboard or a file, and it can output to your printer or a file, and other combinations are possible.

**NARROW.COM** — This little program sets an Epson-compatible printer to compressed characters and 7/72" line spacing, which is the perfect setup for using GBANNER on narrow paper.

**RBANNER.COM** — This program is

similar to GBANNER, but it outputs Roman-style characters, and it is designed to use narrow paper when a printer is running at the default 10 characters per inch.

**SFF.COM** — The Super File Finder can search all of the directories on a disk or partition for a specified file or files. Once the file(s) are found, depending on the command line switch supplied, SFF can delete the file(s), run a file (if it is a program), or make the drive and directory containing a file the default.

**FINDDATE.COM** — This is a file finder that searches by date rather than by file name.

**FILEDATE.COM** — This program lets you alter the "stamped" date and/or time on a file or group of files.

**D.COM** — The latest version of my alphabetizing, columnizing directory program.

**KD.COM** — Version 2 of my directory killer. This program can delete a directory, including all files in the directory, and all subdirectories and their files. If files with the read-only attribute are found, they are listed and you are given the opportunity to delete them or halt the operation.

**DTEST.COM** — This is the latest version of my disk testing utility. It will test all of the sectors on your disk (hard or floppy), and if bad ones are found, it can mark the File Allocation Table (FAT) so that they will not be used.

**720FIX.COM** — This program corrects a bug in ZDS BIOS's below version 3.4C. The bug affects writing to 720k 3.5" disks.

**UNSLFIX.COM** — The UNSLUSHV.SYS device driver (available on the ZDS COM1 bulletin board) does not work properly when used with MS-DOS 5 and the Z-549 video card. The INT 10 vector is not set up properly, and the CX register is destroyed when a mode set command (function 0) is issued. This program fixes the problem. If you have a program that does not work properly under DOS 5 and did under previous versions, it is possible that it could be because of this problem, and if so, this program will allow it to work.

**HTE.COM** — The Heath Terminal Emulator lets you use your PC-compatible computer as an H19 terminal on another computer, such as an H8. Provides partial emulation of H19 graphic characters when you run it in the text mode, or full emulation in the CGA monochrome graphic mode. I wrote this for myself so that I could throw away the old ailing H29 terminal in my



office and still use the old H8 when necessary.

**HTEE.COM** — A version of HTE for use with EGA or VGA video cards. Provides full emulation of H19 graphic characters and Z-100 escape sequences for setting colors.

**BT.COM** — Big Text re-types whatever you type after BT on the command line in very large characters. It is designed to be used in batch files to display messages in large characters.

**BTTEST.BAT** — A sample batch file showing how BT.COM works.

**BTMS.COM** — A version of BT that outputs to the MS-DOS Std Output device,

so that it's output can be re-directed to a file. Use BTMS if you want to make a file containing a message in large characters.

**WAIT.COM** — This program expects a number between 1 and 3600 on the command line, and waits that many seconds before returning control to MS-DOS. It is designed to be used in batch files to introduce delays. For example, it can be used in a batch file that displays screens of text, to provide a time that each screen is left on the screen. If you type any character at the keyboard during the delay period, the program exits before the delay time is up.

**MAGNIFY.EXE** — This is a self-extract-

ing archive containing the Magnify System for vision impaired persons. It is designed to be used on EGA and VGA systems, and provides ways to magnify parts of the screen while the video system is in the text mode (sort of like holding a magnifying glass over the screen). An automatic installing batch file is provided. The Magnify System is also available free on various BBS systems, and is provided here for your convenience.

**SOURCES.EXE** — A self-extracting archive containing the source code to the programs on this disk. ✨

#### Continued from Page 41

periods into account for sorting. Ignore the "-1" to "-4" entries for the moment. CSeq(ASC(K\$)) is the collating value to use for character K\$. Multiplying this value by the proper weight shifts it left:

16,777,216, by 24 bits

262,144, by 18 bits

4,096, by 12 bits

64, by 6 bits.

Watch out for the 12-bit shift. Unless you force long arithmetic with the "&" sign, the intermediate value will be calculated using short integers, and will overflow for any value over 7; that is, for any letter and all digits greater than "6". To build up the

first long-integer key, covering characters 0 through 4 in T\$, you would use some statement like Figure 6.

This looks like a lot of work, but you only do it once for each entry in the array. If you opt for letters only and three 5-bit values, use weights 1,024, 32 and 1; all collating sequence values must be less than 32 in this case.

To get an increase in sort capacity over the 3,600 string entries you can squeeze into RAM, you must throw away each string as you convert it to integers. You can't ever convert back, because you lose all information on capitals and accents. This is no problem for our KWIC index,

because the original text stays on disk and can be picked up unchanged after the sort is finished.

These algorithms are good for short arrays, but for over a hundred items to sort the processing times start to climb. In the next article, I shall show how these programs have been improved to give high-speed sorting of large tables.

#### Sample Code

The full KWIC indexing program will be printed with the second article in this series. Extracts which are referred to in this article are in Listing 1. ✨

#### Continued From Page 46

therefore using LaTeX or similar programs to sidestep this cost and are requiring the author to furnish them the document in that form. This is a way of shifting the cost of producing a book from the publisher to the author. This shift has also been facilitated by the increasing capabilities of low-cost personal computers and their programs. Authors are finding that in the 1990's they must learn many new skills and technologies unrelated to the topic of their writing in order to get published at all!

Another significant per copy cost for specialty books from major publishers is that press runs smaller than 2000 copies are uneconomical. It is also no longer practical to allow unsold copies to gather dust in a warehouse in anticipation of future sales. This is because of a change in the way the IRS treats such inventory, because some of the material becomes rapidly dated, and because students no longer keep their books but sell them after the semester, drying up the market for new sales in a very short time. Large volume textbooks are therefore going to new editions every three years. With the entire

book available as a set of postscript files, it is now feasible to print just a few copies at a time on a fast laser printer. Revisions are easily made in this format and increased printer resolution is rapidly becoming more affordable.

One of the things I had hoped to do with my book — and this is one of the things that was controversial to some reviewers — was to integrate the text more closely than had previous books with a very powerful computer program for solving the problems discussed therein. In fact, the problems that the computer program could solve tended to dictate the topics discussed in the text. The book could not, therefore, be a compendium of all current ideas on what this course should cover as many reviewers and some publishers wanted. The idea behind my approach was that the student still had to conceive the solution formulation but the computer program carried out the process very rapidly with no errors. The availability of this tool also permitted problems more closely resembling those encountered by engineers in the real world to be assigned. To keep the course from being one of learning com-

puter programs rather than the technical subject, I purposely tried to limit the number of computer programs the student had to learn for this course to one.

In looking down the road as to what engineers of the future will do it seems obvious to me that employers will demand more and more productivity from each one in order to justify the salaries of engineers relative to production workers. The computer has become the primary tool for increasing engineering productivity. However, employers cannot demand, except at their financial peril, the engineer use such a tool without his really understanding the limitations of the programs being run. The traditional instructional setting — the university — must therefore find a way to make the student better acquainted with the powers and limitations of this tool without foregoing provision of the necessary physical understanding behind the programs. Unfortunately, we've not really made a good start at this yet. ✨



A decorative border of stylized yellow and orange flames surrounds the central text. The flames are rendered with simple outlines and a gradient from yellow to orange, set against a black background.

# HADES II

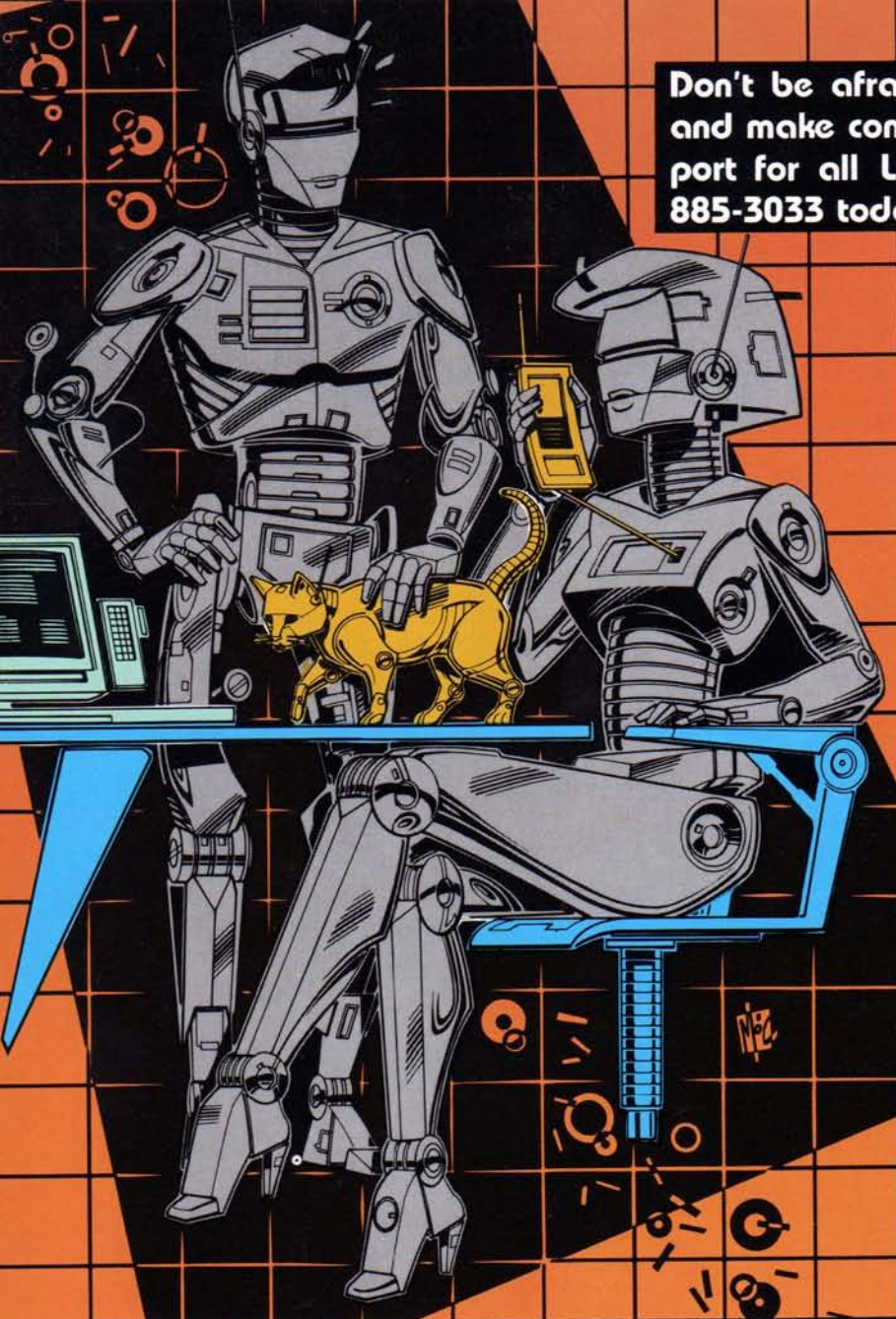
It's HOTTER than ever! Jam-packed with new features, HADES II still remains the easiest-to-use disk editor ever! Just look at some of the features:

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

Storage Buffer = 324288 Bytes
Storage Buffer Usage = 0 Bytes

Select Message (A-O), (F1) To List, Anything Else To Abort --) _
F1=List F2=Msg F3=BufF F4=Disk F5=Send F6=CfgF F7=Clr F8=Print F9=Exit COM
```

```
HUGMCP Configuration Help #1
This function allow The User To Be Chained, Depending Upon Which
    Mode You're In. Normally It's Valid To Use Ctrl Or
    Esc To Exit.

This function allow You To Change The Word Length. Normally You
    Can Set It To 8, 16, Or 32 Bits. This Value Is Acceptable By Most
    Terminals, And It's Necessary For XMODEM Protocol To Work Properly.

This function allow You To Enter Messages, Which Can Be Automatically
    Sent At A Later Date. To Use Character Messages Can Be Used.
    This Functionality Is Used To Contact Your Computer In A Remote
    Location. Be Careful This Program Is Only Controlled By Entering The
    Proper Control Codes.

Type (Ctrl) Esc To Exit Help, Anything Else To Continue
F1=List F2=Msg F3=BufF F4=Disk F5=Send F6=CfgF F7=Clr F8=Print F9=Exit COM
```

```
HUGMCP Configuration Menu:
a -> Modify Band Rate
b -> Modify Parity Type
c -> Modify Word Length
d -> Modify Or Add Baud-Messages
e -> Miscellaneous Functions
f -> Change Screen Color Assignments
g -> Display Current Configuration
h -> Make Changes Permanent

Select a-c, (F1) For Help, Anything Else To Quit --) _
Band 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: XOFF=H
Modem Port Set to: COM1

F1=List F2=Msg F3=BufF F4=Disk F5=Send F6=CfgF 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 885-2145