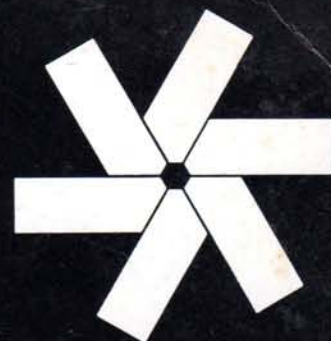


# REMark<sup>®</sup>

January 1992

The Official Zenith Data Systems Computer Users' Magazine



## The MastersPort 5Le



**Zenith Data Systems'  
Newest and Most Innovative Laptop**



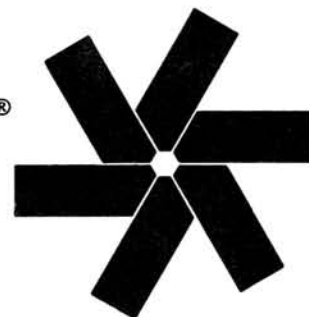
## *Share the Knowledge!*

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>

January 1992



The Official Zenith Data Systems Computer Users' Magazine

## The MastersPort 386 SLe: Assault on Battery

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

## Selecting and Using an Accounting Package: Quicken 4.0

*Kevin Steffen* ..... 7

## Introduction to C++ Twelfth Installment

*Lynwood H. Wilson* ..... 11

## Getting the Most From Your Computer Part 8

*John P. Lewis* ..... 15

## Coherent - Unix on a Budget

*Bill Wittig* ..... 19

## Keeping Afloat with Floating Point Numbers

*David W. Lind* ..... 23

## Perpetual Upgrading of a Z-248

*Nick Visco* ..... 35

## On the Leading Edge

*William M. Adney* ..... 37

## Dial Up Your Background Material

*Harold C. Ogg* ..... 43

## Advertising

<i>FBE Research Co., Inc.</i> .....	21
<i>Micronics Technology</i> .....	27
<i>LS Software</i> .....	34
<i>Quikdata, Inc.</i> .....	22
<i>Surplus Trading Corp.</i> .....	18
<i>WS Electronics</i> .....	10

## Resources

Software Price List .....	2
Renewal Form .....	4
Classified Ads .....	48

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
Initial	\$22.95	\$37.95*
Renewal	\$19.95	\$32.95*
		* U.S. Funds

Limited back issues are available at \$2.50. Check ZUG Product List for availability of bound volumes of past issues. Requests for magazines mailed to foreign countries should specify mailing method and include appropriate, additional cost.

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

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

Articles submitted by users and published in REMark, which describe hardware modifications, are not supported by Zenith Data Systems Computer Centers.

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

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

Copyright (c) 1991, 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 TRANSCIVER	885-8016	HDOS	AMATEUR RADIO	20.00
MORSE CODE TRANSCIVER	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	ATRONOMY UTIL	20.00
TCSPELL	885-8044	MSDOS	SPELLING CHECKER	20.00
ULTRA RTTY	885-6012	MSDOS	AMATEUR RADIO	20.00
YAUD (YET ANOTHER UTIL DSK)	885-6015	MSDOS	UTILITIES	20.00

## Attention!!

### Zenith Data Systems Owners

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

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

### ORDERING INFORMATION

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

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

## REMark Magazine Subscription & ZLink/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 ZLink-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 System 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 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

## The MastersPort 386SLe:



# Assault on Battery

Nathan E. Baker  
Zenith Data Systems  
Technical Writer

In a continuing effort to extend the battery-powered computing time of the portable personal computer, battery makers have been experimenting with dozens of new schemes for lengthening battery life and decreasing the time it takes to put a decent charge into the things. Why, for instance, does it take some batteries eight hours to reach peak charge, when the same battery only provides two or three hours of useful life? I don't know — it doesn't matter. The physics of chemical reactions occurring during charge-discharge processes aren't part of this article. Take some chemistry courses if you want to learn about making better batteries. If you want to learn about what Zenith Data Systems is doing to beat this problem, read on.

### Power Consumption and On-Time

In the classic computer design, all the circuits in the computer are kept powered-up. This amounts to a lot of wasted energy most of the time (do you need the display turned on if you're not looking at it? or the I/O ports enabled if you don't have anything connected to them? or the disk controller circuitry energized if you're not accessing disk drives?). For desktop computer design, this isn't critical at this point in time. In the future it might be, when energy costs will make consumers think about the wattage of the light bulb in the refrigerator. But for now, energy conservation techniques have only been applied to portable computer designs. The techniques I describe here, and which have been masterfully incorporated in the MastersPort series of portable computers, will probably find

their way into desktop designs someday.

A given amount of battery power can only go so far. When a portable computer is powered-up, it has the same host of energy parasites as its bigger desktop cousins, only the amount of energy available is limited. The scheme used by the designers of the Zenith Data Systems MastersPort series is to put the energy parasites on a leash.

The major consumers of energy in a portable computer are the display and the disk drives, followed by the circuits that drive the serial port and the parallel port. Lately, even the cheapest portable computer on the market can figure out when to turn off the display backlight (the part of the display that requires the most energy), and most of them can power-down the hard disk drive if it isn't being accessed. Most can also disable the I/O ports. The MastersPort series controls power consumption in these ways plus some more modern methods. The methods that allow the most battery conservation are possible due to a new group of microprocessors — the SL series.

The MastersPorts use an Intel 80386SL microprocessor that works like an enhanced 80386SX. The 80386SL includes the following features:

- system clock control
- system management interrupt control

**System Clock Control** — The 80386SL generates and controls the clocks sent to the system. These clocks include the AT-bus clock, the system clock, and the numeric coprocessor clock. By manipulating these clocks, the circuit groups in the computer can be systematically halted (which

reduces power consumption). This feature is essential during the rest and standby modes that these computers rely on.

**System Management Interrupt** — The flexible power consumption ability that lets the MastersPort take petite bites of battery power are made available by use of the system management interrupt, which is unique to the SL series of microprocessors. The system management interrupt is similar to the non-maskable interrupt, but is used to control power management functions. (The system management interrupt has higher priority than the non-maskable interrupt.)

A number of events can trigger a system management interrupt, the major ones being the rest/resume/standby modes, which can be invoked by the user or by certain conditions occurring within the computer. All computers in the MastersPort series can use these modes, but the MastersPort 386SLe has the best power saving ability (so far).

### Power Usage: Off, On, and In-Between

Power consumption in the MastersPort series can be categorized in a number of ways, which are all relatively hard to quantify because they have to consider the actual use of the computer. Generally speaking, we can call them moderate, insignificant, and minimal.

Moderate power consumption occurs when the computer is on and under active use (keys are being pressed, the LCD is on, and a disk drive is spinning a disk). Depending on factors such as disk drive accesses and I/O port usage, battery power under moderate usage lasts between two and three hours.

Insignificant power consumption occurs when the computer is off. The only power being used is what is required to keep the real-time clock running. The battery can last a month or more in this state without being charged, and you're only about 10 seconds away from being up-and-running.

Minimal power consumption occurs when the computer is in one of the in-between states; rest and standby. Power consumption in these comatose modes depends on which mode is invoked.

Standby mode can occur at any time you specify. This mode is similar to the classic power conservation schemes, but goes a little further. Standby mode occurs automatically if you don't touch a key on the keyboard for a certain amount of time (which you specify in the Setup program). The screen goes blank and the back-light turns off, and power consumption drops to a level where only essential circuits are kept running (some circuits go into a special "sleep" mode specifically designed to enhance battery life). System memory refresh is also slowed down, which decreases current drain. When you press a key on the keyboard the system springs to life, ready to go back to work at the point where it went into standby mode. If necessary, this mode can remain in effect for up to 8 hours (with a fully charged battery).

Rest mode can only be invoked by a key combination, or by pushing the power-on button after programming the button to invoke rest mode. In rest mode, the computer can keep a program running in memory (even in protected mode memory) in the same state as it was when rest mode was invoked, similar to standby mode. The difference is that instead of certain circuits going to sleep, they are turned off. To resume operation at the point where you left off, all you have to do is touch the power button again. A MastersPort 386SLe, with a fully charged battery, can enter rest mode and stay there for over a week (longer if only 2M of RAM are populated).

### Speaking of Memory

Memory in the MastersPort 386SL and the MastersPort 386SLe comes in a variety of packages.

First there's the base system memory, composed of two 1M x 4, 80 nanosecond, slow refresh DRAMs soldered to the main board. The system memory is the place where the slushware (BIOS code transferred to RAM) and the system management control features reside. By adding SIMMs, this memory is expandable to 8M. The upgrade process can be done by even a novice user — remove a hatch to expose the sockets, and then insert the SIMMs. Note that battery power will diminish somewhat as increasing amounts of RAM are added.

The next important piece of memory is the cache RAM, which gives a turbo boost to most processing tasks. This is a block of 64K of high speed static RAM (two 32K x 8 SRAMs). The 80386SL contains the cache control circuitry.

The system BIOS is contained in a 128K x 8 ROM that can be reprogrammed by the user. At power-up, the system BIOS is transferred to system RAM, where it can execute much faster. The copy of BIOS that resides in RAM is called slushware, and exists in a specially protected area of memory. (During rest modes the video display memory is also stored in the slushware RAM.)

This BIOS ROM is not just any ROM. It can be reprogrammed by the user, without the need to disassemble the computer. If a new version of the BIOS is released (due to new peripherals or with optional features), the system can flush the old BIOS code and reprogram itself from a floppy disk that has the new code. (Another ROM on the system board assists the BIOS ROM with the reprogramming function.)

To store passwords, the computer contains a 4 kilobit EEPROM. This EEPROM also saves setup information in case the user makes a mistake running the setup program, or the backup battery runs down and the system setup information is lost.

The real-time clock also contains 114 bytes of RAM. Some of this RAM is used for time-keeping functions, and the rest is used by the system for storing setup information. When the system is turned off, the real-time clock is still powered. If the computer is not plugged into a charger at this time, the real-time clock will use battery power. But this system has more than one battery.

### The Main Battery and the Backup Battery

The MastersPorts have two batteries; an intelligent main battery pack that provides the bulk of the system power, and a non-removable backup battery that performs two functions.

The main battery pack is microprocessor controlled. When connected to the AC adapter, it charges in one of three modes. These modes occur automatically, and the microprocessor insures that the battery will safely reach peak charge in all charge modes.

- The first mode, trickle charge, is nothing new. It maintains a charge on a battery that is already at or near peak charge.
- The standard charge mode occurs when the system is turned on and connected to the power adapter. In this mode the battery takes about ten hours to reach a full charge, then the battery will trickle charge after that.
- The fast charge mode pumps over an ampere of current into the battery to fully charge it in three hours or less. This mode only occurs when the computer

is turned off. When peak power is reached, trickle charge takes over.

Note that the battery pack must be installed in the computer in order to be charged.

The backup battery is internal, and is composed of power cells that recharge at a rate depending on what the user is doing. There are three basic recharge modes for this battery.

- **Battery powered mode** — This mode occurs when the machine is running on battery power. In this mode the backup battery takes a small charge from the main battery. By doing this, the backup battery maintains enough power to keep the computer running while the user changes main battery packs. Interestingly enough, I found that I could change main battery packs while a program was running, and it didn't stop the program or harm the computer (although this is not recommended).
- **AC power-on mode** — This mode occurs when the machine is connected to an AC adapter and is powered-up. The backup battery charges the fastest in this mode (about 15 hours from a fully dead backup battery).
- **AC power-off mode** — This mode occurs when the machine is connected to an AC adapter and is powered-down. Since most of the power is being routed to the main battery pack for charging purposes, the current available for the backup battery is somewhat less. Average backup battery recharge time is about 18 hours in this mode.

### The MastersPort 386SLe

Since previous articles have already discussed the MastersPort 386SL, the remainder of this article is dedicated to the MastersPort 386SLe. This computer represents the peak of current portable computer design. It packs the following standard features in its 7.0 pound notebook-style chassis:

- an Intel 80386SL microprocessor operating at 25MHz (adjustable)
- a socket for an optional 25MHz 80387SX coprocessor
- 1.44M, 3.5-inch floppy disk drive
- 85M hard disk drive
- 2M system RAM
- 640 x 480 VGA-compatible transmissive LCD (with palette control)
- external VGA video port
- enhanced high-speed parallel port
- serial port
- two-level password protection
- 82-key keyboard (101-key compatible)
- a socket for an optional internal low-power data/FAX modem.

Zenith Data Systems also includes MS-DOS 5.0 already installed on the hard disk drive. They even provide a complimentary

Continued on Page 14



---

---

## Selecting and Using an Accounting Package:

# Quicken 4.0

Kevin Steffen  
319 Sycamore Circle  
Dalton, GA 30721

### Choosing an Accounting Package and the Preliminary Setup

This is the first section of an article on selecting a software package to do personal and small business accounting, and in using it afterward. I will use my own business as an example of the steps to choose an accounting package and doing preliminary setup work in this first section. The remaining sections will deal with how to set up and use the software package I chose; Quicken 4.0.

A little background is in order to understand why I was looking for an accounting package. Since I took a basic accounting course in college I have kept daily expense and income logs in order to determine where the money went in my budget. When I bought a Z-100 in 1983 I started doing monthly and yearly spreadsheets. A help at tax-time, the use of a spreadsheet took around an hour and a half every month in addition to the short time to log the data every day.

When I started setting up my business two years ago I upgraded the computer to a Zenith Data Systems 286 and spreadsheet software to Quattro Pro. Using spreadsheets to track all of the expenses of the business and supporting a family was taking ten to twenty hours a month to update and correlate over twenty spreadsheets (accounts). So, like any good computer consultant, I became one of my own first customers.

I sat down with pen and pad and drew up criteria for the software I wanted to purchase, coming up with this list of mandatory features.

1. Cost — around \$100, preferably below.
2. No double entry, that is what computers are for. Any logging of expenses or

income is done once, the program takes care of entries in all related accounts.

3. Do things my way - cash accrual. It is not real, spendable money until I am paid.
4. Accounts use names, not numbers, so no remembering account codes.
5. Reports available for total budget and totals by categories, personal and business.

A short review and discussion with my business partner and wife added a few nice to have features.

1. Importation/exportation from/to Quattro Pro for spreadsheets from last year.
2. Exportation to tax preparation packages.
3. Print reports I design, ease of modification of existing reports.
4. Billing (accounts receivable) capability.
5. Color capable and easy to learn.

Grabbing the modem, I hooked into the local bulletin boards and downloaded shareware accounting packages. From the reviews and ads in the magazines at the local library I came up with a list of candidates to look into. The list contained a dozen products, four shareware and eight on the commercial market.

The first step was to evaluate the shareware products while arranging to borrow manuals for the commercial products. After checking for virus on the shareware, the documentation was examined. This eliminated all but one as being too basic or designed for a different accounting style. The remaining shareware program was installed and test data entered. It was rejected due to personal preference, not liking the way the data was entered.

The reviews on the commercial products were read and set aside. It was obvious the reviewers did not know what I was

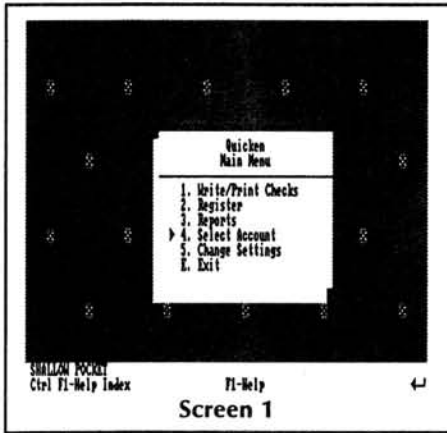
looking for. A cursory examination of the manufacturer's documentation eliminated three of remaining eight products for overly rigid accounting requirements. Two more were eliminated after closer review of the manuals for stressing payroll and accounts payable at the expense of tracking general expenses.

This left three products for evaluation. One was in use at a construction business, one was available for demo/evaluation at a computer dealer in the local area, and the third was available on a thirty day trial period through a mail order software house. The product at the computer dealer was difficult to install; it did not work as the manual said it should. The construction company manager said reports were difficult to generate. When I tried to set up accounts for testing, it was unwieldy but acceptable.

As usual, the last one tried fit best. Ordering Quicken 4.0 from the software house on thirty day trial, I began an analysis on the accounts I would need to set up. As an absolute minimum there had to be a checking account, a cash account, a credit card account, a savings account, an IRA account, an auto loan account, a tuition account, and a money market account. There were over thirty separate categories of fund usage for business and personal activities of which some affected the tax status by being full deductions and some were partially deductible.

In this example I wanted to show you the basic steps I recommend to businesses and people with computers looking for new software.

First, what type of package do you want, and why do you want it. If the package does not save you time or money, it is



Screen 1

a luxury or a nuisance, and unneeded.

The second question to ask yourself is what do you want the software to do. Make a list. Look it over. Expand on it. Once you have a good list of requirements, you can start the process of finding a program to fit your needs.

Begin the process of finding a program by listing all programs from many sources. You can read reviews and ask people using the product, but unless their criteria for the program matches yours, their perfect solution will not be yours. I did not mention names of programs I examined and found lacking because the programs are not bad, they just did not meet my criteria, my expectations, or my personal taste. One of them may be just what you would be looking for.

Trim your list of candidate programs by reading the manuals supplied by the programmers. Third party "how-to-use..." books are an alternate source of this information. Contact local dealers or people in your local PC User's Group for the manuals. These are also a source to test drive a program by looking over their shoulders.

The final trimming of the list is done by evaluating the software. Most commercial programming corporations will supply demo disks, some will have evaluation packages for you, or a thirty day return policy. Ideally, you will come up with one clear winner, but you may have two or three in close running. In that case go back to your criteria list and find the best fit. If all else fails, you can choose the one with the most additional nice-to-have features.

If you have an accountant, use him to help you set up your criteria. Your accountant should know your business well enough to tell you what you really need, and what would help him with your business and personal accounts. Again, after you have selected the package to purchase, your accountant is a resource for you to utilize in determining what you need to track, and how you need to set up your accounting package. It is imperative to have an idea of how your money is placed and used before beginning to use the software. As in pro-

gramming, design first, code second.

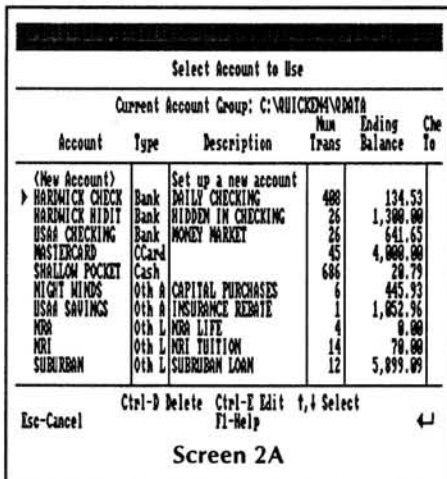
When Quicken came in, it installed easily, looked good, entered data in an easy manner, and acted just the way I wanted. In the next segment of the article we will be installing Quicken 4.0, setting up accounts, and tying accounts together while entering data.

### Installation and Setup

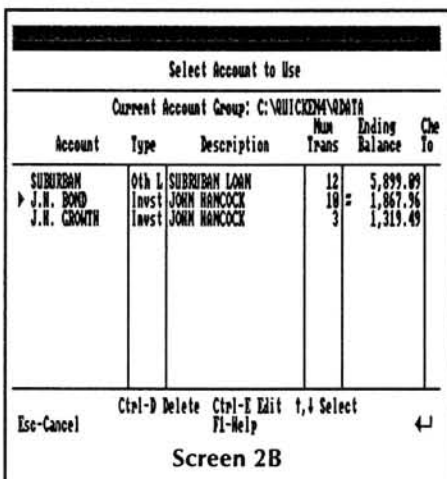
In this second section of my article on selecting a software package to do accounting for a small business, we will cover installation and set-up of Quicken 4.0.

While waiting for the software to arrive, I studied the borrowed manual to determine how the software would affect the account system I was wanting to set up. Reading the manual helped in the design of the system, combining several of the categories by QUICKEN's ability to have categories and classes within the categories. I also saved the entire hard disk to floppy disks during this time.

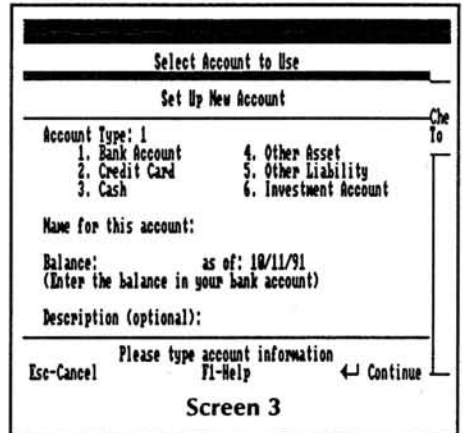
When the package arrived a week later, I was ready to go. Following the instructions, I made a copy of my master disk, then proceeded to install QUICKEN. At the DOS prompt for the floppy drive I typed "INSTALL" and pressed enter. The default was for a monochrome monitor so



Screen 2A



Screen 2B

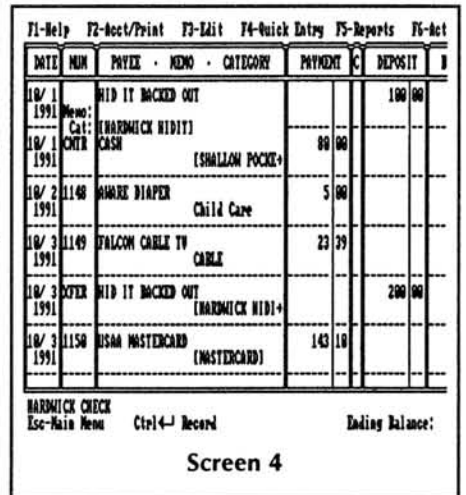


Screen 3

I pressed "F2" to select a color monitor. The next two prompts were to determine where the package was to be installed. I let these default to the hard drive, c:\quicken4. The install asked if I wanted to turn on the bill minder feature, I changed from the default "yes" to "no". Within five minutes from starting, installation was complete.

Installation was quick, simple, and clean. The process placed all the files under QUICKEN4 on my C drive except for "q.bat" in the root directory. I renamed "q.bat" to QUICKEN.BAT. In this way I just type "QUICKEN" from any prompt and I am in QUICKEN. The entire catalog takes less than 1 meg of disk.

When I typed QUICKEN, the program's



Screen 4

main menu came up in full color. Very pretty, you could almost bet it was going to be user friendly from the menu. It is possible to move the pointer down to the desired action and pressing enter to start it. You can also type the number (or E for EXIT) to get there faster. For those stuck on the function keys, pressing the function key is the same as typing the number. It is almost foolproof.

Since we are going to set the system up from scratch, first select option 4 to get to the account screen, then select the <NEW ACCOUNT> option (see screen

2a). The new account screen overlays the primary screen to select an account as seen in screen 3. Going to my prepared list of accounts I began creating accounts with totals as of January 1, 1991 which was two months earlier. First came four bank accounts, two at HARDWICK, and two at USAA. One credit card account, one account for cash, one asset account for business capital, three loans, and two investment accounts were created with meaningful names to cover everything I thought I would need to keep track of where my money was. Of course I was mistaken.

The next task was to place the transactions in appropriate accounts going back to the first of January and ensure the accounts balance between and within themselves. You can enter an account by the number 2 action from the main screen which automatically places you in the account named at the bottom left of the screen (in screen one, shallow pockets), or by moving the pointer to the desired account in the select account menu (screen 2a and 2b). Looking at screen 4, everyone should recognize the ubiquitous check register from their checking account.

Filling in the account register is just a little more complicated than filling out the check book register. The date is an automatic repeat of the last date entered unless you decide to change it. Check numbers are entered next or a comment such as XFER for moving funds between accounts, EFT for electronic fund transfers, or ATM for automatic teller machines. I took mine right off the back of my bank statement. The payee, memo, and payment or deposit fields come right off the check face. The balance is automatically calculated for you.

The complexity comes in the remaining field "category" and its only subfield "class". I made category use mandatory, that way I can tell what the money was spent on. The default categories for home and business were used for this first entry of data. I set up a class for each business I run, one for personal items, and one for each client. Every time I entered a transaction and did not supply a category QUICKEN prompted me for one, when I entered a category such as "POSTAGE" I also entered a class so I could bill the cost to the appropriate entity. This function cut the number of accounts and categories in half.

All accounts were entered and balanced in this fashion. I did not take advantage of the transaction memory feature to remember all the details of reoccurring transactions such as cable TV bills. The only problem came in using a split transaction to enter paychecks were some of the money I earned automatically went to the 401K fund, some deposited to checking, and some received as cash at the bank. A call to the INTUIT help desk solved the problem for me. The manual did not explain that in

an instance like this the amounts for net pay and 401K deposit are entered as positive numbers with appropriate category or account name in the category field. The cash is entered as a negative number after the net with the appropriate cash account name as the category. In screen 4, the names printed under the category heading enclosed in square brackets are account names which QUICKEN automatically made the appropriate transaction to, the others are simple category names.

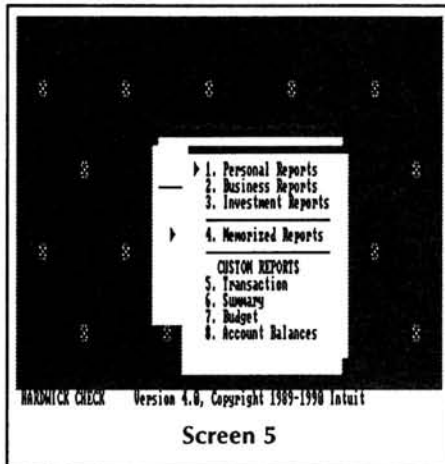
After ten hours of data entry (wish I had a secretary to do this), all accounts were set up. All transactions in each account were entered, and the accounts balanced with the bank statements. Two and a half months data took ten hours, less time than it took to enter on all the separate spreadsheets needed in the way I had been doing it. Best of all, QUICKEN did all the double entry bookkeeping when I moved money from account to account. Next comes fine tuning the system.

### Fine Tuning the Database

In this last section of my article we will cover fine tuning the database of Quicken 4.0 to suit individual needs, and other functions that can be used.

Having entered the previous month's data into the accounts, it is time to tailor the accounts to our use, fix any errors in entry, and prepare to put our new accounting package into production.

From the main menu we travel to the



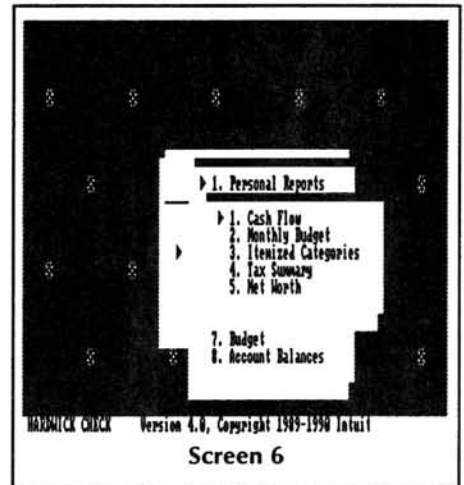
Screen 5

reports screen (screen 5), and select the account balances report. The report is previewed on the screen and not printed until function key F8 calls for printer settings and optional headings.

As the report indicates, I had made two rather gross errors during the initialization of accounts. The 401K plan is a form of IRA because you cannot draw from it without tax penalties, therefore it should be an investment account instead of a bank account. The second mistake was in setting USAA SAVINGS up as a bank account, in

reality it is an insurance rebate fund that cannot be touched unless all insurance is canceled. Therefore, it should be another asset account.

I corrected the accounts and ran the net worth report from screen 6 which is from the personal reports selection of screen 5. It is the same as account balance report, differing only in the heading. In order to change the account type it was necessary



Screen 6

to delete and recreate the account. This was done for the USAA SAVINGS account. The 401K account was deleted since the program was canceled and the funds were marked to roll over into the IRA, J.H. BOND account. Minor corrections to change the name of an account or the description of it is a simple matter of invoking the editor from the SELECT ACCOUNT SCREEN. This can be seen where KAL EQUIPMENT's name and description were changed to NIGHT WINDS in the net worth report.

Having made the changes to the accounts, it is time to verify all transactions in the accounts. To do this a list of all transactions for all accounts is needed. This is found in the ITEMIZED CATEGORY REPORT from screen 6. Since we will be making changes to the categories and classes for the transactions, running the TAX SUMMARY REPORT is called for since the taxation information bit is set in the category. At this time you may note that nine months' data has been entered. The TAX SUMMARY REPORT ran eleven pages and the ITEMIZED CATEGORY REPORT ran thirty-one pages.

Unlike some accounting packages, the transaction is not locked in stone, it is simple to modify a category, class or amount. In doing the easiest first, we tackle the TAX SUMMARY REPORT. There are two tweaks to the data needed. In my particular case, the category "repairs" is not a tax deductible category since it does not refer to a business's capital equipment repairs, I use it for home repair and maintenance. I simply turned off the tax bit by

changing it to a "N" by going in to modify one of the transactions in the "repair" category. The other tweak was a bit more major. When a check is returned uncashed, an insurance rebate comes in, or (in some instances) State and Federal income tax refunds are deposited, this is not new income subject to tax, simply refunds of overcharges. Going to one of the five instances which showed up on page 2 of the TAX SUMMARY REPORT, we find the particular transaction and put in "refund" as the category. Since this category does not exist, we are asked if we want to add it to the list. Following the cursor through the screen adds the category with all pertinent information. We then have to go to each of the other four transactions and change the category to "refund".

The ITEMIZED CATEGORY REPORT was of the same format as the TAX SUMMARY REPORT, but covered all categories for all accounts. There were twenty tweaks needed in the first three months' data; in the next six months' data there were only four. Over half of the tweaks needed were creating new categories or classes for trans-

actions that had been lumped under MISC because I could not find a category to put it in. The remaining fixes were to transactions with mis-spellings, those which had been placed in the wrong category to begin with, and two split transactions with the amount spent showing up as positive instead of negative due to improper entry of the amounts spent.

There are quite a few advanced features that enhance QUICKEN's usefulness for the small business. Check printing is fairly easy if you are going to purchase the checks for it and write a fair number of checks every month. QUICKEN is set up to use CHECKFREE to pay bill electronically, more lead time is required with this method to pay bills, but some will find it useful. QUICKEN will memorize recurring transactions to reduce entry time for these actions. It will remind you when a bill is due or when you need to run some checks when the billminder feature is invoked at start-up. Some new customized reports can be set up when desired.

The features that make QUICKEN valuable to me is that it can be based on cash

bookkeeping. I do not have to make two entries in two accounts when I pay a bill, QUICKEN does it automatically. Reports are available preformatted to help with taxes, cash flow, and budgets. It can be set up for a payroll and billing when the business grows.

Additionally, if someone were to set up a bookkeeping service, or wanted to run multiple businesses, QUICKEN will allow them to set up multiple account groups, one for each business. Year end processing and archival is easily accomplished if desired, but it is not mandatory. QUICKEN will transfer data to spreadsheet programs which can read LOTUS .WKS or .WK1 files. Most tax preparation programs will import data from QUICKEN to eliminate rekeying of data.

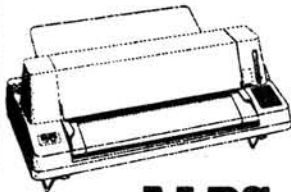
This week INTUIT sent me an announcement for QUICKEN 5.0. The enhancements are truly impressive for many businesses. Laser printer and mouse support, loan amortization calculation, more flexibility in reports, expanded memory support, and a separate version for running with WINDOWS. ✨

# W S Electronics

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

1106 State Route 380, Xenia, Ohio 45385

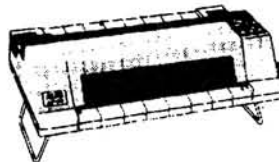
## YOU AND YOUR BIG IDEAS.



- \* 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.

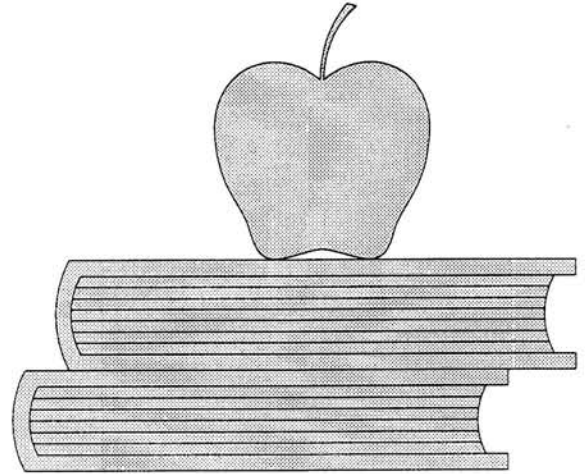
---

---

# Introduction to C++

## Twelfth Installment

Lynwood H. Wilson  
2160 James Canyon  
Boulder, CO 80302



Last month I started describing the design and development of an assembler in C++. I had some difficulty with the design, and found little in the books to help. Just to keep me thinking, the client provided a near continuous flow of changes to the specification as I worked on the design and later as I worked on the code. And the need for early completion of the project denied me the luxury of waiting for the specification to solidify before I began writing the code. I was forced to begin work on the code for any part of the specification which seemed fairly stable. If this sounds like a nightmare, don't be fooled. It is business as usual for a freelance programmer. This project differed from the norm in that the people I was working with understood my position, and did not blame me for the extra work and cost that resulted from their changes and their need for speed. Even though the job has run nearly twice as long as we originally expected, they understood why and they like the program quite well. Clients like these are beyond price, and I look forward to working with them again.

### Design

When we left the project at the end of the last installment we were selecting the classes which would be used in the program. This is the first step in the design. It helps if you have a firm specification before beginning this step, but it is certainly not necessary, as we will see.

As with classical structured design, these decisions are not irrevocable. At any point in the development of the program you may realize that things are not working out as you planned and as a result change

your mind about an earlier decision. In structured design this insight was usually saved for use in the next program since trying to change the basic structure of a program in process usually caused it to disintegrate. In an object-oriented program, major changes are much cheaper, but still not free. Whenever your hindsight shows you a better way to design the program you are working on you must balance the cost to change it (and the possibility that you will hit a block that could take a lot of time) against the potential gain. I have found that I feel a lot freer to make such changes in C++ than I have in other languages because the cost is so much lower. I think that most of us should do more structural rework than we do, and now we can without prohibitive cost.

Build something nice, rather than just sufficient. Think of the maintenance if not the art.

I just read last month's installment, and noticed something I had missed before. I was dealing with two different kinds of objects without noticing. The `Source_File` object is derived from the specification, from the statement of the problem. No matter how the assembler is written there must be a source file. The `Symbol_Table` object, on the other hand, is an artifact of the solution. It results from my picture of how the assembler will work. It is not a necessary part of the program. For example, there are one pass assemblers which do not create a symbol table but instead go back and patch in the values every time they encounter a symbol definition. It seems reasonable to first design (or incorporate into the design) the objects which are derived from the specification, since they

are somehow more fundamental, less likely to be omitted regardless of how the design turns out. Then figure out what auxiliary objects are necessary from the data required in the creations or operations of the first set of objects.

Sometimes it is not clear which kind of object we are dealing with. For example, which kind is a line? Actually, that's not a good example of an ambiguous object, although it's not obvious until you study the specification for the language to be assembled. The assembly language which the assembler must assemble is specified in units of one line. Unlike most high-level languages, assembly language requires only one statement on a line and the carriage return ends the statement. So the line is an object derived from the problem, though it takes a bit of thought to identify it from the specification. So are the fields and arguments in the line since they also come from the description of the assembly language.

The only other object which was not derived from the specification (besides the `Symbol_Table`) is a list of constant strings. I use this to look up strings from the source program to see if they are instructions, directives, reserved words of other kinds, etc. It is a fairly low-level tool which I've used in other programs. Note that the lower the level of a class, the more often you can use it in another program. Alas, the other edge of the sword is that the lower the level of a class, the less work it saves you when you re-use it. I'll include the code for this useful class at the end of this installment.

Depending on how you look at it, this object might be derived from the specification. It seems to me that although the

reserved words are in the specification, the idea of an object consisting of all of them to be used for reference is not directly implied in the specification. However, it does not matter which way you think of it.

Another idea which came out of this project is to treat finding the objects like a brainstorming session. Write down everything you think of, do not be critical, do not mind that some choices seem to eliminate other choices, just write down every word which could possibly be made into an object. After you have every one that you can think of, then start weeding the bad ones out, making judgments, selecting groups, etc.

When you get down to determining the relationships between objects, think of these two. If an object is a kind of other object, perhaps it should be descended from the other object. For instance, a car is a kind of vehicle, and should be a descendent of vehicle and inherit the common vehicle characteristics. On the other hand, look at a car and a wheel. Neither of them is a kind of the other, but there is surely a relationship. A car has a wheel. In this case, the wheel object should be made an element of the car object.

### Symbol Table

Last time we had written the pseudocode for the symbol table constructor when I ran out of space. Here is the complete declaration for the Symbol\_Table object (Figure 1).

The first item, symbol, is an array of Symbol structures, each containing a string which is the symbol and an int which is its value. This is the symbol table itself. The second item is an index to symbols in the array which points to the next empty space. This is where the next symbol is to be added. Address is the current value of the program counter (in the program being assembled). The constructor needs to know the address of the current line of code so it can give the address to an address label at that point in the code.

The constructor function builds the symbol table. It reads in the source file a line at a time from the Source\_File object, and if the line contains a symbol definition and the symbol is not in the table it puts it in.

As for the other functions, find\_sym()

```
class Symbol_Table {
    Symbol symbol[MAX_SYMBOLS + 1] // table of symbols & values
    int index; // next empty space in symbol table
    int address; // address of the next instruction
    int find_sym(const char *token); // true if token is in symbol table
public:
    Symbol_Table(Source_File *src_file);
    int current_address(void) {return(address);}
    Symbol get_sym(const char *token); // get symbol struct from sym table
    Symbol get_sym(int num) {return(symbol[num]);}
};
```

Figure 1

returns true if a symbol matching the string passed into it is in the symbol table. The function current\_address returns the value of the address variable. The function get\_sym() is overloaded. If you call it with a string, it returns the corresponding symbol structure if one exists. If the string is not in the symbol table, it returns a null symbol. However, if you call get\_sym() with an int as argument, it returns the symbol with that number as index in the array of symbols.

So we have the symbol table object designed, and the first pass of the assembler through the source code is complete. Note that we have not yet written any code, except for a few simple bits inline in the object declarations.

### Second Pass

The second time the assembler passes through the source code it will translate each line of the source code into the equivalent object code. It will use the Source\_File object and read the source using Line and Field objects, just as the Symbol\_Table object does. The result will be an Object\_File object. This all seems clear, but at this point I got stuck. How can we best get from the source to the object? The process is not difficult, merely find out whether the line contains an instruction and if so then translate the instruction and its arguments (if any) into machine language. Perhaps this is a problem best dealt with using simple procedural code. But it is unwieldy, a giant list of IF statements to choose the translation code based on string comparisons to decide which instruction it is. There must be a better way.

The method I finally worked out is to create a separate object for each different machine instruction. Each of these objects knows how to assemble its particular instruction, including the variations introduced by arguments to the instruction. These instruction objects will be held in an array. When assembling an instruction the program will step through the array until it finds the object which matches the instruction, and then use that object to generate the machine code for the instruction.

First, I declare a base or ancestor class

```
class Nop : public instruction {
public:
    Nop(void) {strcpy(name, "nop"); base = 0x00000;}
    long assemble(char *sline);
};
```

Figure 2

called Instruction.

```
class Instruction {
public:
    long base;
    char name[9]; // each Instr.
                knows its name.
    virtual long assemble(char *) = 0;
};
```

Note that this is a pure base class. Because we set the function assemble() to 0 it is impossible to create an instance of this class. This function can have no body, it is only here to show that all objects of type Instruction or descended from type Instruction have a member function called assemble() which takes a char pointer for a parameter and returns a long. By declaring assemble() to be a virtual function, we force late binding. In other words, the actual code to be executed when the function assemble() is called will be chosen at run time rather than at compile time. If this isn't yet completely clear, wait a bit and watch how it works.

The next step is to declare a child of the Instruction class for a particular instruction. I chose the simplest one (Figure 2).

This class inherits all the attributes of the Instruction class, plus the ones we have added. The assemble() function of this class is a real one rather than a place holder. And this class has a constructor which gives values to the variables name and base. Note that the constructor is simple enough for its body to be included in the declaration, while the body of the assemble() function will be defined elsewhere.

The assemble() function takes a pointer to the source line as its parameter and returns the object code, the machine instruction for this line. Since the machine instruction for an NOP is 0, what then does assemble() need to do besides return the base value? We haven't discussed errors yet, and I will not go into the subject as deeply as it deserves for lack of space, but the other job done by assemble() is to check the source line for errors and write error messages to an Error\_File object. In this case, since NOP does not take arguments, assemble() will issue an error message if it finds any. So here is the pseudocode for assemble().

```
long assemble(char *sline)
{
    if(there are any args)
        error message
    return(base)
}
```

So now our program can assemble one instruction. How can we best extend it to handle more?

The above assemble() function will

```
class Pop : public Nop {
public:
    Pop(void) {strcpy(name, "pop"); base = 0x20000;}
};
```

Figure 3

work for all instructions that, like NOP, take no arguments. All we must do is to provide the correct base value for each of them. This may easily be done by declaring new objects which are derived from the NOP object and inherit its characteristics, including its assemble() function. See Figure 3 for an example.

Note that the only difference between the Pop object and the Nop object is the value assigned to the variable base. The pop instruction produces the value 20000 (in hexadecimal) and the nop instruction produces the value 0. An error message will result in either case if an argument is present. The instruction objects for the rest of the instructions which do not take arguments are similarly descendants of Nop.

The instructions which do take arguments are a bit more complex. The custom chip for which this assembler is designed uses a relatively long machine instruction word of 21 bits and all machine instructions are complete in one word. This means that any arguments to an instruction are incorporated into the machine instruction. The actual value of the argument becomes a part of the instruction. This is done in several different ways, creating several different instruction formats. The details are not relevant here, but the result is that the instructions are divided into groups, such that the members of each group can each use the same assemble() function and the group members differ within the group only in their base value. The instructions described above under Nop comprise such a group. They all take the same number of arguments and they all use the same assemble() function.

Here is an example of an instruction which takes one argument which must

```
class Sjump : public Instruction {
public:
    Sjump(void) {strcpy(name, "sjump"); base = 0x12000;}
    long assemble(char *sline);
};
```

Figure 4

have a value between 0 and 7. It is a short jump instruction in which the argument is the distance to jump. In this particular format the argument is put into bits 0 through 3 in the instruction. This instruction object will be derived from the base Instruction object since its assemble() function is different from that of Nop (Figure 4).

Note that the object knows its name and has its own base number. In order to

form the complete instruction we must put the value of the argument into the least significant three bytes of the instruction. For example, this line

```
sjump5
```

will be translated into the hexadecimal machine instruction 12005. Since the argument goes into the least significant bits, we need only add it to the base number. In other cases, we could shift the argument to the left so it appears in the correct position.

Therefore, the tasks of the assemble()

```
long assemble(char *sline)
{
    if(the number of args is not equal to 1)
        error message
    if(argument1 > 7 or argument1 < 0)
        error message
    return(base + argument1)
}
```

Figure 5

function will be to check the number of arguments and add the argument to the base to produce the machine instruction. We should also check for illegal values of the argument. Here is the pseudocode for assemble() (Figure 5).

Now that we can assemble one instruction with this format it is easy to add more, just as we did above with NOP. For example, suppose we have a conditional short jump which will occur only if the A

```
class Sjoz : public Sjump {
public:
    Sjoz (void) {strcpy(name, "sjoz"); base = 0x13000;}
};
```

Figure 6

register is 0. We will call it sjoz for short jump on zero. It will use the same format, with its argument (which shows how far to jump) taking the least significant 3 bits of the instruction. Only its base differs from

S J U M P . Therefore, we can derive the object from SJUMP and use the same assemble().

function (Figure 6).

Since we specified only the creator, Sjoz(), the assemble() function and everything else is the same as in the parent, Sjump. The instruction

```
sjoz 5
```

will be assembled as the hex machine instruction 13005. The rest of the Instruction objects will be designed in the same way as our examples.

## Using the Instruction Objects

In order to use the Instruction objects to assemble the instructions in the source code, we first create a NULL-terminated array of pointers to Instructions which includes one of each of the different objects. This is how it's done.

```
Instruction (instr_list[]) = { New Nop,
                             new Pop,
                             new Sjump,
                             new Sjoz,
                             etcetera,
                             NULL };
```

The definition starts by saying that this is to be an array of pointers to Instruction objects, length unspecified. The equal sign indicates that the array will be filled with the following values. Each of the lines following provides another value for the array, and the length of the array will be set to accommodate the total number of values.

The expression "new Nop" creates a new object of the type Nop and returns a pointer to it. This pointer value is assigned to the first element in the array instr\_list(). But how can we assign a pointer to a Nop to a variable which we declared to be a pointer to an Instruction? We can do it because the class Nop is descended from the class Instruction.

Note that the last pointer in the array instr\_list() is set to the value NULL. This is used by the code that makes use of this array to tell when it gets to the end of the array.

Now that we have an array of pointers to all of the different kinds of Instruction objects we can easily use them to assemble the code. We can step through the source file using the Source\_File and Line objects and as we go, use the Field objects to determine whether a line has an instruction. If it does we step through the array of Instructions we created above to find the one whose name is the same as the instruction we are trying to assemble. Here is the function find\_instr() which does that. The function takes a string (the instruction to be assembled) as its argument and returns a pointer to an Instruction object whose name is the same as the argument string (Figure 7).

Note that the while loop ends and the function returns NULL when it reaches the end of instr\_list() because the last pointer in the array instr\_list() was set to NULL which evaluates to 0 and stops the while loop.

The other way for the function to return is to find an Instruction object pointed

```

Instruction *find_instr(const char *opcode)
{
    int i = 0;

    while(instr_list[i]) {
        if(!strcmp(instr_list[i]->name, opcode))
            return(instr_list[i]);
        else
            i++;
    }
    return(NULL);
}

```

Figure 7

to by a member of `instr_list[]` whose name string is the same as the string passed into the parameter "opcode". (Note that `strcmp()` returns 0 when the two strings passed to it are the same.) In this case, it returns the current pointer, `instr_list[i]`, which is a pointer to the Instruction object with the same

```

while(there is another line of source code)
    get the next line of source code
    field2 = second field from line
    inst_pointer = find_instr(field2)
    if(inst_pointer not 0)
        machine_instr = inst_pointer->assemble(line)
        write machine_instr to object file

```

Figure 8

string in its name variable as the string passed into the function.

Using that pointer we can now assemble the line of source code. Here is the pseudocode. Note that in the assembler source code the instructions are always in the second field (Figure 8).

We get a line of source, extract the second field (where the instructions are), and call `find_instr()` with that string. If `find_instr()` finds an Instruction object whose name is the same as the string it returns a pointer to the object, else it returns `NULL` (0). The return value is assigned to the variable `inst_pointer`. If it is not 0 (which is to say the string in field2 is a valid instruction) then the `assemble()` function from the Instruction pointed to by `inst_pointer` is called with the current source code line as a parameter. It returns the machine instruction which corresponds to that line.

Note that the objects the pointers in the array point to are not Instrument objects but objects of classes derived from the Instrument class. Only the member functions declared for the Instrument class are available through a pointer to an Instrument object, even though the derived class has other functions.

This is a different kind of object than the others in the program. It does not represent a data item, but rather a process. It represents the code which assembles an instruction. It is an object of the second kind, derived from the needs of the program rather than the data of the problem.

## The Design Process

It seems to me that the design process can be divided into four main jobs. First, and at the highest level, choose the objects to be used in the program.

Second, specify the relationships between them. There are two parts of this job. The first is specifying the inheritances of one object from another and the inclusions of one object

as an element in another. The second is the declarations of the interfaces (public member functions) through which the objects communicate.

Third, the actual code for all the functions, both public and private, must also be designed.

Fourth, the code which does not belong to any object, the code which creates and uses the objects, must be designed. I find that as I gain experience with Object-Oriented Programming this component of the program becomes smaller. It is possible, of course, to write programs

with (almost) no code except that within the objects. This is customarily the case in Smalltalk, I believe. I still write a small driver program of a page or two in C++. It seems simpler and more straightforward, easier to read and understand. Perhaps this driver will atrophy with time and experience.

## The List Class

Here, as promised, are the declarations for the constant list class (Figure 9).

Note that the constructor for this class takes a pointer to a pointer to a constant character as its argument. The reason for declaring the characters as constants is to make it impossible to change the list once the object is created.

You create an object of this type with a pointer to a null-terminated array of point-

ers to the strings which form the list. First create the array of pointers:

```

const char *pfriends[] = { "Chris",
                           "Roger",
                           "Nancy",
                           "Vicky",
                           "" };

```

and then create the object.

```

Constant_List friends(pfriends);

```

The `member()` function returns 1 if its argument is a member of the list, and 0 if it is not. Thus, you can easily find out if a particular string is in the list or not like this.

```

x = friends.member("Sam"); // x is now 0
y = friends.member("Chris");// y is now 1

```

## Resources

There is a new edition of the C++ Primer by Stanley Lippman. This is the best overall book on the language that I know, and the one I use the most. The new edition is updated to include Release 3 of `cfront` from Bell Labs, which he helped to write. Highly recommended.

## Sources

C++ Primer, by Stanley B. Lippman, Addison-Wesley, 10991. \*

## Continued from Page 6

copy of Windows (if you need it). To further increase its value, many options are available, including the following:

- 80387SX coprocessor
- SIMM memory upgrades
- external floppy disk drive
- extra main battery packs
- a charger that will charge additional main battery packs
- a cigarette lighter charger/adaptor

If you are seriously considering purchasing a MastersPort, its well worth the extra money to get the MastersPort 386SLe. For further consideration, check out the December issue BYTE magazine. They reviewed a MastersPort 386SL (since the SLe version wasn't available at the time of their testing), and you may find their comments helpful. \*

```

#include <string.h>
class Constant_List {
    const char **list;
public:
    Constant_List::Constant_List(const char **lp) {list = lp;}
    int member(const char *item);
};
/* member() takes a string as its parameter and returns TRUE if
the string is duplicated in the list and FALSE if it is not.
*/
int Constant_List::member(const char *token)
{
    for(int i = 0; list[i][0]; i++)
        if(!strcmp(token, list[i]))
            return(1);
    return(0);
}

```

Figure 9



# Getting the Most From Your Computer

## Part 8

**John P. Lewis**  
6 Sexton Cove Road  
Key Largo, FL 33037



Before we get started with an explanation of the code listing in this article, let me address those of you who have just started programming and/or are just beginning an exploration of Turbo Pascal.

The program which is being presented here is fairly sophisticated, but no tricky code is involved. Each procedure and function was created in a straightforward manner (after all, I have to understand it too!). If you have trouble with any part of it, I would suggest that you analyze the code a line at a time, until you have a thorough understanding of each procedure and function. One of my instructors in a company school used to tell us that when faced with the task of eating an elephant, you must eat it a bite at a time — good advice. The same logic applies to any formidable task.

If you have trouble with any of the code presented here, or problems with any facet of programming in Turbo Pascal, feel free to write me at the address given at the beginning of this article. PLEASE include as much information as possible relative to your problem. Also, include a S.A.S.E. if you wish a reply.

Some of my mail reflects a desire for an expanded coverage of my articles, to include Object Oriented Programming. While I can sympathize with those of you who are of the same persuasion, I feel that OOP is a case of extreme overkill in all but a few instances. Those instances are confined to an area of commercial applications where a number of programmers may be involved with different facets of the same project.

I have tried, through a different approach, to accomplish much the same thing that OOP does, and that is through the use of generic code and Turbo Pascal Units, enabling the reuse of code modules and eliminating, to some extent, the creation of additional code to perform similar tasks. As a case in point, the database

program presented here is quite versatile, needing only a few, easily made changes, to re-configure this program to an entirely different job.

My experience with OOP is, admittedly, rather limited, but the reason is quite pertinent. I used an example program, included on one of my distribution disks, to create a sorted directory display. The resulting program was huge, taking a disproportionate amount of disk space. If that was not bad enough, the logic was quite difficult (for me) to follow. Need I say more?

While on the subject of generic code, allow me to inform those of you who have followed this series since its inception to make a correction in one unit. Remove the "Clrscr" statement within the "Boxunit.pas" code. This will enable proper windows management in the programs to follow (including the current presentation). You will notice that I added a "Clrscr" statement to the "Esthetics" procedure within the code presented in Part 7 of this series.

In the last installment, we developed the nucleus for a database program. We also discussed the merits of creating a database which would be compatible with other programs and the methodology needed to accomplish this within the Turbo Pascal environment.

When you have added the code included in this article, you will have a viable database program, capable of searching for and displaying the stored records. I think you will be pleased with the speed and functionality of the program. Another nice feature is the inclusion of a "window" into the database which allows a quick perusal of the contents of one field of each record within the entire database.

The next installment in this series will add some "polish" to the program, allowing the user to sort the database on any

field in addition to expanding on the printing options.

One aspect of this month's presentation which caused me a great deal of concern was the search capability. I was using a program that I had previously created in "C" as a model and I found that Turbo Pascal imposed rather severe limitations in the use of strings (255 characters max.).

The "C" version stored the fields containing the possible "search" keys in one (very large) string. It then parsed this string looking for a match when searching for a record. Obviously this methodology would not work with Pascal due to the limitations alluded to above. What to do?

Not to worry, Turbo Pascal is FAST. My primary consideration was in not sacrificing speed while utilizing the search mode. I was afraid that, by using an array of strings to store the search fields, the Pascal program would be slower than its "C" parent. Au Contraire!

If anything, the Turbo Pascal version is faster than its "C" parent. Quite satisfactory is a rather gross understatement.

Since the "Search" procedure is a little complex and presents a rather important bit of programming logic, we are going to concentrate most of our attention to that fragment of code.

Before actually getting into the listing, let's examine some of the desirable attributes of a database program in the search mode.

In many cases, probably most, the user does not remember a great deal about the record he/she is searching for. Perhaps not even the first part of a name, only the end. Maybe not even the name, but possibly the address. Maybe the record is not a name and address at all, but a part number or a magazine article is the object of our search. In any event, we have to assume a "worst

case" scenario for our problem. After all, the software must be quite powerful and easy to use or the user will be dissatisfied.

Let's look at the listing for "Unitdisp.pas", starting with procedure "Search".

After initializing the variables, our search procedure gets a search "key" from the user and then enters the first of three "while" loops where three parameters (the P^.Next link, # of records is compared to limit, and the Boolean "menu") are monitored.

Next, a case ("find") statement is utilized to establish which field will be used in looking for a likeness of the user entered key. The user may change the search field by employing menu option one, the default field is field2 (Name in our listing example). A second "while" loop is then entered which compares the value of "Index" (character position within "Srch\_String") with the length of our search string (Len2) minus the length of our key, while monitoring the Boolean variable, "Found". Since we are going to parse the variable "Srch\_String" a character at a time, comparing each component with "key[1]" until an exact likeness is found, our current loop ensures that we don't advance past the point where a successful comparison can be made in addition to providing a means of exiting by watching the boolean variable "Found".

A third "while" loop is located just below the second. This loop monitors the variables: Copy\_1 and key[1], as well as the value of index as compared to Len2-Len (if index >= Len2-Len, a successful comparison is no longer possible). When "Copy\_1" (a char within Srch\_String, (Copy\_1:=Copy(Srch\_String,Index,1)) is equal to "key[1]" a successful comparison has been made and we will drop down into a portion of the procedure that will extract a portion of "Srch\_String" (compare:=copy(Srch\_String,Index-1, Len) and then see if our user entered "key" is identical (If compare = key). A match between the two variables will cause the display of the record we have so carefully identified. Notice also that our boolean variable "Found" will be made equal to "True", allowing an exit from the procedure. If "compare" and "key" are unequal, the next record is made available (P:=P^.Next), the number of records is incremented, and the variable index is made equal to 1 in preparation for the perusal of the next record.

A cursory examination of this procedure might lead you to conclude that the computer would take an inordinate amount of time in its execution, don't believe it! You will be amazed at the speed with which the computer can digest a database, looking for a record. My 386Sx will search through over a hundred records in well under a second.

## Listing 1

```
Unit unitdisp;

interface
Procedure Search;
Procedure PrintRec;
Procedure Show_Dat;

implementation
uses crt, unitinpt, printer, boxunit;

Procedure Clear_Down(Lcol, UpRow, Rcol, Brow : Integer);
begin
  window(Lcol, UpRow, Rcol, Brow);      { clears a window defined }
  clrscr;window(1,1,80,25);              { by parameters }
end;

Procedure Display;
Var Col, Len : Integer;
  str : string[10];
Begin
  ch:='0';
  while ch = '0' do begin
    Len:=Length(Field1);Col:=Mark-Len;
    Clrscr;gotoxy (Col,Row1);write(Field1,':',P^.Field1);
    Len:=Length(Field2);Col:=Mark-Len;
    gotoxy (Col,Row2);write(Field2,':',P^.Field2);
    Len:=Length(Field3);Col:=Mark-Len;
    gotoxy (Col,Row3);write(Field3,':',P^.Field3);
    Len:=Length(Field4);Col:=Mark-Len;
    gotoxy (Col,Row4);write(Field4,':',P^.Field4);
    Len:=Length(Field5);Col:=Mark-Len;
    gotoxy (Col,Row5);write(Field5,':',P^.Field5);
    gotoxy (2,17);
    Write('Enter -> (E)dit, (C)ontinue, (P)rint, (M)enu <- your choice ');
    readln(str);ch:=uppercase(str[1]);
    case ch of
      'M':Menu:=True;
      'P':PrintRec;
      E':Begin NotImplemented;Menu:=True;end;
      'C':Ch:=' ' { Return to calling (Search) procedure }
    Else Begin Menu:=True;end;
    end; { End case ch of }
  end;
end;

Procedure search;
Var Len, Len2, Index : Integer;
  Srch_String : String[128];
  Compare : String20;
  Copy_1 : String[1];
Begin
  ndex:=1;Records:=1;Menu:=False;Found:=False;
  clrscr;gotoxy (5,5);
  write('Enter your search "key" '); { Get search key from user }
  Readln(Key);Len:=Length(Key); { establish length of search key }
  P:=Start; { point to first link in list }
  while ((P^.Next <> Nil) and (Records <= Limit)
  and (Menu = False)) do begin{ Outermost Loop }
    Found:=False;
    Case Find of
      1:Begin Srch_String:=P^.Field1;end; { select search field }
      2:Begin Srch_String:=P^.Field2;end { default = field2 }
      3:Begin Srch_String:=P^.Field3;end;
      4:Begin Srch_String:=P^.Field4;end;
      5:Begin Srch_String:=P^.Field5;end;
    end; { end case find of }
    Len2:=Length(Srch_String); { find length of search string }
    while ((Index < Len2-Len) and (Found = False))
    do begin { next outer loop }
      while ((Copy_1 <> Key[1]) and (Index < Len2-Len)) do { inner loop }
        Begin
          Copy_1:=Copy(Srch_String,Index,1); { Parse search string, looking }
          Index:=Index+1; { for a match with first char of key }
        end; { end of "while match", (inner loop), exit to next outer loop }
      { ***** Begin next outer loop ***** }
      Compare:=copy(Srch_String,Index-1,Len); { Create appropriate }
      if compare = Key Then begin { search string & compare w/key }
        Display;
        Found:=True; { show record if match is found }
      end;
      Index:=Index+1;
      end; { end of next outer loop }
      P:=P^.Next;
  end;
end;
```

```

    Records:=Records+1;
    Index:=1;
    end; { end of outermost loop }
    If Found = False then begin
    clrscr;gotoxy(10,8);
    write('Sorry, no match found. Press <Return> for menu. ');
    readln; { make provision for failed search }
    end;
end;

Procedure PrintRec;
Label Copy;
Var Option : String[1];
Begin
    Write(Lst, #27,#21); { Defeat extra line feed on Tandy printer }
    copy:
    Writeln(Lst,#10,#13,P^.Field1,#10,#13,P^.Field2,#10,#13,
    P^.Field3,#10,#13,P^.Field4,#10,#13);
    gotoxy(2,17);clreol;write('Copy? Y/N ');Readln(Option);
    if (Option = 'Y') or (Option = 'y') then goto copy;
end;

Procedure Show_Dat;
Var Col, Row : Integer;
Begin
    Col:=2;Row:=2;
    P:=Start;window(1,1,80,25);Records:=1;
    clrscr;
    while (P^.Next <> Nil) and (Records < Limit) do
    begin
    gotoxy(Col,Row);
    writeln(P^.Field2);
    Col:=Col+LEN2+3;if Col+LEN2 >= 80 then
    begin
        Col:=2;Row:=Row+1;
    end;
    If Row >= 21 then
    begin
        gotoxy(2,23);write('Press return to continue..');
        Readln;
        Row:=2;Col:=2;
        clrscr;
    end;
    P:=P^.Next;Records:=Records+1;
    end;
    gotoxy(2,23);write('End of records, press return to continue ');Readln;
end;
end.

```

## Listing 2

```

Program DbaseToo; { Version 2, modified from part 7 of this series }
uses crt, unitinpt, unitdisp; { UnitInpt from Part 7 }

begin
    OpenFile;
    if Newfile <> True then ReadFile;Find:=2;Modified:=False;

    While Option <> 6 do begin
    clrscr;esthetics; { added clrscr for win ver }
    gotoxy(5,2);write('Please enter...');
    gotoxy(20,4);Write('1. Re-define Search field, default = ',Field2);
    gotoxy(20,6);write('2. Search for/display/edit record. ');
    gotoxy(20,8);write('3. View window into database. ');
    gotoxy(20,10);write('4. Create new record. ');
    gotoxy(20,12);write('5. Sort Records. ');
    gotoxy(20,14);write('6. Exit to operating system');
    gotoxy(17,16);write('the number of your choice. ');
    Readln(option);
    case option of
    1:begin Clrscr;gotoxy(5,2);write('Please enter...');
        gotoxy(20,4);write('1. ',Field1);
        gotoxy(20,6);Write('2. ',Field2, '(Default). ');
        gotoxy(20,8);Write('3. ',Field3);
        gotoxy(20,10);write('4. ',Field4);
        gotoxy(20,12);Write('5. ',Field5);
        gotoxy(10,14);
        write('the number corresponding to the desired search field. ');
        readln(Find);end;
    2:begin Search;P:=Start;End;
    3:begin if Newfile <> True then show_dat else
        begin clrscr;gotoxy(6,8);
        write('No data file exists, press enter to return to menu ');

```

As you will notice, I have made use of a large number of comments within this procedure. A good practice at any time but VERY important when the logic is a little "deep". Many times I have gone back to a procedure or function to utilize the code fragment in another program, only to find that I don't, at first blush, understand my own code! Hence the liberal comments, they make life much easier.

O.K., if you understand "Procedure Search" you have my permission to go out and buy a programmer's hat, you earned it! Actually, I think you will agree, it's not all that difficult if you tackle one line at a time.

Let's take a look at "Procedure PrintRec" next. There is nothing very complex here, but I did take the liberty of utilizing another endearing facet of Turbo Pascal as a bit of a short cut. The entire record is sent to the printer with one statement. Since we need to send a carriage return, line feed sequence to the printer for each individual line (field) of the record, the code is inserted or "tacked on" through the use of the symbol "#". The statement "#10,#13" (included within a writeln command) sends a line feed, carriage return sequence to the computer. Before leaving "PrintRec" notice the "goto" command (considered by some as a no-no) was utilized to allow copies of an individual record to be made. We will utilize this procedure (Printrec) in two different ways. The first will be from "Procedure Display", where the user may opt to print one record. The second will be from the main menu where the user may choose to print the entire database (as in the case of a mailing list).

One, nearly indispensable, attribute of a database program should be some means of taking a peek at the database without accessing an individual record. If we had a "window" to look through at one field of each record, we might save ourselves a lot of grief. Procedure "Show\_Dat" does just that. It displays "Field2" of each record within the database. If the database is huge, you might want to provide some means of escape from this routine. My needs are much more modest and such a device was not deemed necessary.

Only one "while" loop is used in Procedure "Show\_Dat" and it is used to monitor the last record in the database. We must ensure that we don't try to print a record where none exists, the results are not pretty! The rest of the code merely prints "Field2" of each record, in two columns down the screen, pausing near the bottom to allow the user to catch up and then allowing the display of yet another screen of data. There should be no mysteries in this part of the program.

The changes to "Dbasetoo.pas" are for the most part, simply implementations of the code we have reviewed thus far; however, menu option one, which allows

the user to choose the field for purposes of record retrieval, is found to be coded within the main body of our program (DbaseToo.pas). It is merely an expansion of the case statement (case option of) found below the code fragment used to display the main menu. Option 1 first clears the screen and then displays a second, or sub-menu of the choices available for changing the search field to a different part of the record, enhancing the search capability a great deal. Notice, near the top of "DbaseToo.Pas", the statement: Find:=2. This establishes the default value of "Find" which in our code presentation, is the "Name" field. We located this bit of code above the while statement which, in turn, ensures that we stay "glued" to the menu until the user chooses option six (exit to operating system), while option two remains the default.

We have, thus far in the series, accu-

```

readln;end;end;
4:begin P:=Start;
  While P*.Next <> Nil do P:=-P*.Next; { Find end of linked list }
  create_record;NewFile:= False;end; { tack entry on end }
5:begin NotImplemented;end;
end; { end case option of }
end; { end of while }
Option:=0;If modified = True then Write_File;
Textcolor(LightGray);TextBackGround(Black);
window(1,1,80,25);clrscr;
end.

```

mulated quite a bit of Turbo Pascal source code. I hope you feel that the investment in time has been quite profitable. For those of our readers who get a nauseous feeling in the pit of their stomach at the prospect of entering a great deal of source code from the keyboard, or for whatever reason, would like to have a copy of the listings thus far published. I will provide you with a copy, on disk, of all the source code as well as a compilation (executable version) of same, if you send me a FORMATTED disk and

\$7.50 (for postage and handling).

I (actually, my computer) am able to write to disks in the following format: 3-1/2 - 720 Kb, 3-1/2 - 1.44 Mb, 5-1/4 - 360 Kb and 5-1/4 1.2 Mb. That about covers the spectrum I believe, except for IBM's latest offering of 3 1/2 - 2.88 Mb.

Take good care of those programmer's hats, you're going to need them in Part 9 of this series. \*

Zenith Data Systems  
Power Supplies  
Complete Board Assemblies

## SURPLUS COMPUTER PARTS

20 Meg. Hard Drives  
From 50.00

- 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	Ea. 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	75.00
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 SupersPort	300.00
Laptop Replacement Batteries	From 6.00

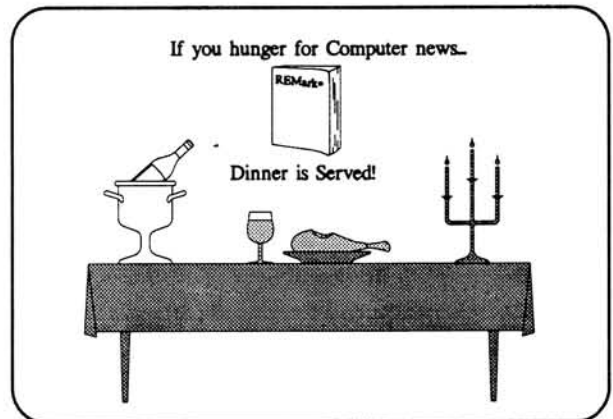
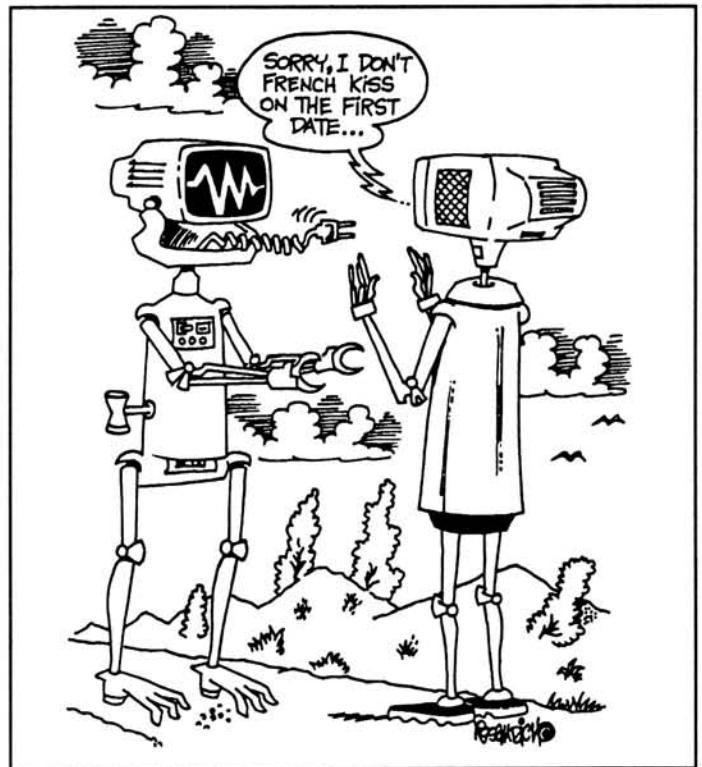
Many Other Zenith Data Systems Salvage Parts Available  
All Surplus/Salvage Sold AS IS - WHERE IS  
- NO GUARANTEE -

No Catalog - Please Call for Quotes  
We Ship U.P.S.-C.O.D. (ONLY) Sorry - No Foreign Shipments

## 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)



---

---

# Coherent

## Unix on a Budget

**Bill Wittig**  
**11215 Birmingham Court**  
**St. Louis, MO 63138**

For quite a while now, one of the things on my "I'd like to learn" list has been the Unix operating system. I keep reading articles that claim that Unix is fast becoming the operating system of choice for PC's. PC's are becoming powerful enough to use its open architecture and multi-user/multi-tasking capabilities and more of the "PC" software is becoming available for Unix. Unix has been the operating system of choice on workstations for a long time. Unfortunately, the price of Unix - even on a PC - has been prohibitive. You can easily spend over \$1000 for just a basic installation, which probably doesn't include any tools - compilers, editors, etc. I just couldn't justify the cost to satisfy my curiosity.

Then I ran across an ad from the Mark Williams Company (MWC) for Coherent which seemed to provide a solution to this dilemma. Coherent provides a full featured clone of the Unix operating system at an amazing price - \$99.95 - and claims to be powerful, inexpensive, and easy to install. To good to be true?

Intrigued, I purchased version 3.2 of Coherent direct from Mark Williams Company. Since version 3.2 had just been released, they said it would take about 3 weeks to arrive - I was surprised a week later when the Coherent package showed up at the front door. This began a pleasant experience - I've been very impressed with both Coherent and the Mark Williams Company.

There has to be a catch, right? Well, Coherent is not the AT&T developed and licensed Unix and it's not the latest revision - System V.4. It's modelled after an earlier Unix release. Despite this, there aren't many differences in the basic operating system, and MWC provides a complete set of tools and utilities which give you many of the

capabilities of the latest Unix. In addition, the included C compiler allows you to "roll your own" programs and gives you the ability to use the thousands of public domain programs available for Unix. You would have a hard time finding the individual components, let alone the whole package, for the price of Coherent.

### Coherent and DOS

Installing Coherent on your PC doesn't mean that you have to give up DOS. DOS and Coherent co-exist very happily side-by-side on the same hard disk; and you can easily select which operating system you want to use during the boot process. During installation, Coherent replaces the boot sector on your hard drive with the Mark Williams master bootstrap program. This bootstrap program allows you to select which partition on the hard drive to boot from. The standard installation leaves DOS on one partition and put Coherent on the other. You then choose which operating system you want to make the default (DOS or Unix). The other is selected by pressing a key during the boot process.

### Hardware Requirements

Coherent requires an AT compatible PC and runs in protected mode. At least 640K of RAM is required (more is better) along with 10 MB of hard disk space (20 MB is probably more realistic). A high density floppy disk drive as drive A: is also required along with a 16-bit (AT) hard drive controller. Coherent works with most standard hard disk drive types, and some SCSI controllers.

MWC has an extensive list of hardware that Coherent has been tested on, including most Heath/Zenith computers. You can call them (see the phone number

at the end of this article) to check on your specific setup. In addition, I've found the technical support to be very knowledgeable and helpful. Coherent does NOT work with some common hardware - IBM PS/1 or PS/2 computers, all XT (8 bit) disk controllers, and some older Zenith Z449 video cards among others, so you should really check with MWC to be sure your particular hardware is supported before ordering Coherent.

### Installation

Coherent comes on five high-density 5.25 or 3.5 inch floppy disks.

The Coherent installation process starts by booting a minimal copy of itself from a floppy in drive A: - make sure you order the proper disk size! It then leads you through the installation process. The main steps in installing Coherent are: 1) Write the MWC master bootstrap program to the hard disk boot sector. 2) Partition the hard disk to make room for Coherent (if necessary). 3) Create a partition dedicated to Coherent. 4) Create the Coherent file system and copy a bootable copy of Coherent to the hard drive. 5) Reboot and install the rest of the Coherent system, utilities, and examples.

Since the installation procedure modifies the hard drive boot sector and partition table, it is essential that you have a backup of the entire hard drive before installing Coherent (unless it is empty). My experience has been that as long as the Coherent partition is at the end of the disk and doesn't overlap any DOS files, the DOS file system will be smaller, but undamaged. Don't trust it! Make a current backup before starting the installation. If you don't need it you've only lost a few minutes. The installation process also lets you back up

cc	C compiler
as	8086 assembler
make	Program maintenance tool
mail	Electronic mail (local to the computer)
MicroEMACS	Coherent version of the EMACS editor
ed, sed, vi	Standard Unix editor clones
awk	pattern scanning language
lex	Lexical analyzer for processing strings
yacc	"Yet another compiler compiler"
nroff, troff	Typesetting utilities
bc	Desktop calculator
UUCP	Remote communications utility.

Figure 1

the old boot sector to a floppy before its replaced — also a good idea in case something goes wrong.

The installation process is extremely well documented, both on screen and in the accompanying manual, with all of the options explained. Although MWC claims Coherent can be installed in 20 minutes, it took me about an hour — mainly due to the time required to read the extensive documentation and consider all of the options at each step. After installing all of the utilities and manual pages (on-line documentation), Coherent took a little over 10 MB of the 30 MB of disk space I allocated to its partition.

There was one bug in the installation process — while uncompressing the manual pages, several disk errors were reported. When Coherent was rebooted, it detected a problem with the file system and offered to automatically fix it. Allowing Coherent to fix the problem cleared up the error and it hasn't recurred. A call to MWC tech support confirmed that this was a known, but harmless, bug.

After completing the installation, Coherent offers to let you set passwords for all of the system accounts and create new user accounts for as many people as you desire. At this point, you are truly running a multiuser system!

### Documentation

MWC supplies two manuals with Coherent — a short installation guide and a paper bound 1200 page manual. The manual is organized into a tutorial section covering an Introduction to Coherent, an Administrators guide, Introductions to the awk, lex, and yacc utilities, the Editors (ed, sed, vi, and MicroEMACS), the C language and compiler, UUCP communications utilities, the Bourne shell, and the nroff and troff Text formatters. The tutorials provide an in-depth introduction to Unix in general and Coherent in particular.

The second (and largest) half of the manual is the Lexicon. The Lexicon concept of documentation was new to me. It

contains descriptions of all of Coherent's commands and functions — along with general information — organized alphabetically. If you need to find out about a topic, you simply look it up in the

Lexicon. The Lexicon will also refer you to related topics via its cross reference, allowing you to really explore a subject. For example, to find out how to connect a cable to the serial port of the computer, you would look under "RS-232" which shows the needed pinouts and points you to the sections detailing how to set up the software to drive the port. All of the Lexicon topics are also on-line and can be accessed with the "man" command while running Coherent.

After a little getting used to, the Lexicon is a very efficient way of organizing documentation — I found that it allowed me to easily find out about almost any

topic. To start using the Lexicon you look up the entry for Lexicon. From there, the cross reference lists will lead you through the entire Lexicon, covering all of Coherent. It's a very logical, easy to use method for organizing a large body of documentation.

### Features

Coherent includes all of the standard operating system commands along with the wealth of utility programs that come with Unix. In addition, many utility programs and tools which usually cost extra are included. See Figure 1.

In addition, Coherent includes a *dos* utility which allows you to read and write to DOS disks — floppy disks and hard drives — giving you easy access to your DOS files from Coherent. *dos* also translates between the DOS CR,LF style end-of-line markers and the Unix style CR newlines.

### Running Coherent

To boot Coherent you must hold down the number key (at the top of the keyboard) which corresponds to the partition you want to boot while the system is trying to boot from the A: drive. This method is fairly easy to use, but it does force you to pay attention while the computer boots. The alternative is to make Coherent your default operating system, which makes booting DOS the less convenient mode. I have

advent.tar.Z	A text adventure game.
almanac.tar.Z	This program calculates the ascension and declination of the Sun, Moon and the 3 major planets when given the calendar date and time. Also calculates the azimuth and altitude of the Sun, Moon and 8 planets.
booz.tar.Z	Portable archiving utility.
calls.tar.Z	Utility to trace hierarchy of called functions in C programs.
chess.tar.Z	A chess game for Coherent.
dumpscreens.c.Z	Screen dump utility for COHERENT.
gnugo.tar.Z	Pd game based upon a game known as 'go'.
gnutar.tar.Z	TAR utility for archiving files. Format can be read by other Unix systems. Requires a libndir.a library. This can be found in the coh-clam.tar.Z file.
hotel.tar.Z	A board game, similar in some respects to monopoly, involving the purchasing and bartering of hotels.
kermit.tar.Z	Ckermit. A pd kermit which sports several enhanced features, too numerous to list here!
lharc.tar.Z	Public domain archiver.
menubar.tar.Z	Utility to be incorporated into c programs for selecting menu items from a window.
prolog.tar.Z	Prolog language ported to COHERENT. Examples and tutorials included.
rcs.tar.Z	Revision control system ported to COHERENT!
sc.tar.Z	COHERENT's first spreadsheet!
tetris.tar.Z	A clone of the popular tetris game for COHERENT.
unzip.tar.Z	Unzips files compressed with zip.
xcmalt.tar.Z	Communications program which can use scripts to automate login sessions to services like CompuServe, etc... Supports xmodem and B+ protocols.
zmodem.tar.Z	Popular file transfer utilities.

Figure 2

to admit I still use DOS most of the time, since most of the programs I'm used to (and have purchased) are DOS programs.

After booting, Coherent checks the file system for consistency and then prints a *login*: prompt — Welcome to multiuser computing! Coherent has full user access and file protection capabilities allowing multiple users to use the system and store files without interfering with each other either by accident or on purpose. Files (and directories) are controlled at three levels of access — world (i.e. everybody), group (as defined by the system manager), and owner. Each of these three levels can be set to have read, write, and/or execute privileges. In addition, you can activate an accounting function which allows you to track how and when the system is used. If you've enabled passwords, Coherent will request a password after the user ID is entered to complete the login process.

On the surface, Coherent is very similar to DOS. The file structure is similar, although forward slashes are used instead of back slashes to separate directories. Unix was the originator of short (2 to 3

can be connected to your PC and used by other people or you can run more than one task at a time. This idea finally sunk in while I was watching the floppy disks format. I realized that I didn't need to waste my time watching the disk lights flash. I connected my MinisPORT to the PC's COM1 port using a null modem cable, fired up ProComm on the MinisPORT, logged in, and started working on my next job. The only tricky part was setting up the serial port, and the Lexicon made that pretty easy.

### Backing Up the Coherent Disks

Since Coherent is a foreign operating system, its floppy disks can't be backed up until after it's installed. I had to dig a bit to find the procedure for making backups, but I finally found it under the boot topic in the Lexicon and in a file on the MWC bulletin board. I created a short shell script (like a batch file) named *.dskcpy* which copies the distribution disk to the hard drive, formats a new floppy, and then copies the distribution disk data onto the new floppy. The script in Figure 3 works for

10 times as much (which should be no surprise). Programs under Coherent are limited to being 64KB in size — i.e., only the 80x86 small memory model is supported. MWC is apparently working on an update to fix this limitation, but they are not giving a release date.

Also, the Coherent dos command doesn't recognize the extended size disk partition capability introduced with DOS 4.01. Coherent cannot access a large hard drive set up as a single partition. This is easily overcome by putting the DOS files on a floppy and reading them into Coherent from the floppy disk. It's not as convenient as reading them directly from the hard disk, but it is workable.

Coherent doesn't include any networking capabilities — NFS, FTP, TCP/IP, etc. Public domain software is available for much of these, notably from Clarkson University and Brigham Young University, if you're willing to work at it.

All in all, I feel that Coherent is a very complete and workable personal clone of Unix, and well worth its price. If you've been at all curious about the current Unix hype Coherent gives you an extremely affordable starting point — and who knows, once you have Coherent you may never need anything else!

Coherent is available directly from:

Mark Williams Company  
60 Revere Drive  
Northbrook, IL 60062-9620  
(708) 291-6700 ✻

```
echo 'Put Master in Drive A (HD) and press Enter:
```

```
\007\c' read Dummy           ! Wait for Enter
export DEV=/dev/fha0         ! Floppy drive device name
export BS=30b                ! Size of data on disk
dd if=$DEV of=dskmstr bs=$BS count=80    ! Copy floppy to hard disk
```

```
echo 'Put blank floppy in Drive A (HD) and press Enter:
```

```
\007\c' read dummy           ! Wait for Enter
/etc/fdformat $DEV           ! Format new floppy
dd if=dskmstr of=$DEV bs=$BS count=80    ! Copy data to new floppy

rm dskmstr                   ! Delete temporary file
```

Figure 3

letter) command names — possibly to save typing and transmission time over slow data links compared to today. Coherent includes print spooling to allow you to keep working while a file prints and also includes the parallel of batch files, called shell scripts.

A large amount of free software is available for Coherent from the MWC Bulletin Board System. The table in Figure 2 shows some of the files available.

Since Coherent is multiuser and also uses disk buffering, it can't be shut down as easily as a DOS system. Simply turning off the power or rebooting guarantees problems of lost data or mad users. In order to halt Coherent safely you must log in as the root (i.e. the system administrator) and run the *shutdown* program. After a few seconds a # prompt appears and you type *sync*. Another # prompt appears after a few seconds. The system is now halted and you can reboot or turn the power off safely.

One of the new concepts PC users running Coherent must get used to is the multiuser capabilities. Remote terminals

5-1/4" HD floppies only.

In order to use this script with 3-1/2" HD floppies the *fha0* must be changed to *fva0* and the *30b* to *36b*. In case you're not familiar with Unix terminology (I wasn't) the *f* refers to a floppy, the *0* indicates the first floppy (drive A:) and the other two letters indicate the number of tracks and sectors per track on the disk.

The script is executed by typing:

```
sh .dskcpy.
```

### Support

Mark Williams Company has a technical support telephone line available for questions, as well as an electronic mail system available to all Coherent users via modem. In addition, the MWCBBBS bulletin board system carries news, technical questions and answers, and a large library of free software for Coherent.

### Limitations

Coherent isn't perfect — there are some limitations and quirks which make it not quite as good as the packages costing

# FBE

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

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

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

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

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

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

### Call Or Write For Additional Information

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

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

# QUIKDATA - 16 YEARS OF H/Z SUPPORT!

YOUR H/Z ENHANCEMENT EXPERTS!

## ACCELERATE YOUR PC/XT/AT!

For your H/Z241, 248 and 386/16 we have the new direct replacement **WESTERN IMAGING Z-33 ZUNI MOTHERBOARD** that's just loaded with features. 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** - \$849 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** - \$119

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

**Z605-1** - \$125

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

**ZA3600ME** - \$89

**ZA3800MK** - \$275 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** - \$569 80 meg with Segate ST-4096 full size drive.

**ST-4096** - \$495 Bare drive only

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

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

**IDECON** - \$89 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.

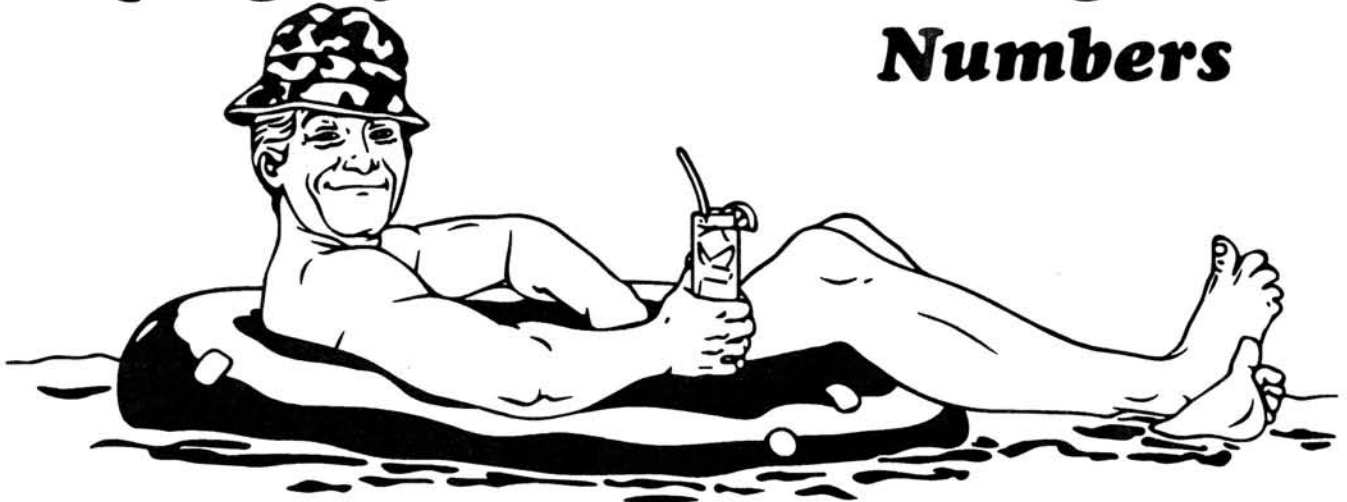
Call or write in to place your order, inquire about any products, or request our free no obligation catalog. VISA and Master Card accepted, pick up 2% S&H. We also ship UPS COD and accept purchase orders to rated firms (add 5% to all items for POs). All orders add \$5 S&H. Phone hours: 9AM-4:30PM Mon-Thu,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



# Keeping Afloat with Floating Point Numbers



David W. Lind  
Rural Route 1 Box 3114  
Bar Harbor, Maine 04609

## Introduction

The microprocessors used in the IBM PC or compatible computers are designed to perform arithmetic operations with Binary digiTS (bits) and Binary Coded Decimal (BCD) numbers. Although one can assume a binary point at some position in a string of bits or a decimal point in a string of BCD characters, the use of these strings for many scientific or engineering applications becomes unwieldy. Very early in the history of computer science, this problem was recognized. As a result, computer scientists developed a number format which made complex computations easier and more efficient. The decimal point in the results of these computations does not occupy a predetermined position as with "fixed point" numbers like accountants might use, instead they can occupy any position in the number as the program at the end of this article demonstrates.

Computer scientists also came to realize that computations with floating point numbers used a great deal of processor time. Thus, floating point processors were developed. The computer's main processor, usually called the "Central Processing Unit (CPU)," passes the floating point operations to the floating point processor if one is available. The floating point processor in IBM PCs or compatibles is called a "coprocessor." Coprocessors are normally purchased as an option, although nearly all computers have designated places where coprocessors may be installed. A notable exception is a computer using an i486 DX microprocessor or equivalent which has the coprocessor functions built into the microprocessor.

It is the existence of coprocessors which makes the use of the floating point

number format attractive. One can write software routines which perform floating point operations using the microprocessor alone; but such routines are lengthy, complex, and inefficient. If a coprocessor is not installed in a computer, it is often best to use fixed point format vice floating point format. Floating point format is generally better than fixed point format if the computer has a coprocessor.

Most computer language compilers provide the option of generating coprocessor code or emulating such code if a coprocessor is not installed. Unfortunately, one has little control over how the coprocessor code is implemented which can result in inefficiency. Those of us who actually prefer to program in assembly language have much better control, but most assemblers have no input or output routines for the floating point numbers. The input and output assembly language procedures listed at the end of this article help fill this void.

This article discusses the various types of number formats and how they relate to floating point numbers. The use of the coprocessor is also discussed. The article ends with a discussion of the listed program which performs arithmetic operations and finds the square root of floating point numbers precise to about fourteen digits.

The most important feature of this article is that it provides enough information to allow assembly language programmers to program coprocessors quickly and efficiently. It also provides information about coprocessors that other programmers will find useful. The listed input and output procedures are adequate for most applications and can be easily modified to accom-

modate more stringent requirements. Coprocessor programming can be enjoyable and very rewarding. Keeping afloat in a sea of numbers using a floating point processor is not as difficult as it seems.

## Number Formats

A typical coprocessor can process binary integers, packed BCD numbers, and floating point numbers. One must be familiar with these number formats to use the coprocessor effectively. In the following discussion, refer to Figure 1 for graphic representations and comparisons of the various number formats.

## Binary Integers

A binary integer is a number consisting of binary digits which the microprocessor processes as an integer. An example is shown in Figure 2.

Here the number 79 is represented in its bit string form, expanded by powers of 2, written as a decimal number, and finally as a hexadecimal number. The hexadecimal number is the representation normally displayed by computer memory dump programs or debug programs.

Binary integer strings may be of various lengths. Word integers are 16 bits long. The example is a word integer. Short integers have 32 bits. Long integers use 64 bits. Other integer string lengths are possible, but are not recognized by the coprocessor.

## Packed BCD Integers

Packed BCD integers use four-bit codes to represent decimal numbers. These codes are:

$0000_2 = 0_{10}$   
 $0001_2 = 1_{10}$   
 $0010_2 = 2_{10}$

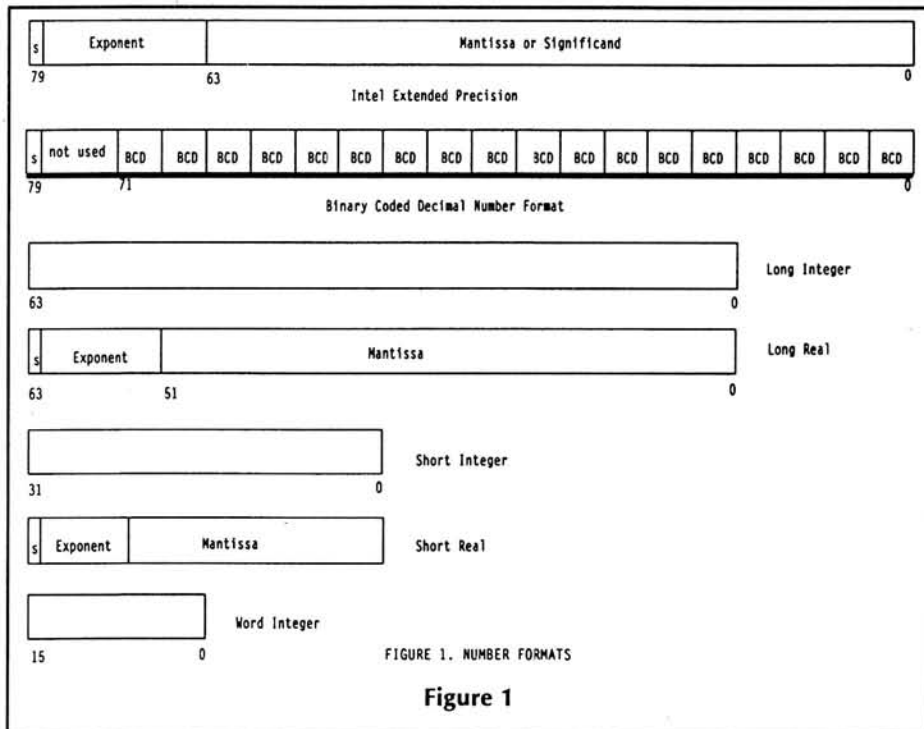


FIGURE 1. NUMBER FORMATS

Figure 1

- 0011<sub>2</sub> = 3<sub>10</sub>
- 0100<sub>2</sub> = 4<sub>10</sub>
- 0101<sub>2</sub> = 5<sub>10</sub>
- 0110<sub>2</sub> = 6<sub>10</sub>
- 0111<sub>2</sub> = 7<sub>10</sub>
- 1000<sub>2</sub> = 8<sub>10</sub>
- 1001<sub>2</sub> = 9<sub>10</sub>

Note that not all combinations of four bits are used. These unused combinations are not numbers in the BCD format. The BCD codes fit into half a byte, a bit string length of 4 bits called a "nibble." A packed BCD number has two coded decimal numbers per byte (one per nibble). An unpacked BCD number uses only one coded decimal number per byte loaded into the low bits nibble.

Packed BCD numbers are strings of 80 bits. The highest or most significant bit is the sign bit. If this bit is set, the number is negative. Otherwise, the number is positive. The next highest seven bits are not used. The remaining 72 bits contain the coded decimals. At one per nibble, this format can represent up to 18 decimal digits or an integer with an absolute value up to 999,999,999,999,999,999 (999 quadrillion +). Packed BCD numbers may also be represented by a hexadecimal string which, not coincidentally, looks like a decimal string. For example the number 123,456 in packed BCD format displayed in a hexa-

$$\begin{aligned}
 01001111 &= (0 \times 2^7) + (1 \times 2^6) + (0 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) \\
 &\quad + (1 \times 2^1) + (1 \times 2^0) \\
 &= 79_{10} \text{ or } 79 \\
 &= 4F_{16} \text{ or } 4Fh
 \end{aligned}$$

Figure 2

decimal dump would appear as 00000000000000123456h. The negative number -123,456 would appear as 80000000000000123456h. The hexadecimal numerals A, B, C, D, E, and F should never appear in the dump of a BCD number, packed or unpacked. If they do, the number is not a proper BCD number and the coprocessor will not process it correctly.

### Floating Point Numbers

Floating point numbers have three parts. The highest or most significant bit is the sign bit. As with BCD numbers, the number is negative if this bit is set and positive otherwise. The next group of high bits represent the exponent, more about the exponent later. The remaining bits represent the "mantissa" or "significand."

The mantissa represents the digits in the floating point number in binary form. A binary point, sometimes called a "radix point," is assumed to be before the mantissa. Therefore, the mantissa is a binary fraction where the highest bit has the value  $2^{-1} = 0.5$ , the next highest bit has the value  $2^{-2} = 0.25$ , etc. To make this format even more confusing, the mantissa is assumed to have a 1 before the binary point, making the actual value of the mantissa 1.+ (that is, a number that is greater than one, but less than 2). Any number within the range of the

floating point number can be represented by dividing or multiplying the number by 2 until the result is within the range of the mantissa (greater than or equal to 1 and less than 2). Of course,

the number of times the original number was multiplied or divided by 2 must be recorded somewhere. That's the part that the exponent plays.

The exponent indicates the number of times the mantissa must be multiplied or divided by 2 to obtain the number that the floating point number represents. To add to the confusion that floating point numbers already cause, the exponent is "biased." A biased exponent is formed by adding a number called the "bias" to the original mathematical exponent such that the most negative exponent is represented by zero. An original mathematical exponent of zero would have a biased exponent value half way between the smallest biased exponent and the largest biased exponent. Therefore, the biased exponent can be treated as an unsigned integer. The value of the bias depends upon the length in bits of the exponent field.

An exponent representation less than the bias, when subtracted from the bias, produces a number which indicates how many times the mantissa must be multiplied by two to obtain the represented number. Subtracting the bias from an exponent representation greater than the bias produces a number which indicates how many times the mantissa must be multiplied by two to obtain the represented number. Multiplying or dividing by two is equivalent to moving the binary point in the mantissa to the right or left one place respectively for each multiplication or division by two. This concept is the binary equivalent to scientific notation.

There are two ordinary types of floating point numbers specified by the Institute for Electrical and Electronic Engineers (IEEE) and which apply to coprocessors. The first is the "short real" or "single precision number." The exponent in this format has eight bits and the mantissa has 23 bits, excluding the implied bit, for a total length of 32 bits or two words. The bias is 127. Therefore, the largest number that can be completely represented (i.e. not rounded) by this format is  $2^{24}-1$  or 16,777,215 (remember that the implied bit must be included). Note that the exponent, which is unsigned, can have values from 0 to 255. The exponent numbers 0 and 255 are reserved for special representations called denormal real numbers and NaNs. Thus, the maximum number of times the mantissa may be multiplied by two to find the number represented by this format is  $254-127=127$  times. The maximum number of times the mantissa may be divided by two to find the number represented is  $127-1=126$  times. If X is a positive real number represented in this floating point format, then

$$\begin{aligned}
 16777216 \times 2^{-149} < X < 16777216 \times 2^{104} \\
 \text{or } 2.35 \times 10^{-38} < X < 3.4 \times 10^{38}
 \end{aligned}$$

So, what about 0.0 or positive numbers smaller than  $2.35 \times 10^{-38}$ ? That's where



that should be followed when using the coprocessor.

The first instruction to the coprocessor should be FSAVE. This instruction has two important functions. First, it stores the state of the coprocessor in a 94 or 108 byte string in the data segment. The 108 byte string is needed when 32-bit addressing is used. When the program is finished with the coprocessor, the original state can then be restored using the FRSTOR instruction. This process is important if some other program was interrupted and was using the coprocessor because the coprocessor will be returned to the state it had before the current program was called. Remember, the operating system does not monitor the coprocessor and will not save data for the coprocessor as it will for other devices. The second function of the FSAVE instruction is to initialize the coprocessor, that is, to set the device to its default values. The FINIT instruction also sets the coprocessor to default values and should be used if the coprocessor must be reset during program execution.

Because coprocessors operate independently and take a considerable amount of time to perform many functions, the microprocessor and coprocessor may need to suspend their operations until the microprocessor or coprocessor is finished. Newer microprocessors and coprocessors, the 80286 and 80386 series or newer, do not require delay of coprocessor functions. Generally, the only time it is necessary to delay microprocessor operations is if the microprocessor will use the results of the coprocessor operations. The instruction that suspends microprocessor and coprocessor operation is FWAIT or WAIT. These instructions are equivalent and essentially cause the microprocessor to stop issuing further instructions until the currently executing instruction is complete. It does not matter if that executing instruction is a microprocessor or coprocessor instruction because the microprocessor controls the instruction queue. If a program is assembled for an 8087 or similar coprocessor, the assembler will normally insert the wait instruction after a microprocessor instruction and preceding a coprocessor instruction. However, one must use the FWAIT or WAIT instruction explicitly when the microprocessor will use the result of a current coprocessor calculation. In this case, the FWAIT or WAIT instruction must be used immediately after the coprocessor instruction and before the affected microprocessor instructions for all coprocessors and microprocessors. The assembler will not insert the wait instruction in this case. Keep in mind that if no wait instructions are used before coprocessor instructions, the program will only execute properly on newer computers.

Some instructions have no wait ver-

sions. No wait means that the instruction does not check for numeric-error exceptions. The use of these instructions will save time because exception checks are generally not necessary in these cases. When no wait instructions are used, they should not be preceded by a WAIT or FWAIT instruction. However, don't confuse no wait instructions with WAIT instructions. The former applies to error exceptions and the latter to instruction queuing, quite different concepts.

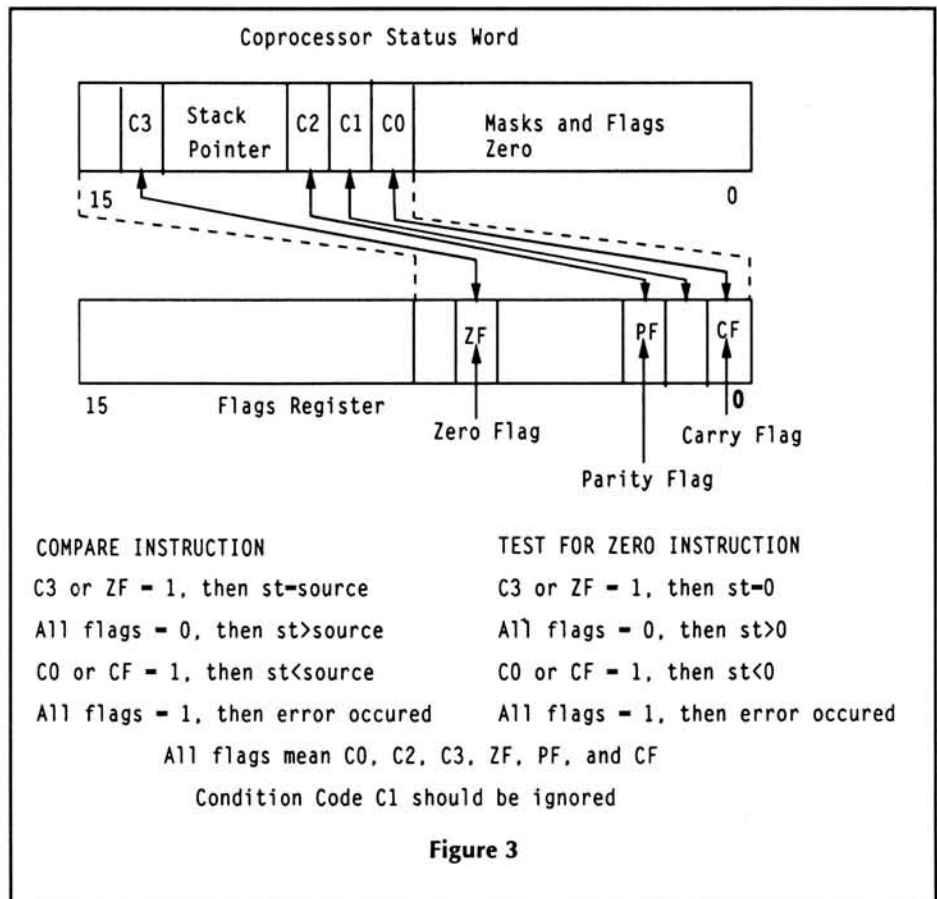
When a coprocessor instruction moves data to and from memory or uses memory information in an operation, the instruction determines how that data is treated. For example, FLD treats the memory item as a real number, FILD loads an integer from memory, and FBLD treats the number as BCD format. Therefore, the same memory location can be used for different types of numbers so long as the assembler permits the use of that memory location for that particular type of data.

When data is in the coprocessor, it is converted to a format that Intel Corporation calls "extended precision." These numbers are 80-bits long. The highest bit is the sign bit. The exponent uses fifteen bits with a bias of 16382. The mantissa is 64 bits long with a major departure from the IEEE format, the single integer bit is explicit. It is not necessary to be concerned with this format unless one wishes to use or display the

extended precision format. Such a use is beyond the scope of this article.

The coprocessor has a stack of eight registers numbered from st(0) to st(7) in addition to control registers. The register st(0) is also designated st. One should try to perform as many operations as possible in the coprocessor vice between the coprocessor and memory to make the operations as short as possible.

Comparisons of floating point numbers using the coprocessor is a bit unusual. Results of such comparisons set condition codes in the coprocessor status word. To check the status word, it must be transferred to memory with an FSTSW instruction and examined. It is best to store upper byte of the status word in the lower byte of the flags register of the microprocessor using the SAHF (store ah into the flags register) instruction and branch with the jump instructions. Only the carry, parity, and zero flags are affected. Therefore, only jump instructions based on these flags can be used. See Figure 3 for a graphic comparison of the status word and flag word. The program at the end of this article illustrates the process of comparison using the FCOM (compare real numbers) instruction. Figure 3 shows what the flags mean for the comparison and test for zero instructions. An error will occur during a comparison operation if one of the numbers is a NaN.



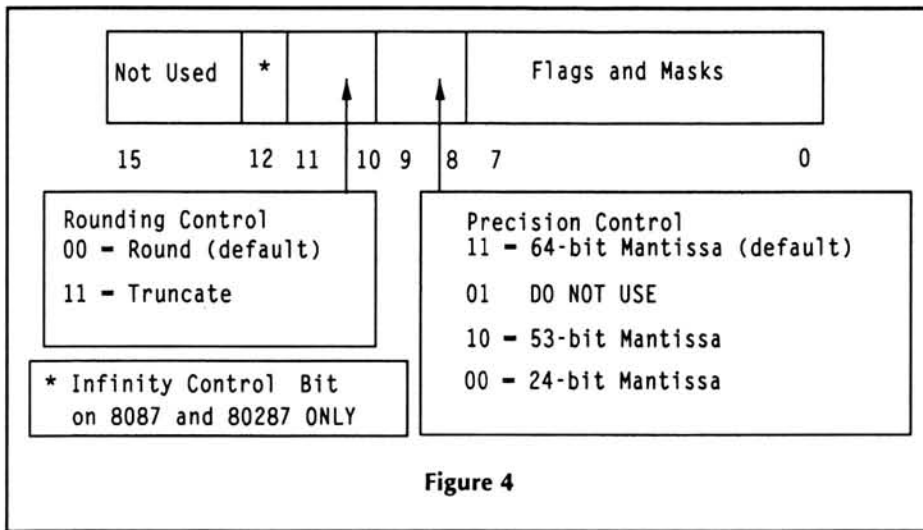


Figure 4

The control word in the coprocessor is important for rounding and precision control. By default, Intel coprocessors round to the nearest number or to the even number if the number ends in five. The program at the end of this article changes this default to truncation. The default for precision control is a 64-bit mantissa. The mantissa can be changed to 53 or 24 bits. More control word information is shown in Figure 4. Other functions of the control word are beyond the scope of this article.

Given this information, the reader should be able to use a coprocessor fairly well. The best way to learn is to do. The program listed at the end of this article illustrates the use of the major functions of the coprocessor and helps to tie together coprocessor functions.

### The Calculator Program

The calculator program at the end of this article performs the operations of addition, subtraction, multiplication, division, and square root using a floating point processor. Two numbers are entered then the operation symbol is entered to perform the operation. To take the square root of a number, the number is entered followed by another entry (blank entry is fine) and then a period is entered alone. The other operations use the familiar +, -, \*, and / symbols. This selection was made because these symbols are found on most numeric keypads. Be sure that "Num Lock" is activated before the keypad is used. The program will work fine using keyboards without keypads. This program should work on all IBM PC compatible computers with coprocessors and Z-100 series computers with coprocessors as well as other computers with Intel Corporation 16-bit or 32-bit processors. It was tested on a Z-110 computer with an 8087 coprocessor and an H-386 computer with an 80387DX coprocessor.

The reader can easily add functions

and graphics to make a very advanced calculator program. The input routine permits the input of floating point numbers between -1,000,000,000,000,000 and 1,000,000,000,000,000. Exponential input is not recognized. However, the reader can easily add this feature if desired. The output routine displays numbers up to  $999999999999999999 \times 10^{199}$  where the upward pointing arrow indicates that the next three digits is the exponent of 10 (the number of zeros to be added to the number). Numbers of magnitudes less than 0.000000000000001 are considered zero. The reader can also add code for negative exponentiation if desired.

The length of this program is intimidating. Actually, the program appears much shorter if the comments are eliminated. It is not necessary to include comments in the source file a reader may generate. The extensive comments in the listing are to help the reader understand the logic of the program.

The output procedure is listed first and uses packed BCD strings. The input procedure uses unpacked BCD strings to show an alternate way to convert from one number format to another. Packed BCD strings could be used in the input procedure to shorten the code.

Although the code is long and complex, the user will find that the program is remarkably fast even on the 8088/8087 computers. Adding transcendental functions (trigonometry and logarithms) requires more work for the 8087 coprocessor than for the 80387 coprocessor, but the effort isn't very great and the user will be rewarded with very fast calculation of these functions to fourteen place accuracy. The results of the square root and tangent (not included in the listed version of the program) functions were checked against the National Bureau of Standards' "Handbook of Mathematical Functions" to verify this accuracy.

### Summary

This article introduces the use of the coprocessor feature of personal computers using the Intel Corporation processors or equivalent. The accompanying program listing includes basic input/output routines for floating point numbers. Once the reader has the basic understanding of the concepts outlined in this article, the reader should be able to easily write very powerful "number crunching" programs.

### References

1. Abramowitz, Milton and Irene A. Stegun, Handbook of Mathematical Functions, 1972, National Bureau of Standards, Washington, D.C.
2. Intel Corporation, 80387 Programmer's Reference Manual, 1987, Intel Corporation, Santa Clara, California.
3. Intel Corporation, i486 Microprocessor Programmer's Reference Manual, 1990, Intel/Osborne/McGraw-Hill.
4. Microsoft Corporation, Microsoft Macro Assembler 5.0 Programmer's Guide, 1987, Microsoft Corporation, Redmond, Washington.
5. Microsoft Corporation, Microsoft MS-DOS Operating System Programmers Reference, 1984, Microsoft Corporation, Bellevue, Washington.

### Trademarks

IBM PC is a trademark of the International Business Machine Corporation.

Intel and i486 are trademarks of the Intel Corporation. ☼

### Quality Heath/Zenith Upgrades

- H/Z-150 Speed Mod 6.7 MHz \$ 35
- \* 8 MHz DMA Controller and ALS Chips \$ 34
- \* MT PAL 704k RAM on Single Memory Card \$ 19
- \* Super PAL 1.2 Meg RAM, 640k + 640k RAM Disk \$ 29
- \* MT PAL 704k RAM on Single Memory Card \$ 19
- SmartWatch No-Slot Clock for ALL H/Z Computers \$ 29
- H89 WIN89 with 20 meg Hard Disk, CP/M or HDOS \$299
- 4 MHz Speed Mod (CP/M or HDOS) \$ 29
- \* H/Z-89: NZ.COM (ZCPR 3.4) \$ 59
- \* ZS-DOS with file time & date stamp \$ 65
- \* MT Modem for HDOS and CP/M \$ 15
- Perfect Money Loan Calculator \$ 19
- Everex RAM 3000 for H/Z-248 supports 3 Meg \$ 79
- H/Z-100 Speed Mod (7.5 or 8 MHz) \$ 37
- H/Z-148 Exp. Buss \$ 69, MT148 704k RAM PAL \$ 19

### Micronics Technology

Voice/FAX: (205)-244-1597 BBS: (205)-244-0192  
 SUITE 159, 54 DALRAIDA RD, MONTGOMERY, AL 36109  
 Call or Write for Our Free Catalog  
 Shipping: Speed Mod/Software \$3, Cards \$4, Hard Disks \$7  
 Hours: 8-8 PM CST M-F. Answering Machine when not available. We WILL return your calls!

```

int      21h      ;Call display character system routine
mov      dl,13   ;Put carriage return character into dl
int      21h     ;Call display character system routine

COMMENT * Transfer the first number to the top of the coprocessor's
stack. The second number will remain in memory. The first
number is the "destination" and the second number is the
"source." *

fld      number1 ;Put #1 on top of coprocessor stack

COMMENT \ The keyboard is now interrogated for the operation desired.

* = addition
- = subtraction
* = multiplication
/ = division
. = square root

Any other character causes the program to exit. Ctrl-C
will terminate the program also, but the state of the
coprocessor will not be restored. \

mov      ax,100h ;Put function number into register ax
int      21h     ;Call console input system routine
cmp      al,'+'  ;Compare to '+'
jne      nextop1 ;If not equal, check next operation
fadd     number2 ;Else, add numbers
jmp      nextop5 ;Display result

nextop1: cmp      al,'.'  ;Compare to '.'
jne      nextop2 ;If not equal, check next operation
fsub     number2 ;Else, subtract numbers
jmp      nextop5 ;Display result

nextop2: cmp      al,'*'  ;Compare to '*'
jne      nextop3 ;If not equal, check next operation
fmul     number2 ;Else, multiply numbers
jmp      nextop5 ;Display result

nextop3: cmp      al,'/'  ;Compare to '/'
jne      nextop4 ;If not equal, check next operation
fdiv     number2 ;Else, divide numbers
jmp      nextop5 ;Display result

nextop4: cmp      al,'.'  ;Compare to '.'
jne      xit     ;If not equal, exit
fsqrt

nextop5: mov      ax,200h ;Put function number into register ax
mov      dl,10   ;Put line feed character into reg. dl
int      21h     ;Call display character system routine
mov      dl,13   ;Put carriage return character into dl
int      21h     ;Call display character system routine
fstp     number2 ;Transfer result to number2 and pop
call     FAR PTR coutput ;Call output routine
mov      ax,200h ;Put function number into register ax
mov      dl,10   ;Put line feed character into reg. dl
int      21h     ;Call display character system routine
mov      dl,13   ;Put carriage return character into dl
int      21h     ;Call display character system routine
mov      dl,13   ;Put carriage return character into dl
int      21h     ;Call display character system routine
jmp      nextnumbers1 ;Get next numbers

```

```

COMMENT * This program uses a coprocessor (floating point processor)
to emulate a high precision calculator. Most of this code
consists of two subroutines (PROCedures) - one for keyboard
input of long real (double precision) floating point numbers
and one for the console display of these numbers. The
calculator functions are addition, subtraction, multi-
plication, division, and square root. Other functions can
be easily added by the user. This program was designed to
be short and quick. Therefore, some precision is sacrificed
and exponential input is not supported. In general, one can
expect fourteen decimal place precision.

The program will accept inputs between minus one quadrillion
and one quadrillion (-1,000,000,000,000,000 < input number <
1,000,000,000,000,000). Exponential expressions (e.g. E+25)
are not accepted as input. However, the output routine will
display exponential expressions up to E+199 (the coprocessor
has a greater range). Fractions less than 0.00000000000001
are considered zero. Negative exponential expressions are
not displayed.

This program is intended as an instructional tool even
though it may be useful. No expressed or implied
warranties are made for this program with respect to
merchantability or fitness for any other purpose. The
user assumes all responsibility for any damages resulting
from the use of this program.

Copyright (C) 1991 by David W. Lind *

data     SEGMENT ;Begin data segment
number1  DQ ?    ;First floating point number
number2  DQ ?    ;Second floating point number
coprocessorstate DB 108 DUP(?) ;Coprocessor state storage
data     ENDS   ;End data segment

code     SEGMENT ;Begin code segment
ASSUME  cs:code,ds:data

start:   mov      ax,data ;Start of executable code
mov      ds,ax   ;Put address of data segment into ax
nextnumbers1: fsave  coprocessorstate ;Save coprocessor state and initialize

COMMENT * Register di must point to the location in the data segment
where the floating point number will be stored. *

lea      di,number1 ;Put offset address of number1 into di
call     FAR PTR coutput ;Call input routine

COMMENT * The following five lines of code perform a line feed and
carriage return to place the next number on the next line. *

mov      ax,200h ;Put function number into register ax
mov      dl,10   ;Put line feed character into reg. dl
int      21h     ;Call display character system routine
mov      dl,13   ;Put carriage return character into dl
int      21h     ;Call display character system routine

COMMENT * In the same way as before, input number 2. *

lea      di,number2 ;Put offset address of number2 into di
call     FAR PTR coutput ;Call input routine
mov      ax,200h ;Put function number into register ax
mov      dl,10   ;Put line feed character into reg. dl

```

```

xit:  firstor  coprocessorstate  ;Restore coprocessor state
      mov     ah,4Ch             ;Put function number in register ah
      int    21h                ;Call terminate program function

cooutput  PROC  FAR
COMMENT * This PROCEDURE displays a long real (double precision)
floating point number on the standard console. Register
di must point to the location of the number in the data
segment.
Copyright (C) 1991 by David W. Lind *
COMMENT * Skip data storage area. *

      jmp     cooutputstart
cooutputexp  DW ? ;Unbiased exponent
cooutputstat  DW ? ;Coprocesor status storage
cooutputcount  DW ? ;Count of integer digits
cooutputhund  DW 100 ;Integer 100
cooutputtempa  DQ 0 ;Temporary storage
cooutputlarge  DQ 999999999999999.0 ;Largest permissible integer display
;or one quadrillion minus 1
cooutputinflarge  DQ 999999999999999.E*199 ;Largest display
cooutputdivreal  DQ 10.0 ;Real divisor or multiplier, value = 10
cooutputplace  DW 0 ;Place value string
cooutputbcd  DT ? ;Packed Binary Coded Decimal string
fcooutputbcd  DT ? ;Packed Binary Coded Decimal string

cooutputstart:
COMMENT * Save registers. *
      push  ax ;Save register ax
      push  bx ;Save register bx
      push  cx ;Save register cx
      push  dx ;Save register dx
      push  si ;Save register si
      push  di ;Save register di
      push  bp ;Save register bp

COMMENT * Transfer data to be output to cooutputtempa. *
      mov  ax,WORD PTR ds:[di][0] ;Put word into register ax
      mov  WORD PTR cooutputtempa[0],ax ;Transfer to cooutputtempa
      mov  ax,WORD PTR ds:[di][2] ;Put word into register ax
      mov  WORD PTR cooutputtempa[2],ax ;Transfer to cooutputtempa
      mov  ax,WORD PTR ds:[di][4] ;Put word into register ax
      mov  WORD PTR cooutputtempa[4],ax ;Transfer to cooutputtempa
      mov  ax,WORD PTR ds:[di][6] ;Put word into register ax
      mov  WORD PTR cooutputtempa[6],ax ;Transfer to cooutputtempa

      mov  cooutputexp,0 ;Zero exponent

      finit ;Initialize coprocessor
      fld  cooutputtempa ;Put number onto the stack
      fcomp ;Compare to limit
      fstsw cooutputstat ;Store status word
      fwait ;Wait for coprocessor to finish
      sahf ;Transfer status to register ax
      jc  cooutputnoreset ;Store result in flags register
      fld  cooutputinflarge ;If smaller, do not change
      fstp cooutputtempa ;Else, load limit
      ;Transfer to cooutputtempa

cooutputnoreset:

```

```

COMMENT * The following code determines and displays the sign. The
size of the exponent is also determined. *
      mov  ax,WORD PTR cooutputtempa[6] ;Put high order word in reg. ax
      shl  ax,1 ;Shift left to eliminate sign
      jnc  cooutputsign ;If no carry, display + sign
      mov  dl,'-' ;Place '-' into register dl
      mov  ax,200h ;Put subroutine # in reg. ax
      int  21h ;Call display byte
      jmp  cooutputexponent ;Examine exponent

cooutputsign:
      mov  dl,'+' ;Place '+' into register dl
      mov  ax,200h ;Put subroutine # in reg. ax
      int  21h ;Call display byte

COMMENT * In the remaining processing, the absolute value of the
number is used to avoid sign problems. Note that the
absolute value of the number is found by anding the sign
bit of the memory high word with zero. This means of
taking the absolute value is much faster than using
fabs. *
      WORD PTR cooutputtempa[6],7FFFh ;Take absolute value of input
      mov  ax,WORD PTR cooutputtempa[6] ;Put high word into reg. ax
      and  ax,7FFF0h ;Select exponent
      cmp  ax,4330h ;Compare to binary exp. of 52
      ja  cooutputexponent1 ;If above, skip next step
      jmp  cooutputplus ;Process with no exponent

COMMENT * When the following block of code is executed, the number
is divided by 10 until it is less than the maximum
permissible display number. The number of times that the
number is divided by 10 is counted using register cx. This
count becomes the exponent in the exponential expression.
The number of digits allowed in the display is driven by
the largest number which can be precisely represented by
a long real floating point number which is 900719254740991.
The next lowest complete magnitude was chosen as the limit. *

cooutputexponent1:
      finit ;Clear the coprocessor
      xor  cx,cx ;Zero register cx which is used to count
      fld  cooutputtempa ;Put the real number into the coprocessor

cooutputexponent2:
      fdiv cooutputdivreal ;Divide by 10
      inc  cx ;Increment the counter
      fcom cooutputlarge ;Compare the quotient to the display limit
      fstsw cooutputstat ;Store status word
      fwait ;Wait for the coprocessor to finish
      mov  ax,cooutputstat ;Move the coprocessor state data to
reg. ax
      sahf ;Store this data in the flag word
to set flags
      jnc  cooutputexponent2 ;If the carry flag is not set, st(0)>
      fst  cooutputtempa ;cooutputlarge - continue division
      mov  cooutputexp,cx ;Save count

COMMENT * The integer part of the number will now be converted to
a packed BCD number. *

cooutputplus0:
COMMENT * The following four lines of code are important because
they set the coprocessor to truncate rather than round

```

```

fmul          ;Multiply by ten
dec          ;Decrement digit counter
jge         ;If > or = zero, continue multiplication
frndint     ;Round to an integer
fbstp      ;Store the result in packed decimal string

cooutdivreal
cx          ;Decrement digit counter
cooutfrac1 ;Round to an integer
fcooutbcd  ;Store the result in packed decimal string

COMMENT * The following block of code examines the digit just above
the high fractional digit position. If that digit is one,
the fraction was rounded to an integer one. Therefore, the
integer portion of the number is incremented. *

mov bx,cooutcount ;Move fractional count to bx
inc bx           ;Increment register bx
shr bx,1        ;Divide by 2
mov al,byte ptr fcooutbcd[bx];Put most significant byte in al
jnc cooutfrac2  ;If remainder, skip 3 steps
cmp al,10h     ;Check high packed BCD
je cooutfrac3   ;If equal to 1, round integer
jmp cooutfrac4  ;Else, end analysis

cooutfrac2:
cmp al,1       ;Check least packed BCD
jne cooutfrac4 ;If not 1, skip rounding

cooutfrac3:
fild cooutbcd ;Put integer in coprocessor
fadd ;Push one onto the stack
fbstp ;Add
cooutfrac4:
mov bx,9       ;Store integer in packed BCD

cooutloop1:
mov bx,9       ;Set index for 10 bytes, 0 base
dec bx         ;Decrement index
j1 cooutloopend ;If less than zero, end process
xor ax,ax     ;Zero register ax
mov al,byte ptr cooutbcd[bx];Put high two packed BCD in al

COMMENT * The following three lines of code put the high
BCD digit into register ah and the low BCD
number into register al. *

mov cl,4      ;Put shift count in register cl
shl ax,cl     ;Shift register ax left
shr al,cl     ;Shift register al right
cmp ah,0      ;Compare high byte to zero
je cooutloop2 ;If equal, check low byte
jmp discooutloop1 ;Else, display

cooutloop2:
mov al,0      ;Compare low byte to zero
cooutloop1   ;If equal, check next byte
dl,al        ;Else, put result into reg. dl
add dl,48    ;Convert to ASCII character
mov ax,200h  ;Put DOS function number in ax
jmp discooutloop2 ;Display character

discooutloop0:
bx          ;Decrement index
dec bx       ;If less than zero, finish
xor ax,ax   ;Zero register ax
mov al,byte ptr cooutbcd[bx];Put packed BCD into reg. al
mov cl,4   ;Put shift count into reg. cl
shl ax,cl ;Shift register ax left
shr al,cl ;Shift register al right

```

```

to nearest (default process). *

init
fstcw
or
fldcw
fld
frndint
fbstp

COMMENT * Now that the integer part of the number is stored in packed
BCD format, a count of these decimal digits is needed to
determine the number of significant digits allowed in the
fractional part. This information in turn determines the
extent to which the fraction is rounded and if it is
necessary to add a unit to the integer part. *

mov bx,9 ;Set the index to 10 bytes
;zero based
;Set the digit counter to 20

cooutint0:
dec bp ;Decrement digit counter
dec bp ;Decrement digit counter again
dec bx ;Decrement index
j1 cooutint1 ;If less than zero, end process
mov ax,ax ;Zero register ax
mov al,byte ptr cooutbcd[bx];Put high two packed BCD in al
al,0 ;Is this value a leading zero?
je cooutint0 ;If yes, continue search
cmp al,10h ;Examine high packed BCD
jae cooutint1 ;If not zero, don't decrement
dec bp ;Else, decrement digit counter

cooutint1:
mov cooutcount,bp ;Store character count

COMMENT * The following code determines if a fraction can be
displayed and isolates the fraction. *

mov cx,13 ;Put digital limit (zero based)
into reg. cx
sub cx,cooutcount ;Subtract integer count
cmp cx,0 ;Compare to zero
jg cooutfraccount ;If greater, continue fractional analysis
mov cooutcount,0 ;Zero fractional count
jmp cooutfrac4 ;If < or =, no fraction display

cooutfraccount:
mov cooutcount,cx ;Store fractional digit count
fld coouttempa ;Put original real number in coprocessor
;Truncate
frndint ;Reload original real number
fchx ;Exchange st(0) and st(1)
fsub ;Subtract to isolate fraction

COMMENT * Now the decimal fraction digits may be found by repeated
multiplication by ten until the desired number of decimal
digits is obtained. *

COMMENT * First set rounding control to round to nearest digit. *

fstcw
and
fldcw
cooutfrac1:

```



```

discooutloop1:
add ax,3030h
mov dl,ah
mov dh,al
mov ax,200h
int 21h
mov dl,dh
mov 21h
discooutloop2:
int
jmp discooutloop0
discooutloop3:
jmp
cooutloopend:
COMMENT * The following three lines display a zero when the integer
part of the number is zero. *
mov ax,200h
mov dl,'0'
int 21h
cooutfract0:
COMMENT * Now the fractional portion of the number is displayed in a
similar manner to the integer display. *
mov ax,200h
mov dl,'.'
int 21h
mov bx,cooutcount
mov bx,0
cooutputxit
je bx
inc bx,1
shl
jnc fdiscooutloop0
mov al,BYTE PTR fcooutbcd[bx];Else, get packed BCD
and al,0FH
mov dl,al
add dl,48
mov ax,200h
jmp fdiscooutloop2
fdiscooutloop0:
dec bx
jl fdiscooutloop3
xor ax,ax
mov al,BYTE PTR fcooutbcd[bx];Put packed BCD into reg. al
mov cl,4
shl register ax left
shr register al right
add ax,3030h
mov dl,ah
mov dh,al
mov ax,200h
int 21h
inc bp
mov dl,dh
fdiscooutloop2:
int
jmp fdiscooutloop0
fdiscooutloop3:
cooutputxit:
;Convert both numbers to ASCII
;Put high number into reg. dl
;Save low number in reg. dh
;Put DOS function number in ax
;Display character
;Put lower number into reg. dl
;Display character
;Continue loop
;Convert both numbers to ASCII
;Put high number into reg. dl
;Save low number in reg. dh
;Put DOS function number in ax
;Display character
;Increment character counter
;Put lower number into reg. dl
;Display character
;Continue loop
cooutloopend:
COMMENT * Before terminating display exponent data if it exists. *
cmp cooutputexp,0
je cooutputxit1
mov ax,200h
mov dl,' '
mov 21h
mov dl,'X'
mov 21h
int
mov dl,' '
mov 21h
int
mov dl,'1'
mov 21h
int
mov dl,'0'
mov 21h
int
mov dl,24
mov 21h
int
mov ax,cooutputexp
mov dx,dx
xor
COMMENT * The following code reduces the exponent (which can be as
large as 308 but limited to 199 in this PROCEDURE) to
hundreds, tens and units digits and displays the ASCII
character representations of these digits. *
mov cooutthund
dx
push
aam
mov dx,ax
add dx,48
mov ax,200h
int 21h
pop ax
aam
push
mov dl,ah
add dl,48
mov ax,200h
int 21h
pop ax
push
mov dl,ah
add dl,48
mov ax,200h
int 21h
pop ax
mov dl,al
add dl,48
mov ax,200h
int 21h
cooutputxit1:
COMMENT * Restore registers and return (FAR). *
pop bp
pop di
pop si
pop dx
pop cx
pop bx
pop ax
COMMENT * The coprocessor should be reset at the end of the PROCEDURE
to ensure future calls to the coprocessor are not affected
by the current state of the coprocessor. *
finit
retf
;Initialize coprocessor
;Pop registers cs and ip from stack

```

```

cooutput ENDP ;End of PROCEDURE
coinput PROC FAR
COMMENT * This PROCEDURE accepts American Standard Code for Information
Interchange (ASCII) floating point number data and converts
it to the Institute for Electrical and Electronic Engineers
(IEEE) standard long real (double precision) floating point
format. A coprocessor is used for this conversion. THIS
PROCEDURE WILL ONLY WORK ON A COMPUTER THAT HAS AN ACTIVE
COPROCESSOR. If it is to be used on computers without
coprocessors, the coprocessor functions must be emulated.
This procedure will NOT accept exponents and numbers whose
absolute value (magnitude irrespective of sign) are greater
than one quadrillion minus 1. The program ignores commas and
makes blanks zeros. Register di must point to the memory
location in the data segment where the number is to be
stored.
Copyright (C) 1991 by David W. Lind
*
COMMENT * Skip data storage area. *
jmp coinputstart ;Jump to start of executable code
dsaddress DW ? ;Data segment address
lroffset DW ? ;Long real offset address
coinwholecount DW 0 ;Number of whole decimal characters
coinfractcount DW 0 ;Number of fractional decimal characters
coinerrormess1 DB ' is an illegal number character input. Input number
DB ' again. ',10,13,'$'
coinofmess DB ' Overflow error on input
(number cannot exceed i qu'
DB 'adrillion-1). Input again.',10,13,'$'
stringin DB 80,81 DUP(?) ;Input string
stringfract DB 20 DUP(0) ;Storage for fraction
stringwhole DB 20 DUP(0) ;Storage for whole number
coinsign DB 0 ;Storage for sign
coinwhole DQ 0 ;Integer portion of input
coinfract DQ 0 ;Fractional portion of input
coincount DW ? ;Number of ASCII characters input
wholenumber DT 0 ;Storage for BCD numbers
cointen DQ 10.0 ;Floating point 10
coinputstart:
COMMENT * Save registers. *
push ax ;Save register ax
push bx ;Save register bx
push cx ;Save register cx
push dx ;Save register dx
push bp ;Save register bp
push di ;Save register di
push si ;Save register si
COMMENT * Save segment address of data segment and the offset of
the long real storage string. *
mov ax,ds ;Put address of data segment into ax
mov dsaddress,ax ;Store address
mov lroffset,di ;Store offset of long real
xor bp,bp ;Zero register bp
coinstrin:

```

```

BYTE PTR stringfract[bp],0 ;Zero stringfract
BYTE PTR stringwhole[bp],0 ;Zero stringwhole
inc bp ;Increment register bp
cmp bp,20 ;Compare to 20
jb coinstrin ;If below, continue loop
coin0:
COMMENT * Initialize variables. *
xor ax,ax ;Zero register ax
mov WORD PTR coinwhole,ax ;Zero memory
mov WORD PTR coinwhole[2],ax ;Zero memory
mov WORD PTR coinwhole[4],ax ;Zero memory
mov WORD PTR coinwhole[6],ax ;Zero memory
mov WORD PTR coinfract,ax ;Zero memory
mov WORD PTR coinfract[2],ax ;Zero memory
mov WORD PTR coinfract[4],ax ;Zero memory
mov WORD PTR coinfract[6],ax ;Zero memory
mov coinsign,0 ;Zero coinsign
COMMENT * Make data segment the same as the current code segment and
input data into stringin. *
mov ax,cs ;Put value in cs register into ax register
mov ds,ax ;Make data segment code segment
lea dx,stringin ;Put offset of stringin into register dx
mov ax,0A00h ;Put function number into register ax
int 21h ;Call DOS buffered input routine
mov ax,dsaddress ;Get address of main data segment
mov ds,ax ;Restore data segment
COMMENT * Read the input string and store the integer and fractional
parts in coinwhole and coinfract after converting to binary
format. *
xor ax,ax ;Zero register ax
mov al,stringin[1] ;Put number of bytes read into register al
mov coincount,ax ;Save number of bytes read in coincount
xor bp,bp ;Zero register bp
xor si,si ;Zero register si
cmp bp,coincount ;Check number of bytes read
jb coinseparate0a ;If below, continue processing
mov coinwholecount,si ;Store count of whole number digits
jmp coinwhole0 ;Else, process whole number
coinseparate0a:
mov al,stringin[bp][2] ;Get next character in string
mov al,48 ;Compare character to ASCII zero
cmp coinseparate1,jb ;If below, process non-number
jmp coinseparate8 ;Else, assume character is a number
cmp al,'.' ;Is the character a decimal point?
jne coinseparate2 ;If not, check for sign
inc bp ;Increment index
jmp coinseparate10 ;Else, process fractional part
coinseparate2:
cmp al,'+' ;Is character a plus sign
je coinseparate3 ;If so, go to next character
jmp coinseparate4 ;Else, check for '-'
coinseparate3:
inc bp ;Increment index
jmp coinseparate0 ;Consider next character
coinseparate4:
cmp al,'-' ;Is the character -?

```

```

je coinseparate5 ;If so, store result
jmp coinseparate6 ;Else, look for blank
mov coinsign,1 ;Put 1 in coinsign
inc bp ;Increment index
jmp coinseparate0 ;Process next character

coinseparate6:
cmp al,32 ;Is the character a blank?
je coinseparate7 ;If so, make zero
jmp coinseparate7a ;Else, check for ','
mov al,48 ;Put ASCII zero into register al

coinseparate7a:
cmp al,',,' ;Compare to ',,'
je coinseparate7b ;If ',,' skip next step
jmp coinerror ;Else, jump to error routine

coinseparate7b:
inc bp ;Increment counter
jmp coinseparate0 ;Get next character

coinseparate8:
cmp al,58 ;Ensure input is a number
jb coinseparate9 ;If below, continue process
jmp coinerror ;Else, go to error section

coinseparate9:
sub al,48 ;Convert to Binary Coded Decimal (BCD)
mov stringwhole[si],al ;Put unpacked BCD into whole string
inc si ;Increment counter
cmp si,15 ;Compare count to 15
jbe coinseparate9a ;If below, continue
jmp coinoverflow ;Else, display error message

coinseparate9a:
inc bp ;Increment index
jmp coinseparate0 ;Continue loop

coinseparate10:
mov coinwholecount,si ;Store number of integer characters
xor si,si ;Zero register si

coinseparate11:
cmp bp,coincount ;Check number of bytes read
jb coinseparate12 ;If below, continue processing
jmp coinseparate14 ;Else, process whole number

coinseparate12:
mov al,stringin[bp][2] ;Get next character in string
cmp al,58 ;Ensure input is a number
jb coinseparate13 ;If below, continue processing
jmp coinerror ;Else, go to error section

coinseparate13:
sub al,48 ;Convert to Binary Coded Decimal (BCD)
mov stringfract[si],al ;Put BCD number into fractional string
inc si ;Increment counter
jmp coinseparate11 ;Continue loop

coinseparate14:
mov coinfractcount,si ;Store number of fractional characters
mov coinwhole0:

COMMENT * The following code multiplies each BCD number in stringwhole
by the appropriate power of ten and accumulates the sum of
these products in memory at coinwhole. *

finit ;Initialize coprocessor
mov si,coinwholecount ;Move the count of whole numbers
xor di,di ;Zero register di
xor bp,bp ;Zero register bp
coinwhole1:
dec si ;Set current offset (from variable

```

## "What Did I Change?"

Answer that question by using Directory Compare. Compare date/size of files in any two directories. Public Domain (source included). Send only \$5.00 shipping & handling. For PC or Z100 (specify diskette size)

## "Can I Improve My Screen Prints?"

Yes - with EGAD Graphics & Text Screen Print PC's and now also for the Z-100 series.

- Prints *IN COLOR* on color printers, or uses black, white and *six gray tones* on most other printers
- Print *any part* of the screen - crop box pops up when Shift-PrtSc (PC) or Shift-F12 (Z-100) pressed; use arrow keys to select region.
- Enlarge graphics 1-4 times
- Supports Super VGA, VGA, EGA, and CGA
- SET program selects printer colors, other options.
- Supports most dot matrix, laser, and ink jet printers (including Epson, NEC-8023, MPI, Okidata, HP, etc.)

EGAD for PC's, Order # 270 \$35.00 postpaid. Specify 3.5" or 5.25" diskette.

EGAD for Z-100, Order # 271 \$35.00 postpaid.

LS Software (formerly Lindley Systems)  
8139 E. Mawson Rd., Mesa AZ 85207  
(602) 380-9175. Call/Write for Free Catalog

```

mov WORD PTR ds:[di][2],ax ;Move to calling program
mov ax,WORD PTR coinwhole[4] ;Put coinwhole word in reg, ax
mov WORD PTR ds:[di][4],ax ;Move to calling program
mov ax,WORD PTR coinwhole[6] ;Put coinwhole word in reg, ax
mov WORD PTR ds:[di][6],ax ;Move to calling program
jmp coinxit ;Exit subroutine

coinerror:
push ds ;Save register ds
push ax ;Save register ax
mov ax,200h ;Put subroutine number into ax
mov dl,10 ;Put LF in register dl
int 21h ;Display character
mov dl,13 ;Put carriage return into register dl
int 21h ;Display character
pop ax ;Restore register ax
mov dl,al ;Move character to register dl
mov ax,200h ;Put subroutine number into register ax
int 21h ;Display character
mov ax,cs ;Put address of code segment into reg. ax
mov ds,ax ;Make code segment data segment
mov ax,900h ;Put subroutine number into register ax
mov dx,coinerrormess1 ;Put offset address of message into
reg. dx
int 21h ;Call display string
pop ds ;Restore data segment
jmp coin0 ;Jump to beginning code

coinoverflow:
push ds ;Save register ds
mov ax,cs ;Read code segment
mov ds,ax ;Make data segment code segment
mov ax,900h ;Put subroutine number into register ax
lea dx,coinofmess ;Put offset of message into register dx
int 21h ;Call display string
pop ds ;Restore register ds
jmp coin0 ;Jump to beginning code

coinxit:
pop si ;Restore register si
pop di ;Restore register di
pop bp ;Restore register bp
pop dx ;Restore register dx
pop cx ;Restore register cx
pop bx ;Restore register bx
pop ax ;Restore register ax

COMMENT * The coprocessor should be reset at the end of the PROCEDURE
to ensure future calls to the coprocessor are not affected
by the current state of the coprocessor. *

finit
;Initialize coprocessor
;Return far (pop registers cs and ip)

retf ;End PROCEDURE
ENDP ;End of code segment

stack SEGMENT stack
DW 256 DUP (?)
stack ENDS ;End stack

.END PAGE
.END PRINT

```



"SINCE I CONVERTED TO THE MICRO SYSTEM, WE SOLD ALL OUR REELS OF TAPE TO A CONFETTI FACTORY AND NOW I'M FINANCIALLY INDEPENDENT!"

---

---

# Perpetual Upgrading of a Z-248

**Nick Visco**  
**5639 Heming Avenue**  
**Springfield, VA 22151-2707**

I didn't start out to upgrade my venerable Z-248 to a 386SX, but as fortune would have it, I am certainly glad I did upgrade. Let's start at the beginning with my purchase of a Z-248 and go all the way to my present upgrade.

## Before the Beginning

In February 1986, Zenith Data Systems was awarded a three year government contract to provide Z-248s to the government. In the spring of 1987, Zenith Data Systems decided to offer military and government workers the opportunity to buy Z-248s just above the price that they had been selling them to the government. The Z-248s were configured with dual floppy 5-1/4" disk drives, 512 KB of memory, EGA monitor, and optionally a 20 MB hard disk, just in case you ever thought you could fill one up. I decided that this was the perfect time to buy my first microcomputer. I will digress for a short while here for two purposes; first, to increase the length of this article to the required number of words, and secondly, to give you a little background of myself to let you identify with me in some small way.

In March of 1975, I started a masters program in something called computer science at the Naval Postgraduate School at Monterey, California. Gary Kildall, who wrote the 8-bit operating system CP/M, was one of my instructors, and Gordon Eubanks, who is president and CEO of Symantec, was a classmate. Microcomputers were just getting started, with 8-inch floppy disks from Shugart, Omron terminals with keyboard and monitor attached (4 KB of memory). I forget how much internal RAM was available. We had the familiar C>prompt, Debug.com, Edlin.com, Dir, and other ancient software (?) and

hardware. Altair computers were the rage of microcomputer buffs. I thought these little toys were cute, but the IBM 360 with punch cards (our military paychecks were printed on these every 15 days) and COBOL we had was real computing. I listened to Gary Kildall in class, and all about Intel 8008 and 8080 assembly code, which sounded very interesting, but FORTRAN was so easy (?) to use. In December of 1976, I completed a masters degree in computer science and went on with my job as a Naval Officer. I was sure Gordon and Gary would get real computer jobs with big computers in the future. I almost got it right, they got big computer jobs with real computers. Way to go guys!

## The Beginning

I originally purchased my 8 MHz Z-248 in April of the 1987. The package included a 80286 processor, 512 KB of RAM, one 5-1/4" floppy (360 KB), one Seagate 20 MB (model 225) hard disk drive, one serial, and one parallel port, an EGA adapter (256 KB), a 84-key keyboard, ten 5-1/4" diskettes, MS-DOS 3.2, GW-BASIC, Windows 1.0, diagnostic software, and an EGA color monitor (Model ZVM-1380). I failed to mention the 10 slots (some already filled with a disk controller card, a CPU card, an I/O card, and a video card. This left 6 open slots). All this for the small price of \$2,482 which included tax, insurance, and delivery. I bought a copy of WordStar in June of 1987, a NEC P6 printer (which is still my only printer, no upgrades) in July of 1987, and I was on my way. In November of 1987, I received a surprise from Zenith Data Systems, a check in the amount of \$150. They had recently dropped the price of the CPU by \$100 and the monitor by \$50 and decided to give a

rebate to all those that had purchased their computers before the price decrease. I was happy and amazed that shortly over seven months after buying a product, that I considered worth every penny I spent, I would receive an unsolicited refund. I was hoping for more price cuts! No such luck.

As faith would have it, my Z-248 died in November of 1988. I opened up the machine and noticed that on power up, some of the green LEDs did not light up. It turns out that the I/O board was bad. Since the I/O has other functions besides just providing a serial and parallel port, I decided (I'm not sure if I had a choice) to purchase a replacement from Zenith Data Systems for a mere \$192. E.G come, E.G go.

## Peripheral Upgrades

In July of 1988, I was assigned to the United States Naval Academy as an instructor in Computer Science. All instructors in the Computer Science department had a Z-248 on an Ethernet network with E-Mail on MILNET and BITNET to all colleges on the network. I checked out a 2400 bps Hayes modem from school. I thus had access to all the computers on campus that were on the network including my own if I left it in host mode. Plus, I had the ability to send E-Mail nationwide. These microcomputers had come a long way. I noticed in the computer trade journals that Gary and Gordon were doing quite well. Maybe they had been on to something big.

We taught Computer Literacy and PASCAL as an introduction course to all incoming plebes (freshmen). Since all plebes were required to purchase a Z-248 in August of 1986 and 1987, and a Z-286 in 1988 and 1989, I had zero compatibility problems during my two year tour. A LAN was installed in 1989 in Bancroft Hall where

all midshipmen live. This was great for instructors and maybe not so great for students. I could send assignments via E-Mail from my office or home to all my students using an automatically generated mail list, instead of individually. The students did get back at me by sending me E-Mail on weekends, asking many questions. Why were they studying on weekends anyway? I also used disk space on a local mini-computer to store sample programs that could be downloaded by the students and run on their computers. Completed assignments and homework were also required to be sent over the network. Programs were sent in source code only. I would compile and run the student written programs to prevent virus infection. One professor used to accept compiled programs until one of them infected his computer.

I quickly determined that my Z-248 would require moderate upgrading if I was to keep pace with the technology explosion going on all around me. I purchased a serial Logitech mouse and had to install a serial card since my original serial port was being used by the modem. I also purchased 128 KB of memory from First Capitol Computer to fill the memory from 512 KB to 640 KB. More about this memory upgrade later. I purchased a high density 3-1/2", 1.4 MB (not 1.44 MB as is commonly referred to; because 1 MB is 1 KB X 1 KB or 1,024 X 1,024 = 1,048,576 bytes = 1 MB. A high density disk has 2,880 sectors of 512 bytes which is 1,474,560 bytes. 1,474,560 divided by 1,048,576 is 1.40625 MB. If we try our math with two times 720 KB to get 1.44 MB we will also fail. Two times 720 KB (1,024 X 720) is 737,280 bytes and two times this number is 1,474,560 bytes which divided by 1,048,576 gives us the same answer of 1.40625 MB. You are probably asking why this is so important. Well it is not very important, unless you have just finished your article and are about a hundred words short) disk drive.

I purchased this 1.44 MB (oops! 1.4 MB) drive to be compatible with the midshipmen that I sponsored (i.e., gave them a place to crash on weekends) who would come over my house with these little disks and try to put them in my 5-1/4" 360 KB floppy drive. Unsuccessfully I might add. I also determined that with all the software that I was purchasing, the assignments I was generating, and all the lesson plans that I was creating, a 20 MB hard drive was not going to be enough. So I purchased a 40 MB Seagate model 251-1 hard drive. Zenith Data Systems, thank their technical expertise, not only provided a disk controller board that allowed two floppy drives, and two hard drives; but also allowed the space to fit each internal bay of the machine with required power wires from the 200 watt power supply. Of course I had to

purchase updated ROMs to access the 1.4 MB drive. No problem as I had already replaced numerous ROMs in the three labs that we had on campus for an identical upgrade.

### Video Upgrade

Upon departing the Naval Academy and retiring from the Navy in June of 1990, I returned my Hayes modem and decided that I could not live without one, so I purchased a Practical Peripheral 2400 bps external modem. After retiring, I took a short vacation and tried to find out what I really want to do with my life. I decided I would become a computer consultant. The last time I checked, Gary and Gordon were doing quite nicely in this microcomputer business.

I set up shop in my basement with my trusty Z-248 and started to build up a clientele. I purchased Windows 3.0 for two purposes. One, to make it easier on my wife and daughter to point and select whatever program they desired, and two, for me to play solitaire. I mean for me to become familiar with Windows. Of course, I could only run Windows in real mode since I only had 640 KB of memory, but so what, I didn't have a need to run it in any other mode.

One day before a big assignment was due, my monitor died. No green power light on the front panel. I had seen (no pun intended) this problem before on some machines in the computer labs at the Academy. The power supply in the monitor was bad. I called some local repair places. They would fix it for \$150. What to do? Fix an EGA monitor that still has a good adapter card or go with the new technology, VGA? Well I decided to upgrade my computer. I purchased a 14" VGA monitor with .28 mm dot pitch and an inexpensive VGA card for \$329 and \$89 respectfully. Later I found a repair place that would have fixed my EGA monitor for \$89. But, I am glad I now have a VGA monitor. Smart of Zenith Data Systems to have a video card slot which can be replaced instead of a video port in the mother board. As with the floppy drive, hard disk drive, modem, memory, ROM, and mouse upgrade, I was back in action in no time. I was under a lot more time pressure this time, but no problems were encountered. My assignment was completed on time.

### Memory Upgrade

I mentioned that, back in late 1988, I purchased a 128 KB memory card from First Capitol Computer in St. Peters, Missouri. I have had three big problems with this memory upgrade. The first problem I encountered was when I tried to install Prodigy on my computer. The install process would freeze up my computer. I tried various problem solving techniques (plain

vanilla AUTOEXEC.BAT and CONFIG.SYS files, booting from a floppy with the 360 KB as my A: drive and then the 1.4 MB drive as the A: drive, no luck). The solution that finally worked was to remove the 128 KB memory card and install Prodigy. Prodigy worked fine with just the original 512 KB of memory, but I had grown accustomed to having 640 KB of memory. I decided to reinstall the 128 KB memory card. To my surprise, Prodigy worked as well as all my software. When Prodigy updated their software a few months later, I ran into the same problem. This time I was ready. I took out the 128 KB memory card, installed Prodigy, and then reinstalled the 128 KB memory card. My next problem with the 128 KB memory card came when I tried to retrieve a WordPerfect document from either floppy disk drive. I could not retrieve the document without getting a file conversion error. I could save the document to my hard drive with no problem and retrieve it. I could save the document to my floppy disk but could not retrieve it using my machine. I could retrieve the document from the floppy disk if I placed it in another machine that had WordPerfect. I tried various problem techniques again with no success. When I removed my 128 KB memory card, this problem disappeared. I figured I could live with this known problem, so I reinstalled my 128 KB memory card. The third problem that I encountered with this card was when I decided to upgrade my operating system to MS-DOS 5.0. During the install process some of the files would not expand from their compressed form when being copied from the MS-DOS 5.0 floppy to my hard drive. I manually expanded the files that would not do so on installation. I guess these files were located in the region 512 KB to 640 KB since it happened to groups of files every so often. I did not attempt to reinstall MS-DOS 5.0 without my 128 KB memory card so this is only conjecture. But I still had a problem. When I tried to format a disk with the operating system on it, everything appeared to work. The problem was that I could not boot off this floppy. I removed the 128 KB memory card. I could now boot off of this floppy. I reinstalled my 128 KB memory card and decided to live with these three known problems. My guess is that the transient portion of COMMAND.COM that loads in 640 KB downward is somehow incompatible (timing?) with the 128 KB memory card. I called First Capitol Computer to see if this is a known problem with a known fix, but they never returned my call.

I decided now was the time to upgrade my memory beyond 640 KB and rid myself of this 128 KB memory card. I decided on an AST 286 memory card since they used standard SIMMs which I could use later in another machine. I purchased

Continued on Page 48

---

# On the Leading Edge

William M. Adney

P.O. Box 531655

Grand Prairie, TX 75053-1655

Copyright © 1991 by William M. Adney. All rights reserved.

If this is your first introduction to our users' group, let me be among the first to welcome you. I have written this particular column to include specific information that is of special interest to new Zenith Users' Group members. I began writing this special column a couple of years ago, and it turned out to be so popular that I will continue the idea. But before we get too far along, I think it might be a good idea to introduce myself again for the benefit of our new members.

## An Introduction

Regardless of what my byline says, my friends call me Bill, and I would be pleased if you would do so. I have over 24 years' experience in various computer-related and data processing functions ranging from systems analysis and design to management. During this time, I have worked for various companies like Honeywell, McDonnell Douglas, TransAmerica, and Atlantic Richfield Company (ARCO). My educational background includes a B.S.E.E. (Electrical Engineering) degree from Purdue University and an M.B.A. from West Coast University in Los Angeles, California. I have written over a dozen books or courses as well as several hundred computer-related articles of one kind or another.

I am currently a freelance computer consultant and writer. As some of you may recall, I was a Senior Consultant at TAP, Incorporated (Total Assets Protection) in Arlington, Texas; however TAP ceased business operations on September 17, 1991 as a result of major cash flow problems. I have worked with all kinds of computer systems since 1967, and my personal specialty is the analysis and development of information security and disaster recovery programs for large mainframe data centers, as well as business function recovery planning for business operations. As a result of this consulting experience, I have also worked with all major brands of microcomputers and software as well as a number of

Local Area Networks (LANs) and Wide Area Networks (WANs). This has given me the opportunity to use and compare a lot of computer systems from major manufacturers (including Zenith Data Systems, IBM, Compaq, and Tandy), add-on boards, and a wide array of word processing, spreadsheet, utilities, and data base software. Regardless of what hardware I have used, I still think that the Zenith Data Systems computers are the best, and those are the ones that I personally chose to buy and write about.

As you will see on the masthead, I am a Contributing Editor and have been writing this column for *REMark* since 1983. I have been a HUG/ZUG member since 1982. During this time, I have seen a major shift in the ZUG membership from primarily hobbyist-oriented interests to the business-related use of computers. Many new ZUG members are not necessarily interested in all of the technical details of their computers, but at least some knowledge is required to successfully operate and use these systems. You can find nearly all of the information you will need to begin using your computer in the documentation provided with your computer and the *Powering Up* books available from ZUG.

## What's the Purpose of ZUG?

The Zenith User's Group, usually called ZUG, was originally established over 13 years ago as the Heath Users' Group or HUG. Due to the widespread use of Zenith Data Systems computers, the "long" name of the user group was changed to Heath/Zenith Users' Group; however, the HUG name was still used to refer to this expanded group of users. In 1991, the HUG staff officially transferred from Heath Company to Zenith Data Systems, and the official name was changed to ZUG.

As published in each and every *REMark*, ZUG is provided as a service to its members for the purpose of fostering the exchange of ideas to enhance their use of

Zenith Data Systems equipment. In other words, the whole concept of ZUG is to help members exchange ideas on the use of Zenith Data Systems computers. And the whole idea of *REMark* is to provide a medium where members can write about, and exchange information on, various topics that can help other ZUG members.

Articles published in *REMark* are written by ZUG members for ZUG members. If a ZUG member (including the ZUG staff) does not write about it, then there is no way you will see it in this magazine. If you have a specific question or suggestion for a topic that you need additional information on, I suggest you write to ZUG or directly to me with that idea. Most of the topics I write about are a direct result of questions or suggestions in the letters I receive. Or, if you have an idea for an article that would be of interest to other ZUG members, consider the idea of writing about it yourself. Information about compensation for published articles is included elsewhere in this issue.

## The Purpose of this Column

Since it began, the purpose of this column has always been to help ZUG members keep "On the Leading Edge" for information about current computer software and hardware. That includes various topics, such as new Zenith Data Systems hardware, operating systems, and third-party products which can be used on these systems. When I find out about new items of general interest, such as ROM changes that may be required to use new software, that information has been and will be published here. When a new release of Zenith Data Systems MS-DOS is available, I include general information about it to help you decide if you should upgrade or not. Sometimes it takes a little time for me to explore new features so that I can provide you with ideas on how to intelligently use them. In other cases, I may not use a new feature on a regular basis, and I may not

write about it unless I receive a letter which indicates that the subject may be of general interest to all ZUG members.

Some members have suggested a continuing series of articles that includes specific information about Zenith Data Systems computers and MS-DOS. In short, the purpose of this column is exactly that. Now let's move on to some other topics of particular interest to new ZUG members.

### Compatibility

The most common question that we receive at ZUG has to do with the various model numbers of Zenith Data Systems PC compatible computers. Perhaps the most common question of all is something like: "Why don't you have any articles about MY computer?" That kind of question is either preceded or followed by a statement about which model that user has. Sometimes the question is phrased in such a way that it is clear the user does not know what the general "type" (e.g., PC compatible) category the system falls into. Before we get into specific model and series numbers, it is important to define what we are talking about when the term "PC Compatible" is used.

### IBM Compatibility

What the term "PC Compatible" means is really determined by the context in which it is used. As used by ZUG and in *REMark*, the term "PC Compatible" refers to *SOFTWARE* compatibility. That is, software generally listed as "PC Compatible" will run on Heathkit and Zenith Data Systems computers as well as other compatibles, such as IBM, Compaq, etc. In other words, it is generally safe to assume that any software which will run on an IBM PC (where PC compatibility is essentially defined) will also run on a Zenith Data Systems or Heathkit PC compatible computer. For purposes of ZUG software, the term "PC Compatible" means that the software will run on any Heathkit or Zenith Data Systems PC compatible computer, and most other compatible systems too. If you read this paragraph carefully, you will note that I said "generally safe to assume", which of course means that there are some exceptions. To understand why that is true, it is necessary to review a brief bit of history on the development and "compatibility" of IBM computers.

The first IBM microcomputer was called the IBM Personal Computer, or IBM PC for short. The original IBM PC featured a whopping 64 kilobytes of memory (later increased to 128 K), SINGLE-sided/double density 5.25-inch disk drives (180 K) that were later changed to double sided/double density (360 K), a cassette tape interface, a single parallel port for a printer (a serial port was optional at extra cost), an 8088 CPU running at 4.77 MHz, and a 65-watt power

supply. Most early PCs were equipped with a CGA video card and monitor. In the hardware sense, true PC compatibility was essentially defined as this kind of configuration. Because of several interesting problems with this specific configuration, IBM introduced the IBM XT computer.

The XT (eXtended Technology) was essentially a "reworked" PC with one very important difference: it had a larger power supply. As I recall, it was rated at something around 90 watts or so. A beefed-up power supply was required because the other significant difference in the XT was that it included a hard drive, usually on the order of 5-10 MB. In other words, the original PC was so underpowered that it did not have a big enough power supply to support internal electronics (including memory), two floppy drives, AND a hard drive; otherwise the technical specifications were about the same, including the 8088 CPU and the 4.77 MHz clock speed. Through the normal development process, IBM introduced the AT.

The original AT (Advanced Technology) contained an 80286 CPU and ran at a screaming 6 MHz clock speed with one wait state. This original AT came equipped with a whopping 512 KB of system memory and a 5.25-inch high density (1.2 MB) floppy disk drive. This was the "standard" configuration, but most ATs were also equipped with a hard drive. Later, another AT was released, and it had a clock speed of 8 MHz with one wait state. These two computers essentially define what "AT compatible" really means; however, advances in this technology caused all kinds of problems. And general software and hardware compatibility with other manufacturer's systems were the biggest.

### Copy Protection and Compatibility

In the early days of computer software, many paranoid software manufacturers developed software with a copy-protection feature. At least one form of copy protection checked a computer's clock speed, and many irate users found out the hard way that true PC compatible software would not run on their new AT systems because of the increased clock speed. In particular, many games checked to see if the computer was running at 4.77 MHz, and if it wasn't, the game program would not run. Many users viewed this approach as a rather shabby marketing technique to force them to buy new software. And many software manufacturers at the time would not even give a legally registered software user a discount for an update to a new version. This particular problem is one reason that Zenith Data Systems introduced the dual-speed (4.77 MHz and 8 MHz) Z-158 computer system.

Although some copy-protected software still exists today, most enlightened

software manufacturers have removed it. Lest you believe they did that because of a user rebellion against it (which may be a small part of the reason), most manufacturers dropped copy protection for the very good business reason that it was causing incredible problems (especially installation) for their technical support groups. In other words, copy protection was causing lots of problems and costing them lots of MONEY — so much so that it was more cost effective to remove copy protection than to support it. Some software manufacturers might have you believe that they removed copy protection for altruistic reasons, but I am more than a little skeptical of those claims.

There are many other forms of copy protection — some of them use one or more "hidden" files (from the DIR command), a few actually change the "format" of a disk, and one form of copy protection actually has a hole punched in a disk (usually by a laser) that prevents it from being copied by DOS. Another form of copy protection requires a "key disk" that must be inserted in a floppy drive to run a program, even if the software is installed on a hard drive. One of the latest forms of copy protection is in the form of a "hardware key" — called a DONGLE — (yes, that's really the name of it) that plugs into a serial or parallel port and is checked by the software during operation. A dongle looks like, and is about the size of, a gender changer for a serial or parallel port. It usually has a DB-25 connector on both ends, so that you can plug it into the computer and then plug in a peripheral, such as a printer or modem, directly into the dongle. It is claimed that the dongle does not interfere with any peripheral in any way.

Even today, copy protection is still not quite dead. With the introduction of DOS versions 4.0 and 5.0, some manufacturers are using the DOS-generated Volume Serial Number as a copy protection feature. When you use FORMAT to initialize a disk or DISKCOPY to copy a disk, DOS generates a unique Volume Serial Number that it uses to detect whether a disk has been changed in a drive. Unfortunately, that now means that DISKCOPY no longer makes an exact duplicate of the source disk (normally used as a backup) because the Volume Serial Numbers are different. You can see the Volume Serial Number on any disk by using either DIR or CHKDSK.

### Hardware Compatibility

A discussion of hardware compatibility is difficult because it means different things to different people. Unfortunately, many people have the mistaken idea that a "compatible" computer from any manufacturer, including Zenith Data Systems, is 100% compatible for both hardware and software. That is very often not true, be-



cause of some things that hardware and software manufacturers do. Speed-sensitive software that was mentioned earlier is one of those problems, but there are two other common problems.

The first problem is that some software is not compatible with other software, regardless of the general hardware compatibility features of a system. Memory-resident programs, like device drivers and many desktop utility programs (e.g., SideKick), sometimes have conflicts with each other. In many cases, it is at least difficult, if not impossible, to try to predict what software will produce conflicts unless it is actually tested. Some manufacturers still use non-standard programming techniques that are used to improve speed, functionality, or both. Windows 3.0 is an example of that.

The second problem that is directly related to hardware compatibility is that some manufacturers, including Zenith Data Systems, have built better systems than IBM. I think the best example of that is when Zenith Data Systems introduced the AT-compatible 6 MHz Z-241, which could not use some of the memory boards developed for the IBM AT. As it turned out, Zenith Data Systems used better engineering to produce a computer with zero wait states (memory) which was demonstrably faster than its IBM counterpart. Unfortunately, many of the early memory boards were developed using the IBM "standard" of one wait state, and these memory boards were simply not "fast" enough to keep up with a Zenith Data Systems computer using a zero-wait-state configuration. If you are interested in knowing more about a computer's memory, the chapter on "Adding More Memory to Your Computer" in *Powering Up* goes into considerably more detail on the information you must know before you buy anything.

The issue of hardware compatibility goes even deeper. In the true hardware sense, I don't think Zenith Data Systems has ever produced a "real" PC compatible desktop computer. The first PC compatible, called the Z-151, was actually an XT compatible because its larger power supply COULD support a hard drive, which is really the major difference in hardware between a PC compatible and an XT compatible. That's why you sometimes see the term "PC/XT compatible" when you look at software.

As time goes on, you will probably continue to see the term "AT compatible" which usually refers to a computer with an 80286 Central Processor Unit (CPU). In many cases, 80386 and even the 80486 systems are also compatible, at least from a software point of view. Whether or not a system is really AT compatible depends on how the definition is established by a manufacturer. In general, though, "AT compatible" means that SOFTWARE that runs on

an IBM AT will run on an 80286-based system as well as an 80386 and 80486 system. If you examine most of today's software, you will find that it is clearly labeled as being "PC, XT, and AT compatible." Of course, you must be careful in reading the label because a lot of current software now requires a high-resolution video display (e.g., EGA or VGA), so the fact that it is generally compatible with the non-video system hardware is not enough. Some software requires a minimum of an EGA video system, so be sure to check that before you buy anything. If you need to know more about your computer's display system, you may find the *Powering Up* chapter on "Understanding Video Hardware" helpful.

Advances in technology have produced the 80386 CPU and the 80486 CPU. From a software perspective, Zenith Data Systems computers (e.g., the Z-386/16, the Z-386/25, the Z-386/33, and the Z-486) are still AT compatible. From a hardware perspective, the Z-386 and Z-486 systems are generally AT compatible, except for the fact that at least one system (i.e., the Z-386/16) must currently use Zenith Data Systems 32-bit memory cards for memory expansion because of the proprietary bus. That brings up one other point.

#### The Hardware Bus

What type of bus do you have in your computer? A BUS can be defined as the set of electrical connectors on the backplane or motherboard inside the computer. Simply stated, the bus contains the expansion "slot" connectors that are used for the boards you can plug into the computer, such as memory boards, video boards, and so on. In general, you will find four types of bus architectures.

The first type is the standard PC/XT and AT buses which are commonly known as ISA, which means Industry Standard Architecture. The second type is a modified ISA bus, which is generally proprietary to a specific manufacturer, and was required to support the 80386 and later systems because the ISA standard could not support those CPUs. This bus was developed independently by several manufacturers, including Zenith Data Systems and Compaq. This particular bus is used in the Z-386/16 and other similar computers.

Because the ISA bus could not support 80386 and newer CPUs, a new bus or at least modified standard was required. The third bus architecture was created by IBM for the new PS/2 series computers, and it is called the Micro Channel Architecture or MCA bus. At the time IBM developed this bus, they decided to get REAL finicky about licensing, royalties, patents, and copyrights associated with this bus. As a result, many manufacturers decided NOT to use this IBM standard bus, and a group of nine

major computer manufacturers created a standard bus called the Extended Industry Standard Architecture or EISA. The EISA bus systems are the fourth and last type.

The real distinction between the MCA bus (in the IBM PS/2 series) and the EISA bus is generally important only when one is looking at add-on boards for the system. Many boards developed for the ISA standard will work just fine on an EISA bus because there are still ISA slots available. That is also true of virtually all manufacturers modified ISA bus designs. My research indicates that is NOT generally true for the MCA bus in the IBM PS/2 series because the system was developed for an entirely different design layout. MCA boards, for example, will simply not plug into a slot in an ISA or EISA bus. I should also note that you will occasionally see software labeled as PS/2 compatible, but it will generally say that it is PC/XT/AT compatible as well. If software is ONLY PS/2 compatible, be sure to check that it will run on your system by asking someone.

The whole point of the discussion so far is that you really must have some kind of technical knowledge about your computer system. As you can see, the whole issue of compatibility gets very involved very quickly, and it frequently depends on whether you are talking about hardware or software. Hardware compatibility most often depends on the type of CPU a computer has, the type of bus used, the resolution of its video display system, and sometimes its clock speed and number of wait states. In some cases, it also depends on exactly which Zenith Data Systems computer model you have in terms of how much built-in expandability the system has. For example, the eaZy PC (all models) hardware configuration cannot be upgraded by using the usual add-on cards, such as memory or video, because it was designed to be a low-cost system. More on that in a minute.

#### ZDS Computer Classes

There are at least three things that all computer users should know about their systems: the CPU, the type of bus, and the type of video display. In some cases, knowing about the system clock speed can also be critical. However, knowledge of the exact CPU used in your computer is becoming more and more critical when selecting software, and the discussion of Zenith Data Systems computer models will be based on the type of CPU that each system has. Today, there are four classes of Zenith Data Systems computers: the 8088, the 80286, the 80386, and the 80486. To find out which class of computer you have, all you need to do is look at the "Specifications Page" in your Owner's Manual. Let's begin with the 8088-based computer models which will generally be discussed in the

order of release dates.

### 8088-based Computer Models

The very first compatible computer released by Zenith Data Systems was called the Z-151 which was basically an XT compatible running at 4.77 MHz. In addition, Zenith Data Systems also released the Z-161, which was a "portable" (I call it "luggable" because it was heavy) version of the Z-151, except that it had a self-contained CRT.

When a faster 8088 CPU was available, Zenith Data Systems released the Z-158, which looked the same as the Z-151, except that it contained a switch that allowed a user to choose between the 4.77 MHz and 8 MHz clock speeds. As far as I know, Zenith Data Systems was the first major manufacturer to release a production system that used the fast clock speed that is now commonly known as a "turbo" system. All of these systems were generally expandable and had at least four slots for add-on cards. A Z-159 was also released, and all of these I have seen have featured a 101-key keyboard and EGA video support. One model that received little attention was the Z-157, which was limited in the way it could be expanded (using a daughter board), and this system was later placed in a different "box" and called the Z-148. All of the Z-15x computers are generally known as the Z-150 series computers, even though the Z-157 was quite different in its internal hardware.

The Z-148 is a low-cost system with very limited expandability. Although it will accept some add-on cards (half size), it requires the purchase of an additional board, called a daughter board, to add a card. The traditional use of this expansion capability was to add a hard drive and controller. There were several different "versions" of this computer produced (some with floppies only and one with a hard drive), but it is generally referred to as the Z-148 or the Z-140 series. The Z-138 portable is essentially the Z-148 system hardware in a luggable box, and its expansion capabilities are also limited. This computer is sometimes discussed as the Z-130 series.

The Z-171, sometimes called the Z-170 series, was a "lunchbox-style" computer that was one of the first of the true portables, weighing in at about 15 pounds. It was not really a "laptop", unless your lap just happens to fit its shape.

Zenith Data Systems has developed a number of high-quality and highly-rated laptops that are generally known as the 180 series. These include the Z-181, Z-183, and Z-184. The model number generally defines what features were available on the system that was primarily related to whether the system had a hard drive. In the model number series, the Z-184 actually refers to the SupersPort computers, and there are

three different SupersPort models: the original SupersPort (with an 8088 CPU), the SupersPort 286 (with an 80286 CPU), and the SupersPort 386SX (with an 80386SX CPU). This is one example of where the actual series includes three different classes of computers.

Small, light computers are the current rage, but a few people may remember the ZP-150 laptop that was released a number of years ago. Its ROM contained some programs, like word processing (and of course MS-DOS), but it was not very popular, probably because its display was very difficult to read under most conditions. And of course Zenith Data Systems MinisPort computer is the latest in the super-small and light laptops, but it has not proven to be very popular in the last year or so. Although I suppose it is inevitable that I will miss one, Figure 1 is intended to include all of the Zenith Data Systems models. If you have one that I missed, I apologize in advance.

Before we go on, it is important to remember that these systems are generally compatible with all software which have a label of "PC or XT Compatible." Depending on the system, the level of hardware compatibility varies as previously discussed. There is one important exception that is listed in Figure 1, but does not actually have an 8088 CPU.

### The eaZy PC

From the letters I receive, the eaZy PC seems to be the most controversial system that Zenith Data Systems has ever released. The entire eaZy PC series (PC-1, PC-2, and PC-3) was discontinued some time ago. Unfortunately, it does not have the expansion capabilities of the more expensive desktop systems, and many eaZy PC owners have only learned about that AFTER buying one. My guess is that some discount firm must have obtained a rather large inventory of the discontinued eaZy PCs because I get a lot of mail with questions about them. The worst part of this particular unit is that whoever is selling them can't help and support the buyers, probably because the seller doesn't know anything about computers either. If you have an

eaZy PC, you might be interested in my comments about it in the "On the Leading Edge" column that was published in the April 1988 REMark. Back issues are generally available from ZUG, and I suggest you call to check on availability. But here are some things you must know about the eaZy PC, if you have one.

First, the eaZy PC contains a NEC V40 CPU, which is compatible with the 8088 CPU systems listed in Figure 1. The eaZy PC contains its own monochrome CRT (with 16 shades of gray) which provides CGA-compatible video resolution (640 x 200), but it cannot be upgraded to color because the video circuitry is part of a single board inside the computer.

The eaZy PC is completely limited in its expansion capabilities to those manufactured by Zenith Data Systems: a 128 K Memory Expansion Module (to increase memory to 640 K from the original 512 K), a Serial Port Module (required to support any serial device, including a printer), and a Modem Module. You may find one or more of these accessories as a single unit, if you are able to find them at all. I recommend that you buy the Memory Module, and the Serial Port Module may be important because the eaZy PC's built-in serial-type port is designed to only support a mouse, not a printer or external modem. Other than that, you cannot expand or add to the eaZy PC in any other way. You cannot add a color monitor, nor can you add ANY disk drives beyond what was included in the original system. Now let's move on to the more advanced systems.

### 80286-based Computer Models

The Zenith Data Systems 80286-based systems began life as the Z-200 series, and many people (including me) still refer to any system that contains an 80286 CPU as a Z-200 system. I have already mentioned the 6 MHz Z-241 and 8 MHz Z-248 as AT compatibles, but sometime after these were released, Zenith Data Systems changed the model number designations to the point that they are confusing to everyone. Not to be outdone, Heath also came up with some incredibly odd model numbers to describe the same computer available in

Series	Model/Description
Z-130	Z-138
Z-140	Z-148
Z-150	Z-151, 157, 158, 159
Z-160	Z-161 (usually included as the Z-150 series)
Z-170	Z-171
Z-180	Laptop. Z-181, 183, 184 (except SupersPort 286)
eaZy PC	PC-1, PC-2, PC-3
ZP-150	Laptop.
MinisPort	Advanced Laptop.

Figure 1. 8088 CPU PC Compatible Systems.

Model/Series	Description
Z-241	Large, desktop unit. 6 MHz.
Z-248	Large, desktop unit. 8 MHz.
ZW-286-25	Smaller desktop unit. 8 MHz. (Z-286/8).
ZCF-2326-EY	Compact desktop unit. 10 MHz (Z-286/10 or Z-286 LP). Same as Heathkit HS-40.
2500 Series	Large desktop unit. 12 MHz (Z-286 or Z-286/12)
5100 Series	Small desktop unit. 12 MHz.
SupersPort 286	Laptop with CGA display. 12 MHz.
SupersPort 286e	Laptop with VGA display. 12 MHz.

**Figure 2. 80286 CPU AT (Z-200) Compatible Systems.**

kit form. I won't even pretend that I know how to decode all of these strange numbers, but there is still an easy way for you to figure out what kind of system you have. As mentioned previously, you need to refer to the "Specifications Page" in your Zenith Data Systems Owner's Manual to determine the type of CPU in your system.

Sometime around the beginning of 1988, Zenith Data Systems began introducing several different "packages" containing an 80286 CPU. The first one that I found was the ZW-286-25 which was an 8 MHz 80286 system (Z-286/8) that appeared to use the same cabinet as the Z-150 series. Then, a 10 MHz "Compact AT Compatible" appeared that reminded me of the Z-148 cabinet with a different front plate and a strange model number of ZCF-2326-EY. Heath released the same computer as a kit which, for some reason, was called the HS-40. Depending on when you purchased this computer, it may be called the Z-286 LP (Low Profile) or perhaps the Z-286/10 (10 MHz). All of these systems were somewhat less expensive than the large Z-241/248, and they had fewer expansion capabilities and slots too. Their biggest advantage is that they used SIMMs (Single Inline Memory Modules) for memory expansion, and problems with having to use Zenith Data Systems memory boards were a thing of the past.

As 12 MHz 80286 chips became available, Zenith Data Systems and Heath released the "2500 series" which was a large desktop unit. As a Heathkit, it was available as the HS-2526. As an assembled Zenith Data Systems computer, it was available in various models such as the ZBO-2503-EK, ZBF-2526-EK, and the ZBF-2527-EK. This computer may also be called a Z-286 or a Z-286/12, depending on the documentation. The cabinet is virtually identical to the Z-386 system with four drive bays.

The SupersPort 286 is a popular and well-designed 80286-based laptop that runs at 12 MHz. It has CGA-compatible resolution, but it provides a high-resolution display (640 x 400) when used with its own LCD screen. The SupersPort 286e is similar, but has VGA resolution. Figure 2 sum-

marizes the various Zenith Data Systems and Heath 80286-based systems.

Now for the latest and greatest in the 80386-based and later systems.

### 80386-based and Later Computer Models

In about 1988, the original Z-386 was released, and it contains an 80386 CPU running at 16 MHz. Because of its zero-wait-state technology, it can still process as fast as other 80386 systems with higher clock speeds — the Compaq 386/20 for example. As more powerful and faster computers are developed, it is becoming more important to understand the relationship between the computer's clock speed and the number of memory wait states. When *Byte* magazine reviewed the Z-386 that was featured on the cover, the 16 MHz Z-386 was just a few hundredths of a second slower in processing speed than the Compaq 386 that was running at 20 MHz with one wait state. In short, the processing-speed penalty for using slower (and cheaper) memory chips becomes significant even with adding just one wait state.

In 1988, Zenith Data Systems released the 25 MHz Z-386 (Z-386/25), and a 33 MHz system (Z-386-25) with an EISA bus was released in 1989. The 25 MHz version is also available as a Heathkit called the HS-3629. And of course the 12 MHz TurboPort 386 with the page-white CGA double-scan screen was Zenith Data Systems's

entry into the high-powered laptop market. The SupersPort 386SX laptop was also released in 1989, and it features an 80386 SX CPU with a VGA screen. Figure 3 summarizes the Zenith Data Systems 80386 and later systems.

Zenith Data Systems has also released several new computers in the last year that use the 80486 CPU as shown in Figure 3.

### Are Model Numbers Important?

Sometimes, but not often, at least in terms of the articles in *REMark* and most other magazines that cover personal computers. How often have you seen an article about the model 99 or the model 339 computer? How about the model 30 or the model 50, or the model 70? What you do see are articles about the IBM PC, AT or the PS/2 series in general, but very few articles (unless they are hardware comparisons) will refer to the specific model number or even mention a series number. For Zenith Data Systems computers, you will occasionally see model-specific articles in *REMark*, such as the ROM installation on my SupersPort 286 (November 1989), but virtually all *REMark* articles are written by people who have done something specific with a Heath or Zenith Data Systems computer. For IBM computers, the model number usually doesn't matter, so long as the computer can run the software. For Zenith Data Systems and Heath computers, the model number is also not that important in many cases because the computers ARE compatible.

I mentioned IBM model numbers specifically because I have noticed that many new ZUG members have a preoccupation with computer model numbers for some reason. I suppose that many new members expect to see a specific article about the Z-159, Z-148, eaZy PC or whatever, but I continue to be puzzled by the fact that some computer users spend a wad of money on a computer and apparently don't understand that these are compatible systems. It's true that each model has some unique features, such as the Z-386/33E

Model/Series	Description
Z-386	Large, desktop unit. 16 MHz. (Z-386/16)
Z-386/25	Large, desktop unit. 25 MHz. (Z-386/25)
Z-386/33	Large, desktop unit. 33 MHz. (Z-386/33)
Z-386/33e	Large, desktop unit. 33 MHz, EISA bus. (Z-386/33e)
TurboPort 386	Laptop. 12 MHz.
SupersPort 386SX	Similar to SupersPort 286e, but 80386SX CPU
MastersPort 386SL	Notebook computer. 20 MHz.
3600 series	Large 80386 desktop unit. 25 MHz. (Z-386/25).
HS-5100-X	Small desktop unit with 80386SX CPU. 16 MHz.
Z-486	Small, desktop unit with 80486 CPU.
Z-486i	Tower unit with 80486 CPU for LANs.

**Figure 3. 80386 CPU and Later Systems.**

with the EISA bus, and some have specific limitations (e.g., the eaZy PC), but these systems are still compatible and will still run compatible software. As I mentioned earlier, I think it's unfortunate that many new computer buyers only find out after the purchase that a computer has some kind of limitation, such as the eaZy PC. Perhaps the only good approach is to say that that's the "cost" of education and chalk it up to experience.

In short, the exact Zenith Data Systems computer you have is not generally too important for most articles you will see in this and other similar magazines. The whole idea of buying a Zenith Data Systems compatible computer in the first place is to have reasonable assurance that you can use a wide variety of software on it, regardless of the exact model number you have. Computers that cost less, such as the Z-148 and eaZy PC, don't have the hardware flexibility and expansion capability that the more expensive systems have, but that should not be a surprise either. The cost of computers, like cars, is based on the type and number of standard features. A Cadillac costs more than a Chevrolet, but both will still get you from here to there. A Z-386 or Z-486 costs more than an eaZy PC, but both will do about the same job even though eaZy PC is a LOT slower. For the most part, virtually all of these systems will run still compatible software, regardless of which CPU is used. All of the computers listed in this article will run ZUG software that is listed under the "PC Compatibles" or "H/Z-100 and PC Compatibles" headings.

### The ZUG Membership

With a ZUG membership, you can buy Zenith Data Systems computer-related products through the ZUB Bulletin Board Bargain Center, not to mention receiving a year's subscription to *REMark*. Articles include information that is useful to the beginning as well as the advanced computer user ranging from "how-to" discussions (e.g., the "Powering Up series) to programming in various languages. Some articles include specific hints and tips on how to use various application software and utilities. And this "On the Leading Edge" series includes information about new hardware, software (including new releases of ZDS MS-DOS), and other information that is specific to Heath and ZDS computers.

If you have a question about your Heath or ZDS system, I strongly encourage you to write to us. Although the ZUG staff does not have every single Heath and ZDS computer model, we can usually help solve a problem because of our experience with these systems, including ZDS MS-DOS. If you do have a question or a problem, be sure to include the information mentioned at the end of this article so that we can provide a specific answer to help you.

If you have found something that you think needs to be shared with other Heath and ZDS computer users, I also encourage you to let us know about it. It may be something that solves a problem, or it may be about a software problem or bug. At ZUG, we do not use or test every possible software and hardware combination, and there may be some little-used software (especially networking-related software) or hardware that has a conflict with something

in a system. When I know about these kinds of things, they usually appear in this column with a credit (by name) to the provider of the information.

### Powering Down

If this particular issue of *REMark* is your first as a new member, welcome to ZUG. I think you will find that the articles published here will help you learn about and use your Zenith Data Systems computer system more effectively. Although it is my usual practice to list the models and prices at the end of each article, they are not included for the general discussion of model numbers because many of the older systems mentioned have been discontinued.

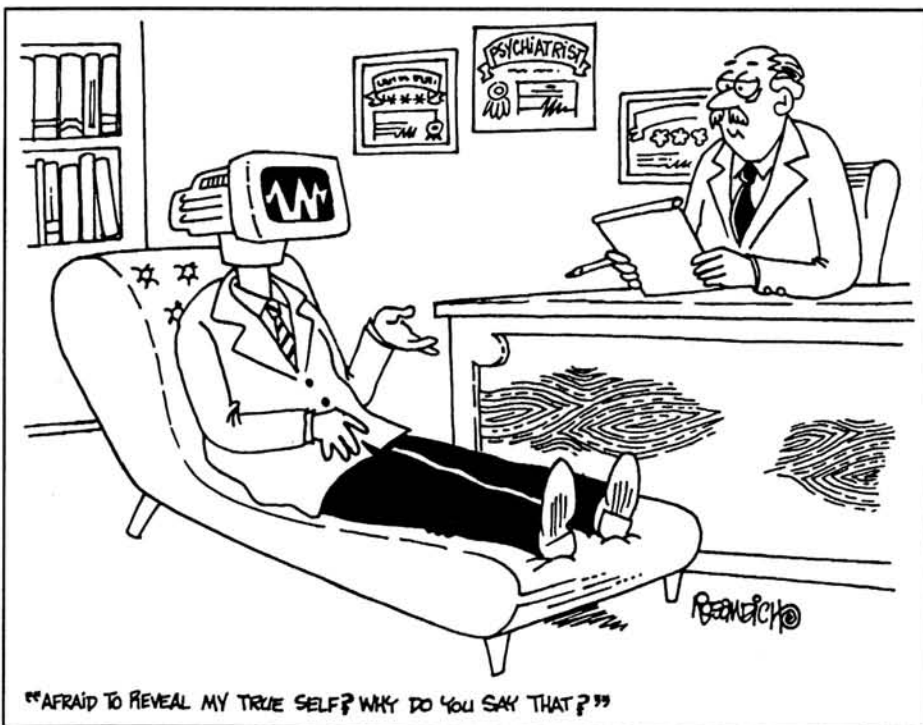
As I mentioned earlier in this article, one of the purposes of this column is to review the latest versions of Zenith Data Systems MS-DOS. As I was finishing this article, I just received my version 5.0 upgrade, and one of my columns in the near future will discuss what's new in that version.

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

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

### Products Discussed

*Powering Up* (885-4604) \$12.00  
 Zenith Users' Group  
 P.O. Box 217  
 Benton Harbor, MI 49023-0217  
 (616) 982-3463 ✻



---

---

To Create a REMark Article:

# Dial Up Your Background Material

**Harold C. Ogg**  
**357 W. Diversey Avenue**  
**Addison, IL 60101-3508**

There's an old saying about genius being composed of ten per cent inspiration and ninety per cent perspiration. In addition to being an information scientist, I've been a professional researcher/librarian for the past twenty years. I can attest to the aforementioned statistics and, as reflected in the several articles REMark has been gracious enough to publish when I have submitted, I can assure any potential authors that the effort ratio is relatively stable no matter what subject you decide to profess.

REMark is not the only publication to which I offer my occasional prose, but it is quite typical of those in the technical field. The "How to Submit an Article to REMark" section in the January, 1990 issue is quite clear in the types of material REMark favors, and this is evident in the articles of code examples, hardware routines, experiences, and fixes presented in each issue. There is a common denominator in them all, and that is the matter of background research. It is with the latter that authors exhibit the "ninety per cent perspiration" factor and gather background material. It was once said that by the time you have all your materials and notes assembled, your paper (or essay) writes itself. The trick to all this assembly is where to find references to your subject and how best to get everything organized.

Whenever I get an idea for a REMark article (or one for any other publication, for that matter), I consider two important elements: has anyone else recently written about the subject, and have I uncovered all the relevant background material? Unless I have a unique approach to a topic, I don't want to repeat what someone else has just accomplished. Also, once I have decided to proceed with an article, I want to be

certain that I've covered all the bases. There is nothing more embarrassing than submitting a manuscript, only to discover that, because of your own carelessness, you omitted references to one or more programs, articles, or studies on your topic. Worse yet, there is the sinking feeling you get when your essay appears in the current issue of a magazine and a reader sends a letter to you (or to the editor) gleefully pointing out some glaring omissions you made. I learned my lesson some time ago with a bibliography of periodicals for C language programmers which I had written. The editor liked my work, but was quick to point out, after the article hit the newsstands, that I had omitted references to the publications of two of his bigger (and most respected) advertisers!

The problem is defined by the question, "How can I be certain that I've done exhaustive research? And further, "How can I make time to examine all the sources of information available to cover the bases?" The answer is that there is no foolproof method, or at least, not a conventional one. But as long as we're all writing about computers, let's use our machines to ensure the quality of our work. We're surrounded by dozens of data bases filled with all the reference information to guarantee the exhaustiveness of our background studies. The trick is in knowing where to find them and how to search once we're online. The information is closer than you think, and it all may be no more than a telephone call away.

## Dial Access to the Rescue

The idea of "electronic information exchange" is not new. Among the first distributed applications for the H-8 computer was a program for linkup of the

computer to amateur radio hardware, to make communications more versatile. In fact, if you consult the latest REMark, you'll see that there are several CP/M, HDOS, and MS-DOS based utilities in the ZUG (Zenith Users' Group) software price list for amateur radio users. These programs assist your shack in various ways from allowing you to maintain electronic log-books, send and receive Morse code, and in general, communicate more efficiently. The keyword is "information," that is, the gathering and disseminating of facts. The aforementioned is really the predecessor of dial access, and in this article, I'll show you how, with the addition of a modem and some elementary software, you can be on your way to efficient research and authorship for REMark, or even for other journals of your choice.

A couple of winters ago, I received a phone call from a publisher for whom I had done some writing in the recent past. He explained that his company was creating a user's manual on DOS, and he needed a comprehensive listing of current reference books on the subject. The problem: could I deliver a manuscript to him within two weeks? He needed at least thirty titles with annotations, and preferably as many as fifty. I agreed to the project, immediately afterward looking out the window at the foot of snow on the ground and not much less on the roads. I was going to have to visit a dozen or so libraries and as many bookstores in the middle of this Midwestern winter. The problem was: which libraries would have the material I needed to examine? I was beginning to have second thoughts about the deadline to which I so readily agreed.

Fortunately, my situation wasn't so acutely desperate. A phenomenon of the

Welcome to ILLINET Online at University of Illinois-Urbana

Select Type of Search

1. Subject
2. Title
3. Author
4. Author/Title
  
5. Call number, ISBN, etc.
6. Direct command mode

TO START: Enter a number from above and press <ENTER>.

Type the letter I and press the <ENTER> key for instructions.

IODIAL22 ----- Press <ENTER> key after making choice ----- NMI100M1

X - Exit      I - Instructions  
==> \_

F1: Help | F2: Main Menu Functions | F3: Session Menu | F4: Rotate ...Alt-Tab

Figure 1

past few years is the so-called dialup facility provided by most libraries. I'll condense the process here briefly and go into details later in the article. Armed with a list of agencies with dialup ports on their computerized book catalogs, I set my communications software to "capture" mode and began dialing. The process is no different than ordinary bulletin board linking and should be familiar to readers who use the ZDS-COM1 bulletin board to download files and share information with other ZUG members. After a few hours of searching, I had about seventy kilobytes of usable background material on disk. A couple of the larger bookstores also had bulletin boards for customers, and I was able to identify some of the newer DOS titles from their inventories. The only "legwork" with which I was left was to visit a few of the bookstores that didn't have telephone linkups. When the titles I found began to get redundant, I knew that my research could be considered "exhaustive." I arranged and classified my notes and wrote the manuscript, which I sent to the publisher two days early. He was pleased, and my bank account was a few dollars the better for the efforts.

### Your Computer Setup

To access the majority of remote dialup facilities, the only additional accessories you need to go online with your PC are a modem and a communications software package. While the modem generally should be Hayes compatible, the choice of communications software is largely a matter of personal preference. I've seen some public domain packages that were as good for dialup purposes as programs costing \$200.00 or more. The deciding factor is generally not what's inside your PC, but what's on the other end of the telephone line. There is no one standard for dialup

(i.e., remote) computers, so you have to make the machine on the other end "think" your PC is a terminal friendly to its (the called computer's) environment. Thus you enter the necessary (and sometimes complicated) genre of emulator software.

The concept of emulator software is simple: by redefining the keys on your PC keyboard to communicate with a remote computer, you can fool the dialed computer into thinking you're communicating from a VT 100/102 terminal, an IBM 3101, or any number of others. For example, if you're dialing into one of several IBM mainframe-based services, you may need to send one or more of the so-called "PF key" sequences to the host operating system. My software, BitCom, for example, requires a two-key sequence, Esc-3, to send the PF3 code to the host in order to move through menu levels of one of my more frequently dialed data base's operating systems. It generally doesn't matter if you're using an 88-key (PC) or 101/102 key (AT-style) keyboard; most emulator software

will make provision for all the special codes for you. The only typical hassle is that you often must keep a keyboard map template beside your computer; some of the special key sequences are not obvious or intuitive, and each communications software author has his/her own preferences for keymapping. But with a minimum of practice, you'll soon memorize the more frequent key sequences and the alternate keyboard schemes will become second nature in your daily computing sessions.

Complication arises in circumstances where you access a number of online services and data bases, as do I. It may be necessary for you to shift between two or more communications programs from time to time to change to a less common keyboard emulator required by an atypical data base. As an example of what you may encounter, BitCom ships with an overlay file for an IBM 3101 emulation template; SmartCom III,<sup>1</sup> however, does not, even though the 3101 terminal is popular. You can, of course, write your own keyboard emulation code for either program if you want to take the time to do the programming and remap all the necessary keystrokes. TeleVideo 920, Wyse, and IBM 3278 terminals are among commonly required emulations on dialup computers. Communications software is beyond the scope of this article; you'll need to shop around for the modem utility that has most (or all) of the emulator features you need for the repertoire of data bases into which you'll dial in your part of the country.

BitCom, incidentally, currently ships version 4.0 as BitCom Deluxe. A previous release, version 3.6, on a 5-1/4" disk was included in On-Line with BitCom,<sup>2</sup> a book which includes much information about online database access similar to that found in this article. For the price of the book, the software is essentially free and is upgradeable. The book also includes information for writing your own terminal emulator and script files. Please be advised that

Welcome to the Learning Resources Center of the College of DuPage

You may search for library materials by any of the following:

- A > AUTHOR
- T > TITLE
- W > WORDS in the title
  
- S > SUBJECT
- C > CALL NO
  
- R > Search RESERVE Lists
- I > Library INFORMATION
- Q > QUIT

Choose one (A,T,W,S,C,R,I,Q)

Please see the Reference Librarian for Assistance

Help F1 Option Menu F2

Connect: 00:00:40

Figure 2

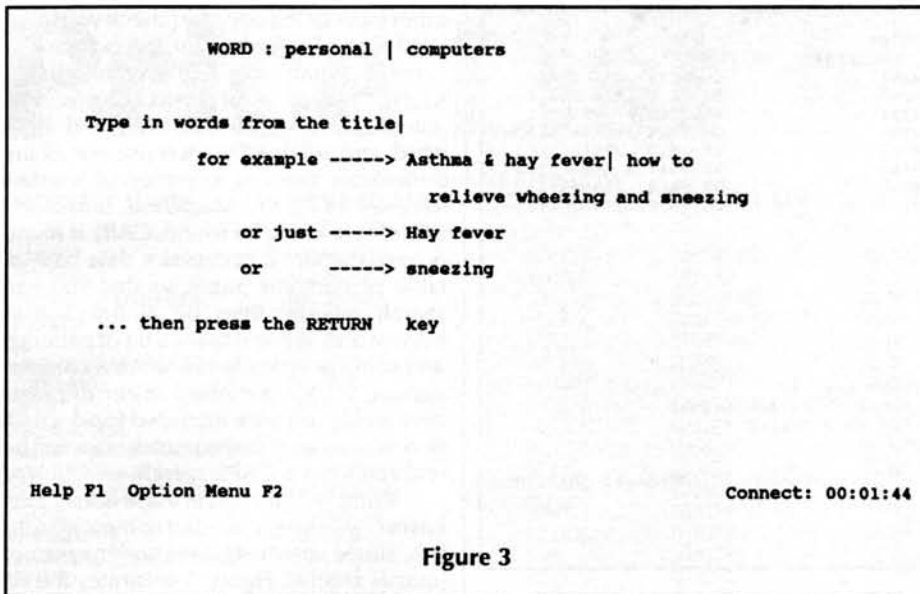


Figure 3

I am not advocating SmartCom and/or BitCom as the only communications programs suitable for dialup accesses; it's just that these two programs offered the cleanest screen dumps for the purposes of illustrations in this article. Figure 1 shows the initial query screen for the ILLINET data base (described below), using SmartCom III as the communications program, and Figure 2 shows the same screen for the Innopac data base using BitCom.

#### Database Sources

How do you find informational data bases into which to dial? Your most likely, and most accessible sources are the public and academic libraries in your immediate and surrounding communities. There are usually no password or membership requirements, and almost anyone can invoke a browse mode. The information is free for the taking. In the long run, you may be asked to register as a library patron (for which there may be a fee) or, in the case of schools, you generally must be a registered student or alumnus if you want to take books out of the building. However, most libraries will allow you to use materials you previously located on the computerized data bases in house as much as you like.

Since my home is the Chicago area, I'll use my own metropolitan area as typical of how you might use a dialup service. Residents of smaller metro areas note: you can still dial into data bases of areas other than your own (although you're going to run up more telephone toll charges). You can write to metropolitan area libraries for brochures describing their dialup services, or you can consult the Dial In 1990-91<sup>3</sup> directory (it's updated annually) for a nationwide list of library dialup services. For readers in New York, Los Angeles, etc., the following discussion will work for you as well — just call your local library for a copy

of the telephone connect numbers and the modem setting protocols, and you're in business!

Illinois divides the State into eighteen Library Systems (Chicago by itself is a single system) for the purpose of resource sharing. What this means is that the majority of member libraries (typically public and corporate) pool the records of their book and materials collections into a dedicated cluster computer. If you have the dialup number of the your local system's headquarters computer (it is available to the public) and dial into the mainframe with the proper modem settings (also available for the asking), you not only find out whether a book on a particular topic is owned by any library in the cluster but also whether it is currently on the shelf of the owning library. The data bases of some library clusters have sufficient detail for screen entries that, if you're attempting only to determine whether a book(s) on your intended topic exists or

has been previously written, you'll get what is called "full bibliographic entry" — the author, title, publication date, and sometimes partial contents of the book you call up. You can almost always search by title or subject, many times by keywords in the title and sometimes in Boolean fashion. With a bit of practice, you can prepare an exhaustive list of resources using about a half hour or so of connect time. Figure 3 shows a screen from the Innopac data base set up for a keyword search of the terms "personal" and "computer." Note the '|' symbol between the two terms denoting the logical "AND" Boolean operator.

One thing you should ascertain — and this is true of most areas and states — is whether significant libraries (usually the larger ones) have decided to "go it alone" and form their own standalone data bases or spinoff clusters. You can ascertain this when you call a local system office to obtain telephone access numbers and modem settings — the systems maintain lists of offline data bases since, even though hardware might not be shared between system cluster members and nonmembers, information generally is. The decisions of local libraries not to cooperate with neighboring institutions is typically more politically motivated than otherwise, and professional courtesies regarding information exchange take precedence. You should also obtain the hours of operation of the data bases into which you intend to dial. Many libraries shut off their systems several hours per week for performing system maintenance, and others shut down for any hours during which the building is not open. This measure, of course, saves the dialup host some budget money in sysop costs, but it wreaks havoc with searches you intend to perform after peak hours. It's a bit disheartening to get the "System not Available" message when you find your creative juices peaking at three o'clock in

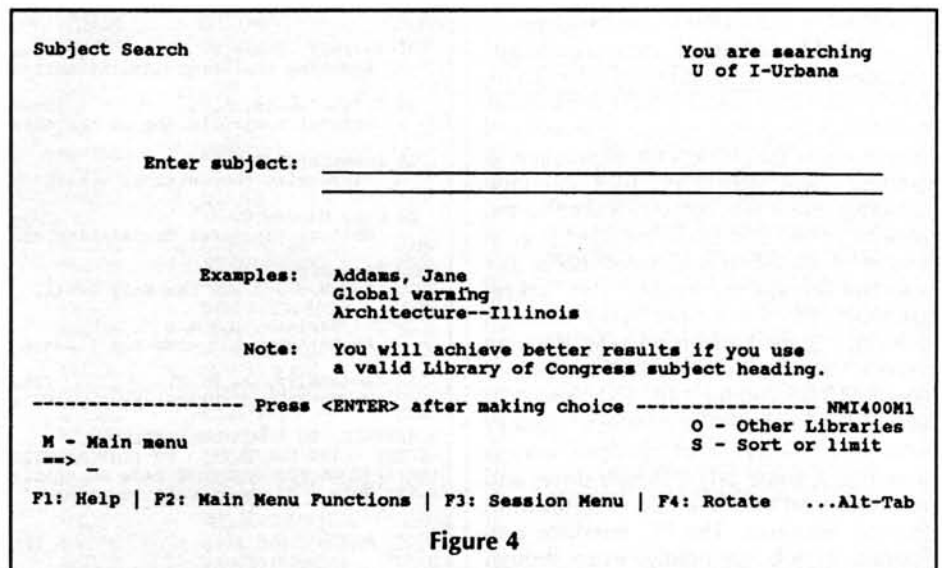


Figure 4

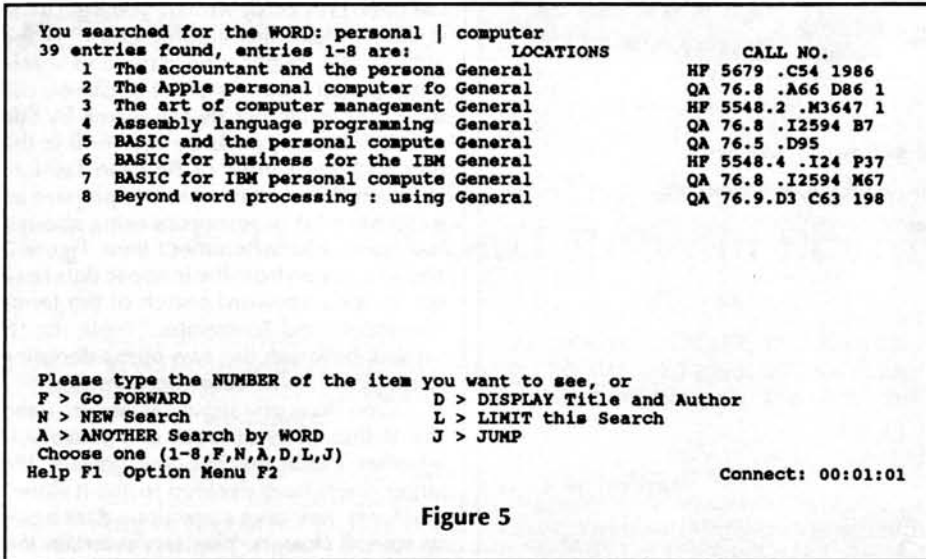


Figure 5

the morning during an insomnia attack!

Illinois separates its university data-base system from the public/corporate library data bases. A system called ILLINET Online, based on the holdings of the University of Illinois' computerized catalog, forms the foundation of the system of thirty-five college and university library collections statewide. Only Northwestern University, Loyola University, and the University of Chicago are the major nonparticipating institutions — they have their own standalone data bases, but they will give you their telephone access numbers and modem protocols along with user brochures on request. Otherwise, Illinois covers the state with mostly local (although non-800) telephone access numbers, and you can dial in from anywhere in the state (or nation). User brochures<sup>4</sup> are available with a list of dial access numbers, and, although the full ILLINET user manual is itself a very lengthy and complicated document, you can still browse the entire data base with only the dozen or so commands described in the ILLINET user brochure.

ILLINET provided two significant changes for dialup users in 1991. Until mid-1991, a dialup user had to access materials in the data base via a system of shorthand search terms that took a bit of practice to master. There was a PC user interface available, but it was not distributed to the general public and could be used only in house, at public access terminals in the member schools themselves. This was regrettable, for my former work location (Chicago State University) was deriving several years' additional life from retired Heath/Zenith model 148, 151, and 159 computers because the interface required only 512 kilobytes of random access memory, a single 5-1/4" floppy drive, and the serial port to cause PCs to perform as "smart" terminals. The PC interface performed its job splendidly, even though

written in interpreted BASIC, but has since been replaced with a user friendly interface which now runs directly off the mainframe. You can still use the shorthand commands if you wish, but the user interface is available now to anyone using the dialup facility as a program resident on the mainframe. There's a similar difference in difficulty of use as between programming in assembly language and in a high level language like COBOL. You can have either speed or ease of use, but not both. But either search method works equally well, and choice of shorthand commands vs. the interface makes no difference in the end result of the research you would perform for your authored works. Figure 4 shows a screen dump of a second-level workscreen of ILLINET, using the online interface, ready for a subject search of the data base.

The other unveiling made by ILLINET this past summer was the online version of CARL UnCover. CARL is a copyrighted program which may also be available in

other parts of the country (check your own local library), which is for the purpose of accessing magazine and journal articles. ILLINET's version of CARL begins with journals dated from mid-1989 and after-ward, and, while REMark is not one of the periodicals indexed, commercial journals such as BYTE, PC Magazine, InfoWorld and others are to be found. CARL is menu driven; therein, it accesses a data base of table of contents pages so that you can search articles' titles by author and/or keyword(s). While it takes a bit of patience and some practice to compose a comprehensive list, for a current roster of magazine articles on your intended topic, confidence is assured that completeness will be realized from a CARL search.

What will you find in these library data bases? I've already alluded to monographs (i.e., single volume books) and magazine/journal articles. Figure 5 illustrates the resulting list of the "personal | computer" word search from Figure 3, and Figure 6 recapitulates the list of articles found in CARL UnCover using the phrase "optical character." However, you'll also find anything else that the member libraries have over the years entered into their machine readable catalogs. This would include such items as degree theses, research papers, conference proceedings, government documents, and perhaps even source code listings for computer programs. You also may discover non-book materials such as audio/visual media, computer software, and CD-ROMs. A hint: if, after searching an online data base you wish to examine the materials you've identified, check for two things. First, there is usually an indication for each entry whether the material is "on shelf" or "in circulation." The latter (or a variation of it) means that someone has borrowed the book and not yet returned it. If you have borrowing privileges at the library whose data base you've searched,

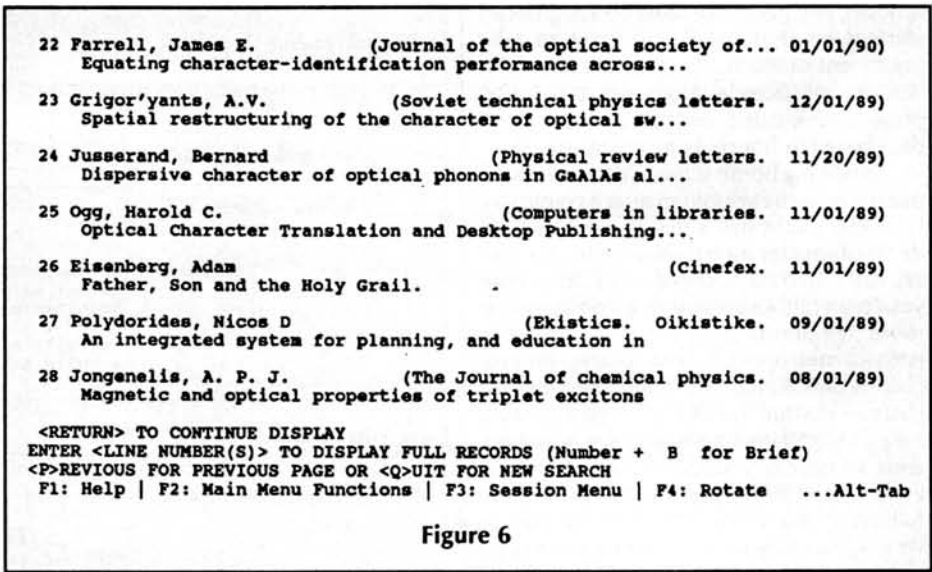


Figure 6



you can generally call the library and place a "hold" or "reserve" on any book currently on loan to another person. If you identify an item at a distant library, you can ask for the item at your local library through an "interlibrary loan" arrangement. The other thing to watch is whether the entry indicates the designation "REF," "reference," or "REF ONLY." This indicates that the item you've identified is for use solely in its owner's building; you can't check it out even if you have borrowing privileges there. However, this also means that there is a high possibility that the book is available most of the time; this is also usually the designation given journals and magazines, a fact which makes your research easier and less time consuming.

Some libraries have dialup services that you may not be able to access directly. BRS (Bibliographic Retrieval Services), Dialog Information Services, Dow Jones News Retrieval, ERIC (educational papers and documents), and Psychinfo, among the more popular, may be available to you but only for a fee. The reason for the on-site charge is that the connect times are relatively expensive due to subscription prices being added to telecommunications costs. The library offering these services usually performs the dialup searches for you based on your written application, in which you state a search stratagem and relevant keywords. You may also be asked to pay for printouts of detailed summaries of the materials found. Some data bases such as OCLC and WLN, while popular in libraries themselves, are not available on direct dialup at all. These services are in-house facilities from which libraries obtain detailed information for their own local electronic book catalogs. They are usually duplicative of what is available to you through modem dialup, and you shouldn't be concerned if you're not invited to access them.

It is becoming increasingly popular for some of the former mainframe-based data bases to be made available on CD-ROMs. ERIC and BRS (mentioned above), as well as D/SEC (often known as Compact Disclosure, a business information data base), standard reference books such as Grolier's Academic American Encyclopedia, publications of the U.S. government, and many others are now published on compact discs. Your local library may make these services available through a dialup facility, or, if not, at least via a user accessible PC (i.e., public access terminal) in the library itself.

#### Other Sources of Dialup Information

Consider also the commercial dialup services as potential resources. PRODIGY, for example, recently announced the availability of an online encyclopedia. If you don't have a CD-ROM drive on your computer (or for that matter, a collection of CD-ROM based reference tools) or cannot get

to a library where an encyclopedia on this medium is available, it might be worth the connect time to perform a keyword search in PRODIGY's offering as a starting point in your research. CompuServe and Genie are also worth searching, and they both include downloadable programs and utilities similar to those found on ZDS-COM1 for the asking. Both CompuServe and Genie also run special interest forums (Genie's seems to be more "chatty") wherein, if not strictly for factual information, you can join a forum to see what others are saying about a new topic and exchange ideas that might have a bearing on the format and content of your planned article. BIX (Byte Information Exchange), ZiffNet and MagNet (which is now a subset of ZiffNet) are set up more for making available their publishers' utility programs than for providing services, but there is still a wealth of information available in those data bases. Browsing through the aforementioned dialup services, you'll find programs that appeared in PC Magazine and in such paperback publications as PC DOS Power Tools, which, as the name of the latter implies, are for power programmers and authors. For the more technical articles you'll write (which are, obviously, more saleable), the above are online sources you'll want to examine to ensure comprehensiveness of your research before releasing your manuscript to a potential publisher. A list of the various dialup services appears at the end of this article.

If you are fortunate to work for a university or other large government or corporate agency, BitNet or Internet may be available to you. These are but two of several international information sharing networks, based primarily on a very sophisticated electronic mail (E-mail) setup. If your aim is to determine whether someone has done some work in the area you're researching, these are the networks of final discovery. There are generally membership considerations for joining one of these networks, but the comprehensiveness is worth the effort. The User's Directory of Computer Networks<sup>5</sup> details several academic exchange networks, including AARNet, THEnet-DECnet, SPAN, and others, with names of operations personnel, (electronic) addresses, and information on user organizations.

#### How to Get the Most Benefit from a Dialup Service, and How to Organize What You've Found

If you dial long distance, have your search strategy planned in advance. Most of the commercial information services (i.e., CompuServe, PRODIGY, Genie, discussed above) have local access numbers available via one or more national telephone network setups such as Telenet, Tymnet, and Dialnet. A charge for accessing these supplemental telephone services

may be added to your time connect bill, if not to your telephone bill itself. It is handy to use a communications program that displays elapsed connect time on your monitor's screen (see Figures 2, 3, and 5 for examples) so that you don't "forget" and let time melt into huge telephone charges.

One final note: although this article doesn't attempt to teach you how to compose your REMark article (you'll have to take a writing course for that purpose), I would like to conclude with some suggestions on how to manipulate all this dialup material you've found. There is a fair amount of public domain software you can use as an "electronic file card" system; Magbase<sup>6</sup> in the ZUG software collection is a good example. And other inexpensive programs are available: BIBL, The Reference System, Library Card, Biblio, and The Political Researcher's Assistant are among several writers' tools available from shareware vendors.<sup>7</sup> Such utilities can aid your writing efforts in two ways: the programs can organize your article references themselves into a data base, and they can serve as outlines for the actual narrative once you're ready to write. You may also discover some of these shareware utilities on electronic bulletin boards in your area. You'll find a considerable amount of redundancy of utilities from one bulletin board to the another, but if you're perseverant, you'll run across the occasional unique program that provides the finishing touch to your article's research — and gives your writing a sense of closure and completeness.

#### Sources and Programs Mentioned

<sup>1</sup>SmartCom III, published by Hayes Micro-computer Products, Inc., 705 Westech Drive, Norcross, GA 30092. \$199.99.

<sup>2</sup>Smith, Bud E. On-Line With BitCom. New York: Bantam Books, 1989. Paper, 333 pages, \$39.95 (includes a 5-1/4" diskette).

<sup>3</sup>Schuyler, Michael, editor. Dial In 1990-91: An Annual Guide to Library Online Public Access Catalogs in North America. Westport, CT: Meckler Corporation, 1991. Paper, 228 pages, \$49.95.

<sup>4</sup>The Illinois Library Computer Systems Organization. ILLINET Online: Dial Access. Available from ILCSSO Office, Suite 205 Johnstowne Centre, 502 E. John, Champaign, IL 61820. Free.

<sup>5</sup>LaQuey, Tracy L. The User's Directory of Computer Networks. Austin, TX: The University of Texas System, (published by Digital Press). Paper, 630 pages, 1990.

<sup>6</sup>Magbase, available from Zenith Users' Group, P.O. Box 217, Benton Harbor, MI 49023-0217. Part #885-1249-[37] for CP/M, \$25.00; part #885-3050 for H/Z-100 and PC operating systems, \$25.00.

<sup>7</sup>Two sources for shareware utilities for writers are: Public Brand Software, P.O.

Box 51315, Indianapolis, IN 46251, tel. (800) 426-3475, or (317) 856-7571 in Indiana; and The Software Labs, 3767 Overland Avenue #112-115, Los Angeles, CA 90034, tel. (800) 359-9998. Catalogs are available from both vendors.

Some of the more popular online dialup services (in order of appearance above) are as follows. Check with the individual vendors for current pricing.

**PRODIGY**  
445 Hamilton Avenue  
White Plains, NY 10601  
(800) 759-8000

CompuServe Information Service, Inc.  
5000 Arlington Centre  
Boulevard, Columbus, OH 43220  
(800) 848-8990  
(614) 457-8650 in Ohio

Genie  
(GE Information Services, Dept. 02B)  
401 North Washington  
Street, Rockville, MD 20850  
(800) 638-9636

BIX  
(Byte Information Exchange)  
One Phoenix Mill Lane  
Peterborough, NH 03458-9990  
(800) 227-1983

ZiffNet  
One Park Avenue  
New York, NY 10016  
(800) 346-3247 ✻

Continued from Page 36

the card for \$99 with no memory. I called AST and asked if I could use 70ns 1 MB SIMMs instead of the 100ns chips recommended in the book. They refused to give me an answer. I decided to buy two 70ns 1 MB SIMMs anyway for \$40 each. I took out the 128 KB memory card, back-filled the 512 KB to 640 KB according to the instructions. My machine came up. I tried to enter Windows and got a memory parity error. I had to do a cold reboot, and when I tried to run mem/c to take a look at my memory configuration, I received another parity error check. I called AST. They told that the 70ns chips were the problem. I exchanged them for 100ns chips and had the same problem. I called AST. They now said that their board did not work with Z-248s (I had told the first technician that I had a Z-248) since the parity bit was opposite of what was on my original 512 KB of memory (bit 0 vice 7). I put back in my 128 KB memory card and decided I didn't need a memory upgrade after all.

#### CPU Upgrade

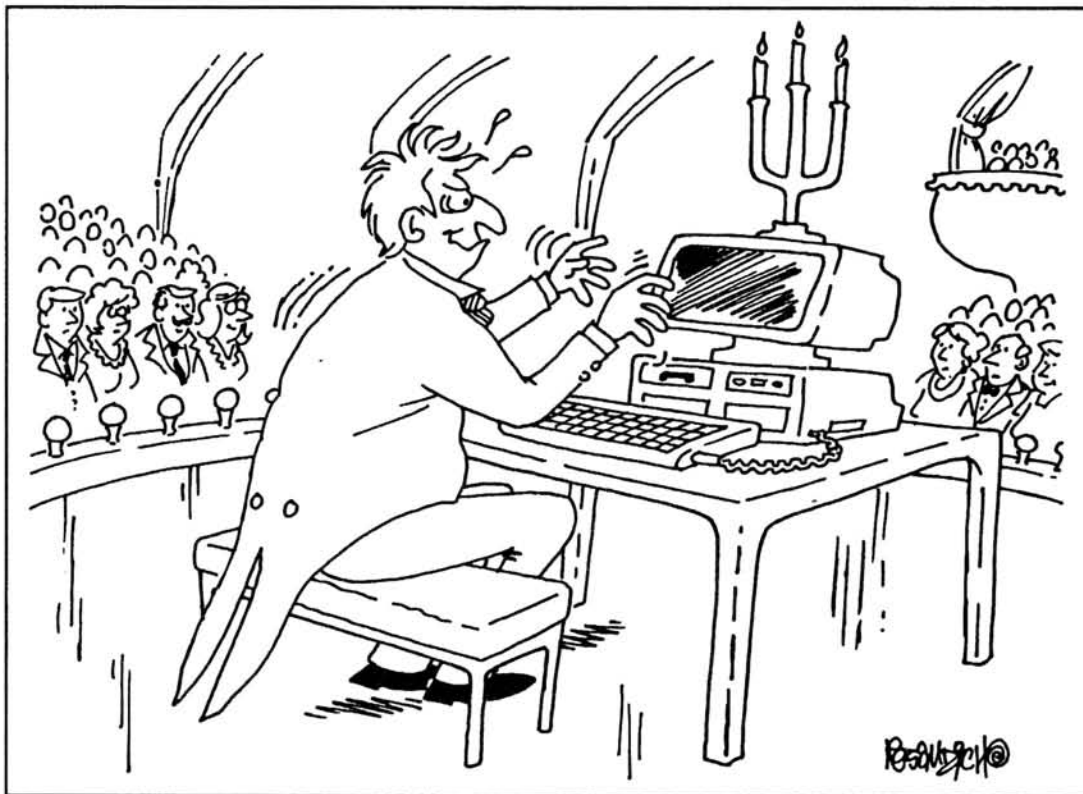
For my last upgrade, I purchased an

# Classified Ads

**FOR SALE: COLLECTORS ITEMS:** H-89 8-BIT COMPUTER W/DISK DRIVE AND H-77 DUAL DISK DRIVE. PERFECT CONDITION. ALL NOTEBOOKS, MANUAL, PROGRAM DISK, EXTRA DISK, ETC. CALL (312) 254-4029 OR WRITE W. DAVENPORT, 4516 ALBANY AVE., CHICAGO, IL 60632.

**ZENITH DATA SYSTEMS CP/M-85 DESKTOP COMPUTER.** KEYBOARD; MONITOR; SOFTWARE, GW-BASIC 2.0, MS-DOS VERS. 2, PASCAL COMPILER, MULTI-PAN, FORTRAN COMPILER. CALL (607) 589-4514. \$350 TAKES ALL.

SXZ-248 from Technology Specialists, Inc. (3110 Columbia Pike, Suite 303, Arlington, VA 22204, phone 703-521-1886 or 1-800-4-UPGRADE). This has turned my 8 MHz Z-248 with 640 KB (?) of memory into a 20 MHz 80386SX with 4 MB of memory, but that is another story. I finally was able to get rid of the 128 KB memory card. ✻



A decorative border of stylized yellow and orange flames surrounds the central text.

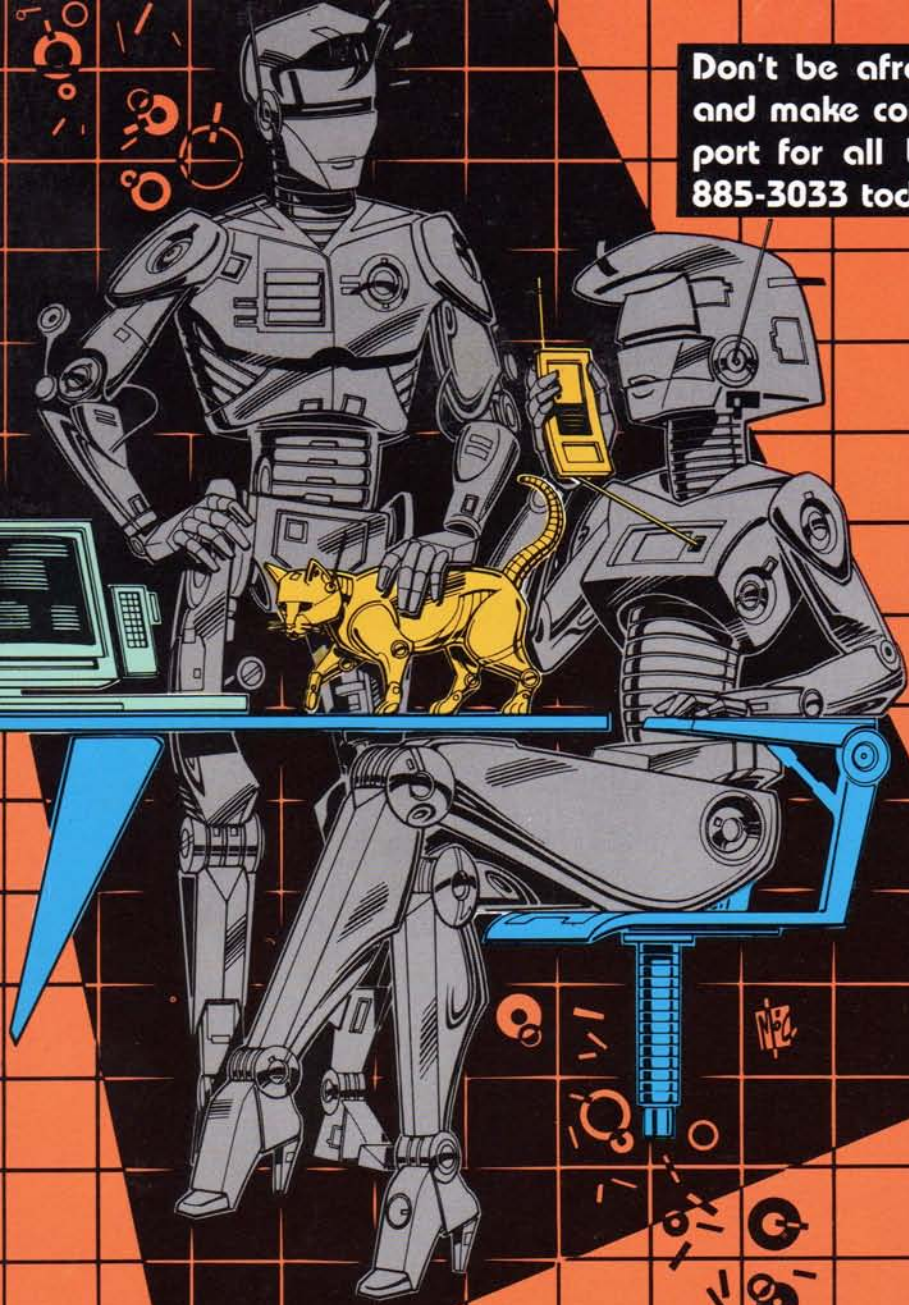
# HADES II

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

- Sector Display/Editing
- Sector HEX/ASCII String Search
- File Display/Editing
- Physical and Logical Cluster Display
- File HEX/ASCII String Search
- Drive Parameter Display
- 512 MegaByte Drive Size Limit
- File Attribute Display/Edit
- Automatic Erased File Recovery
- Manual Rebuild File Recovery
- Works with Headerless MS-DOS Disks
- PC-Compatible or H/Z-100

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

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



```
HUGMCP Commands
F1 -- Points This List, Your Storage Buffer Size, And How Many
    Bytes Are Presently In The Storage Buffer.
F2 -- Allows Sending A Defined Message, Or Character Sequence.
    These Messages Are Entered Using The (F6) Setup Command.
F3 -- Toggles The Storage Buffer On and Off. When The Buffer
    Is On, The (Ctrl) On The 25th Line Will Be High-Lighted.
F4 -- Allows Sending Data To Disk From The Storage Buffer, Or
    Directly From The Modem By Use Of XMODEM Protocol.
F5 -- Allows Sending Data From Disk, Using Either XOM-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 MC-DOS.

Storage Buffer = 524288 Bytes
Storage Buffer Usage = 0 Bytes

Select Message (A-0), (F1) To List, Anything Else To Abort --> _
F1=Hlp F2=Msg F3=BufR F4=Sw F5=End F6=Cfgr F7=Cle F8=Print F9=Exit CM
```

```
HUGMCP Configuration Help
This function allow the baud rate to be changed. Depending upon which
mode you are in, normally, it would be either 300, 1200, or
2400 bps. Select a number to work properly.

This function allow you to change the word parity. Normally you
select a number to work properly.

This function allow the changing of the word length. Normally the
length would be set to 8 data bits. This value is acceptable to most
terminals, and is necessary for XMODEM Protocol to work properly.

This selection allow you to enter messages which can be automatically
sent with the (F1) key. Up to 24 character messages can be saved.
Selection 0 is special. It allows certain help completion. (1) Number
and function selection. (2) None (empty). (3) Select by key. Auto-
matically be sent when the program is first executed by selecting the
proper option during setup.

Type (F6) for more help. Anything else to continue:
F1=Hlp F2=Msg F3=BufR F4=Sw F5=End F6=Cfgr F7=Cle F8=Print F9=Exit CM
```

```
HUGMCP Configuration Menu:
--> Modify Baud Rate
--> Modify Parity Type
--> Modify Word Length
--> Modify Or Add Auto-Messages
--> Miscellaneous Functions
--> Change Screen Color Assignments
--> Riping Current Configuration
--> Make Changes Permanent

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

Baud Rate: 1200
Parity: NONE
Word Length: 8
Bps/Chr: 10/1
Response To Unhashed Disables: NO
Storage Buffer Data Parity Bit: SET TO ZERO
Send Modem Initialization Text: NO
Delete Character: ^@^M
Modem Port Set To: COM1

F1=Hlp F2=Msg F3=BufR F4=Sw F5=End F6=Cfgr F7=Cle F8=Print F9=Exit CM
```

**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 885-2144