Software Tools

OK, you asked for it, an issue dedicated (mostly) to software.

The Art of Disassembly  page 8
Using Sourcer to create commented, assembly source from object files.

Handling Interrupts  page 16
With Any C
Hacking Sprint:  page 36
Creating Display Drivers

Annual C Reviews
Comparing the latest, greatest C compilers.

And More . . .
Turning A PC Into An Embedded Control System  page 24
Bringing Up A Surplus  page 28
68000 Board
Part 2 of Karl Lunt's cheap 68000 project.

Plus:
Practical Fractals  32
Shareware you must register  62
And Much, Much, More
Aztec C ROM Cross Development Systems
Produce Fast, Tight C Code with Less Effort

Aztec C ROM Cross Development Systems give you the best results — clean, tight and fast running code. Aztec C systems are available for a variety of targets and for both MS-DOS or Apple Macintosh hosts! And, Aztec C systems come complete with all the tools to edit, compile, assemble, optimize and, now, source debug your C code in less time and with less effort.

Quality, tight code that’s fast and efficient. An abundance of tools to produce better results in less time. That’s why Aztec C systems are the choice of more professional ROM developers.

So when you’re looking for the best results, insist on Aztec C ROM Cross Development Systems. Call today and find out more about our complete line of Cross Development Systems.

Supported targets include: the 68xxx family, the full 8086 family, the 8080/280 family and the 6502 family of microprocessors.

1-800-221-0440 (outside NJ)
1-201-542-2121 (NJ and Outside U.S.)
Telex: 495812MANX
Fax: 201/542-8386

Aztec by MANX SOFTWARE SYSTEMS
One Industrial Way Eatontown, New Jersey 07724
Reader Service Number 17
2 MB EMS MEMORY BOARDS

Super 80586 System
8/20 MHz features an 80386-20 CPU on a full size DTK motherboard with 2 8MB 32 bit memory slots. Accepts 80387-20. This machine runs an incredible Norton SI test of 24! For quality and reliability we've included a 40 MB Miniscrbe hard drive, 1.2MB & 360K Toshiba or Teac floppy drives, monographics with green or amber monitor, 101 keyboard, 1 MB of 80 ns DRAM, DTK Bios, slide case, 2 serial ports, 1 parallel port, power supply, clock, calendar. FREE assembly and testing. One year warranty.

Now!
$2395!

Other configurations:
tower case, color, etc... $Call
DTK 8 MB RAM Card ......... 99
(32 BIT./64)

Disk Drives (Continued)
AT 40 MB MiniScribe ....... 339
3650 (61ms) ............... 489
AT 40 MB MiniScribe ....... 3603 (25ns) ......... 489
AT 71MB MiniScribe ...... 649
AT (MFM) HD & FD Controller card DTK ............. 110
Controller ......... 101
AT RLL HD & FD Controller ............. 189

Monitors
EGA/CGA
(Autoswitch 31 dot) .......... 385
CGA/EGA/VGA
MultiSync (31 dot) .......... 489
CGA Color .................. 249
Amber 12" TTL ............. 89
Green 12" TTL ............. 89
VGA Analog
(Mitsubishi 28 dot) .......... 549
Color/Graphics/Par ........ 49
Mono/Graphics/Par ........ 49
CGA/EGA/VGA (640x480) .... 169
VGA Analog, STB Extra ..... 235

Keyboards
Chicony Click 101 .............. 49
Keytronic KB101 ................. 67
Focus 101 Tactile,
Switchable, Control Caps Lock,
Dust Cover ............... 89
(#1 find by MicroC Staff)
* All keyboards, XT/AT switchable*

Prices are subject to change without notice.
Shipping CHARGES will be added.

Building Your Own Clone V2.1
****FREE BOOKLET****
*60-day warranty/30-day money back
(subject to restrictions)
Tech Cals: (503) 388-1194
Hours: Monday-Friday 9:00-5:30
SERIOUS DEBUGGING at a REASONABLE PRICE

All the speed and power of a hardware-assisted debugger at a software price

**Soft-ICE**

Hardware-level break points
REAL-TIME break points on memory locations, memory ranges, execution, I/O ports, hardware and software interrupts. More powerful break points than ANY software-only debugger on the market. Soft-ICE gives you the power of an in-circuit emulator on your desk.

Break out of hung programs
With a keystroke - no external switch necessary. Even with interrupts disabled.

Breaks the 640K barrier
Soft-ICE uses ZERO bytes of memory in the first 1MB of address space. This is especially useful for those subtle bugs that change when the starting address of your code changes. With Soft-ICE your code executes at the same address whether the debugger is loaded or not.

Works with your favorite debugger
Soft-ICE can be used as a stand-alone debugger or it can add its powerful break points to the software debugger you already use. You can continue to use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated.

Solve tough systems problems too
Soft-ICE is ideal for debugging TSRs, interrupt handlers, self booting programs, DOS loadable device drivers, non-DOS operating systems, and debugging within DOS & BIOS. Soft-ICE is also great for firmware development because Soft-ICE’s break points work in ROM.

How Soft-ICE Works
Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to provide real-time hardware-level break points.

“Soft-ICE is a product any MS-DOS developer serious enough to own a 386 machine should have.”
Dr. Dobb’s Journal - May 1988

RUN CODEVIEW IN ONLY 8K!

**MagicCV**

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 microprocessor to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView using less than 8K of conventional memory on your 80386 PC.

Don’t let 640K be your limit!
If you are closing in on the 640K limit and would like the power of CodeView, MagicCV is for you.

Don’t let the debugger hide the bug!
Even if you’re not closing in on the 640K limit, running CodeView with MagicCV makes your debugging environment much closer to the end user’s program environment. You can use CodeView to locate subtle bugs that only occur when there is plenty of free memory, or those difficult bugs that only occur when your program is running with a couple of TSRs loaded.

How MagicCV works
MagicCV uses the 80386 to create a separate virtual machine for CodeView. MagicCV uses between 4K & 8K of conventional memory as a bridge between the DOS environment and CodeView.

MagicCV is easy to use
If you are a CodeView user, you already know how to use MagicCV too. Just type MCV instead of CV; everything else is automatic.

Save $86

**MagicCV with Soft-ICE**

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

Both require 80386 AT compatible or IBM PS/2 Model 80. MagicCV requires at least 384K of extended memory. CodeView is a trademark of Microsoft Corporation.
The Art of Disassembly
Disassembly has long remained one of the black arts: black because few do it well, black because of its hacker reputation. However, it’s a skill that will serve you well. Very, very well.

Handling Interrupts With Any C
Some Cs support interrupts elegantly, some support them barely. Here’s how to interrupt with any C in style.

Turning A PC Into An Embedded Control System
Bruce goes back to hardware and produces an EEPROM board. This is the first of a series on embedded controllers.

Bringing Up A Surplus 68000 Board
What would you do with a fancy, but undocumented, 68000-based processor board? Figure out how to make it run, of course.

Practical Fractals
What does a fractal have that’s practical? What does a fractal have in common with the Deschutes River? Larry brings us the answers to these burning questions.

Hacking Sprint: Creating Display Drivers
Now that you’ve hacked everything else you might as well write a display driver for a text editor. (You’ll, of course, use the editor’s interface language. Right? You knew about that, I know you did.)
This has probably been the most difficult editorial I've had to write. It's been difficult because I've had to make some hard choices, and it's been difficult because I can't explain everything that went into those choices.

Over the past couple of SOGs (Semi-Official Get-togethers), a number of you have approached me about future SOGs. Would there be future SOGs? Was I as exhausted as I appeared?

"Of course," I said, "there will be another SOG."

"Yes, I'm exhausted," I added, "but this is a wonderful time, the highlight of my year. Sure it's fun to write for 30,000, but it's ten times more exciting to speak with 400."

However, I'm not sure I'd survive another SOG. The Monday after SOG VII, Laura Logan left Micro C. It wasn't a bad leaving, she'd always wanted to live on the Oregon Coast. However, when she left, we lost our SOG organizer. She planned to continue the project, remotely, so we announced SOG VIII. But finally, as things became busy for her there (making waves, I presume), she bowed out.

For SOGs I and II, loss of a key person wouldn't have been a problem. One hundred people showed up at our house for a day of potluck food and potluck discussions. Sure, it was a long day — two days, really, but it was wonderful.

However, SOG has grown to 400 people, too many for any facility short of the local college. Laura made sure there were rest rooms, dorm rooms, motel rooms, and classrooms. She also set up rafting, volleyball nets, food services, apple barrels, and that myriad of wonderful, soggy things that had to be done.

We no longer have someone who has either the time or the experience to effectively handle another SOG, so I've been mulling over some options.

Charge for SOG. That way we could afford to hire a convention manager or service. But who pays? It's a family event with activities for spouses and kids and babysitting and all that. So far we've not charged for admission. (If we did charge, who'd patrol the corridors to make sure everyone had paid?) Plus, everyone from staff to speakers donated their time. That really made SOG unique.

End SOG. That's it. Kaput. No more. When I mention this option to folks who call in, I always get a stunned silence. (Thanks.) But some have offered to

Continued on page 72
YOU’LL LOVE THESE UTILITIES.

SAYWHAT?! The lightning-fast screen generator

TOPAZ. The breakthrough toolkit for Turbo Pascal 5.0

If you’d like to combine the raw power and speed of Turbo Pascal with the simplicity and elegance of dBASE, Topaz is just what you’re looking for. You see, Topaz (our brand new collection of utilities, sample screens, sample programs, and commands) is snappy and crisp, the way you like it. That means screens pop up instantly, whenever and wherever you want them.

With Saywhat, what you see is exactly what you get. And response time is snappy and crisp, the way you like it. That means screens pop up instantly, whenever and wherever you want them.

Whether you’re a novice programmer longing for simplicity, or a seasoned professional searching for higher productivity, you owe it to yourself to check out Saywhat. For starters, it will let you build your own elegant, moving-bar menus into any screen. (They work like magic in any application, with just one line of code!) You can also combine your screens into extremely powerful screen libraries. And Saywhat’s remarkable VIDPOP utility gives all languages running under PC/MS-DOS, a whole new set of flexible screen handling commands. Languages like dBASE, Pascal, BASIC, C, Modula-2, FORTRAN, and COBOL. Saywhat works with all the dBASE compilers, too!

With Saywhat we also include a bunch of terrific utilities, sample screens, sample programs, and outstanding technical support, all at no extra cost. (Comprehensive manual included. Not copy protected. No licensing fee, fully guaranteed.) $49.95

ORDER NOW. YOU RISK NOTHING. Thousands of satisfied users have already ordered from us. Why not call toll-free, right now and put Saywhat and Topaz to the test yourself? They’re fully guaranteed. You don’t risk a penny.

SPECIAL LIMITED-TIME OFFER! Buy Saywhat! and Topaz together for just $85 (plus $5 shipping & handling). That’s a savings of almost $15.

To order: Call toll-free

800-468-9273

In California: 800-231-7849

International: 415-467-6840

The Research Group

100 Valley Drive

Brisbane, CA 94005

☐ YES. I want to try:

Saywhat! your lightning-fast screen generator, so send ______ copies ($49.95 each, plus $5 shipping & handling) subject to your iron-clad money-back guarantee.

☐ YES. I want to take advantage of your special offer! Send me ______ copies of both Saywhat! and Topaz at $85 per pair (plus $5 shipping & handling). That’s a savings of almost $15.

NAME ____________________________

ADDRESS __________________________

CITY __________________________ STATE __________ ZIP __________

☐ Check enclosed ☐ Ship C.O.D. ☐ Credit card

# ______ Exp. date ______ Signature ______

THE RESEARCH GROUP

Reader Service Number 129

MICRO CORNUCOPIA, #46, Mar-Apr, 1989 5
DBASE™ On Line

"Pop-Up" DBASE Reference System
Powered by The Norton Guides™
Only $99

DBASE On Line is a "pop-up" quick reference system that provides you with instant access to all aspects of the DBASE language and programming environment.

No More Searching Through Books
Each DBASE On Line reference database is a complete and thorough reference library eliminating the need for tedious time-consuming searches through books and manuals. All reference databases are organized so you can find relevant information fast.

Comprehensive DBASE Reference
DBASE On Line gives you quick reference to important information such as: syntax, description, options, notes, library, example of use and complete cross reference to all related keywords.

Replaces Books & Manuals
Reference topics include: Commands, Functions, Operators, Cursor Navigation Keys, Error codes and messages, Config settings, Technical specifications, dbf file structure, descriptions for all utility programs along with tables for reserved words, Inkkey(), Readkey(), ASCII codes and line drawing characters. The Clipper and Quicksilver databases also provide complete reference on such topics as: Compiling, Linking, Debugging and full reference to the Clipper Extend System including C interface functions, Assmber macros and much much more.

60 Day Guarantee
Free Upgrades

DBASE On Line... Is Easy To Use
You will be using DBASE On Line productively in less than 5 minutes. Guaranteed!

Create Your Own Reference Databases
In addition to our databases, you can also create your own reference databases with The Norton Guides reference database Compiler and Linker which is included in every DBASE On Line package. Our well written manual will take you through each step of the creation process.

DBASE On Line is powered by The Norton Guides reference engine To power DBASE On Line. We have included the Norton Guides "reference engine". It features ease of use, small size and fast automatic lookup capability. It instantly "pops-up" information on your screen. right next to your work, right where you need it. In either full screen or movable half screen.

Instant "pop-up" Reference
DBASE On Line is a "pop-up" quick reference system that provides you with instant access to all aspects of the DBASE language and programming environment.

To Order, Call Toll Free: 1-800-622-6435

Continuous on page 80

Oops Indeed
I thoroughly enjoyed Micro C's Object Oriented Programming issue (#44). Indeed, I am one of the religious fanatics of the WWLFSOG (We Who Live For SOG) sect that Micro C caters to so well. But I seem to be suffering from an acute case of Groping for Acronyms Syndrome (GAS).

In issue #44's articles and advertisements, the letters OO, OOP, or OOPS are used for the same phrase — Object Oriented Programming. Zak Urolfner calls it OOP on page 18, Bruce Eckel calls it OO on page 32, the Digitalk ad on page 12 uses OOPS (the picture on page 13 does include the word "system"), and the Zortech ad on the inside back cover uses both OOP and oops (oops?)

First, I think OO is incomplete (maybe Bruce saw fireworks as he bathed in the Pleasure of Object Oriented Programming (POOP). Then there's OOPS. What's the "S" for? Software? System? Or is it just Silent? I suspect some advertisers think it's cute or has a nice ring. But, frankly, oops is something I say when I've made a mistake.

I can only hope these Fanatical Object Oriented Language Sellers (FOOLS) drop the "S" before we all Generate Adverse Gossip (GAG).

Larry Hollibaugh
2714 SE 19th Ave.
Portland, OR 97202

A Dose Of Reality
With all the Object hype being published, there is one article which I fear will not get the attention it deserves. In the May '88 issue of IEEE Software, the creator of C++ published an article titled, "What is Object Oriented Programming?"

Before some readers groan, "Oh God, not another one," I must say this article is different. I may do it injustice, but to brutally sum up the article:

In the beginning there was flat programming a la BASIC, but it had a,b,c wrong with it. Trying to fix a,b,c gave us Pascal and procedures. But now we had d,e,f problems. Trying to fix d,e,f yielded modular programming, but with problems g,h,i. Fixing g,h,i yielded abstract data type programming, but with problems j,k,l. We fixed j,k,l with object oriented programming, but now we have problems m,n,o, and we have no idea how to fix these!

The article goes on to say much more, but the fact that it showed objects as part of a progression rather than being brought down off the mountain carved on stone tablets just

Continued on page 80
From C To Shining C
Greenleaf Has

The Best Reputation
Over the last five years the name Greenleaf has become synonymous with quality software.

As a leader in the professional C programming library market, we are recognized by the industry in general, and our customers in particular, as the premier supplier of quality products that consistently lead the industry in performance.

Our customers rate our products, support and documentation EXCELLENT. It's a fact that almost 50% of our clients bought one or more of their Greenleaf products without ever considering the competition.

Greenleaf's dedication to the attainment of excellence is driven by our customers. They tell us what they need and want in order to perform their jobs as professionally as possible. It is this partnership in excellence that has made Greenleaf the most respected provider of C language libraries in the world.

If you don't own a Greenleaf product, talk to one of our users. We suspect that you will be a Greenleaf owner soon. If you already own Greenleaf products we thank you for your trust and support and we look forward to our continued partnership.

The Best Products

Greenleaf SuperFunctions
An advanced collection of functions for the experienced C programmer. Functions include expanded memory support, extended mouse functions, advanced time manipulations, unique date functions, DOS critical error handler along with Windows, menus and keyboard functions.

Greenleaf Business MathLib
The only product available that gives the programmer a complete set of math functions while providing the accuracy of BCD math to the C language. Business functions such as compound interest, internal rate of return, cash flow analysis, bond calculations, amortization, array processing, statistics, trigonometry and print flexibility make financial output a breeze.

Greenleaf DataWindows
This easy to use and powerful windowing C library features overlaid logical windows, transaction data entry and three styles of menus. Functionality such as context sensitive help, menus within data entry, keyboard idle, list boxes and ROM BIOS redirection. Pop-up, pull-down and Lotus style menus are also included.

Greenleaf MakeForm
Working in conjunction with DataWindows, MakeForm will reduce the time it takes to design and build a human interface. Forms stored in special files enable you to change them without recompiling your application. Cut and paste between multiple forms while using the fully supported editing features.

Greenleaf Comm Library
XMODEM, XMODEM CRC, RTS/CTS, and XON/XOFF protocols are all available on up to 17 ports with this interrupt driven, asynchronous communications library. Your programs can fly at up to 19,200 baud.

Greenleaf Functions
The original. Still as popular and powerful as it was 5 years ago. Over 300 functions to support your programming efforts. DOS and BIOS calls, string color text, time, date, and graphics functions are just the beginning.

Greenleaf Software Inc.
Bent Tree Tower Two - Suite 570
16479 Dallas Parkway
Dallas, TX 75248
1-800-523-9830
Texas and Alaska: (214)248-2561
Telex: 559-068
Reader Service Number 146
Disassembly is largely misunderstood. Many folks think it's some kind of reverse voodoo. Others think it's a nefarious sport better left to hackers. But the reasons to disassemble are many: including silencing an irritating bell, or recreating source after a hard disk crash.

I've divided this article into two parts. In the first I do a general disassembly of disassembly; in the second I use a disassembler called Sourcer.

What Is Disassembly?

Disassembly is the process of taking machine code and translating it back into assembly language. The program that does this is called a disassembler. There is no requirement that the original code be written in assembly, but no matter what language was originally used, disassembly will only produce the assembly translation of the machine code. An example of a disassembler is DEBUG's unassemble command.

The disassembler can translate numeric instructions into assembly mnemonics, but not every byte or word in an .EXE or .COM file is an instruction. There are tables for data, embedded messages, and so on, so it's often up to a human to make the disassembler's output useful.

A knowledgeable operator can greatly improve the quality of the final disassembled code by making multiple passes, each time gleaning new information. This new information can be fed back to the disassembler to improve the quality of the next run.

Why Disassemble?

Now the question, why on earth would anyone but a hacker want to disassemble code? There are many reasons.

First, to find out how the program works. To practice safe computing, you may want to check out any new pro-
grams to be sure they do not contain a virus.

Once you know how the program works, you can modify the program. I have a utility program called "browse" which allows me to view an ASCII file. It can move through the file one line at a time, or one page at a time.

However, to search for something, I must page a screen at a time until I find it. Tedious. So I disassembled the program and added a string search feature.

A third reason is to document a program. Sometimes programmers produce only object code as their finished product. With a disassembler, the company can at least regain the assembly code.

You can also reverse engineer a program. However, be very careful since distributing the results would probably be illegal.

How A Disassembler Works

There are two types of disassembler programs. The first assumes that every byte or word it reads is an instruction. This works until the disassembler reaches the first data area. The "unassemble" command in DEBUG is a good example of this kind of disassembler.

Figure 1 is a printout from DEBUG. Is this data, instructions, or both?

The main disadvantage to this method is that it requires a great deal of highly skilled operator intervention.

More sophisticated disassemblers also assume the binary code is all instructions, but they can later correct themselves and determine which addresses are data and which are instructions. But even these disassemblers will sometimes need help. Sourcer falls into this category and the intervention it requires is discussed in detail later.

A Different Type Of Disassembler

The second type of disassembler makes no initial assumptions about data or instructions. In this case the operator must tell the disassembler the entry point into the target program. The disassembler then starts translating instructions from that point until it encounters an unconditional jump or a subroutine return. It then stops translating and stores the beginning and ending addresses that define this portion of the program as instructions.

As the disassembler translates, it notes the destination addresses of all conditional jumps and subroutine calls that it encounters. It stores these addresses in a table and uses them as new entry points into the program since it can be sure that these addresses are for instructions.

The disassembler goes back and forth through the code until it has used up all the destination addresses. Thus, it generates a map of the instructions. It assumes everything else is data.

What this disassembler calls instructions, are usually instructions. However, the rest isn't always data. Instructions will be missed if they aren't called by the rest of the program.

There are two primary causes for uncalled code:

1. Multiple external entry points to the program.
2. Non-accessible code produced by compilers.

The non-accessible code is not important since it is never executed. However, multiple entry points should be caught by the operator.

The listing produced by the disassembler should be in a standard assembler format. If not, you will have more difficulty reassembling the program after modifications. A good disassembler will even add the assembler directives.

The disassembler has no knowledge of what the program does or how it does it. The listing will be labeled for the explicitly called data and instruction locations, but they will have some sequential format such as "loc_12" or "data_24" instead of intelligible names like
BIOS SOURCE

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER $99.95  BIOS Pre-Processor* $49.95  SOURCER w/BIOS Pre-Processor $139.95

USA Shipping & Handling $3; Outside USA $15; CA Residents add local sales tax 6, 6.5 or 7%. *requires SOURCER

All our products come with a 30 day money back satisfaction guarantee. Not copy protected. To order or receive additional information just call!

1-800-662-VCOM  (1-800-662-8266)

V COMMUNICATIONS
3031 Tisch Way, Suite 905, Dept. M3, San Jose, CA 95128  (408) 296-4224

PS/2, AT, XT, and PC are trademarks of IBM Corp.

Reader Service Number 62

Unbelievable!

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Complete support for 8086 through 80286, V20/V30, 8087, and 80287 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

On my list of programs that I simply won't do without!
—Robert Hummel, Senior Technical Editor, PC Magazine

BIOS SOURCE

CHANGE AND ADD FEATURES

CLARIFY INTERFACES

"get_rs232_input" or "screen_offset."

One of your major tasks is to work through the code, figure out what's happening, and change the sequential labels to meaningful ones. This process is complicated if the disassembler mistakes data for instructions or vice versa.

Most of the commercial disassemblers advertise a feature called commented code where the disassembler automatically puts comments in the listing. This is not really as useful as it seems. Remember the disassembler does not understand the target program. So its comments can only tell you a bit about what is happening, not why.

These three goals, deciphering program logic, adding comments and labels, and ensuring accuracy are synergistic. As you improve one, it's easier to improve the others.

For example, Sourcer will comment a LODSB instruction with "string [si] to al." This helps if you're an inexperienced 8086 programmer who would otherwise have to look it up, but what you really want to know is the purpose of the instruction. In this case, it might be "get byte from filename for parsing."

How The Disassembly Process Works

Now that we have seen what a disassembler does, and doesn't do, let's look at the overall disassembly process. The basic steps are:

(1) Examine the machine code using DEBUG.

(2) Feed the machine code to the disassembler.

(3) Examine the disassembler's output.

(4) Determine new settings for the disassembler.

(5) GOTO step (2) until the listing is satisfactory.
Steps (2) and (5) are self-explanatory and step (4) is disassembler specific, so we will concentrate on steps (1) and (3).

In step (1) you use DEBUG to dump the target code. This is to give you a general idea which addresses contain data. Look for ASCII strings and initialized data areas.

ASCII strings are readily recognizable from the dump listing. Record their data. Look for ASCII strings and initialized data areas.

If the strings end with a zero, they are in the ASCII form. This means the program was probably compiled with a high level language (for instance, C uses ASCII strings). If the strings end with a $, then the source was probably an assembly program.

You will be able to recognize the initialized areas if you get large areas of the same byte. In particular an area initialized to zero is easy to notice.

If you are not sure, use DEBUG to unassemble the code. If you get the same instruction 100 times, then it's probably data. Two in a row, well, they could be data or instructions. When in doubt, assume instructions.

Step (3), examining the disassembler's output, is the hardest part of disassembly. This is a job for an experienced assembly programmer. The key to successfully disassembling the target program is to examine the listing, instruction by instruction, to decipher the original program logic.

Deciphering The Listing

You must understand the target program so that you can add labels and comments, and ensure the disassembled version is an accurate recreation of the original. The mechanics of deciphering the listing vary, but I have found the following effective:

1. Start at the beginning and follow the program flow. When you come to a conditional jump, try to figure out why the jump exists. Record the conditions and take both branches. You may not initially know why the jump is there, but by comparing the instructions following both branches you can figure it out.

2. When you get to a subroutine call, note the contents of the registers, then go to the subroutine code. Try to determine the purpose of the subroutine. Record which registers it uses for input and which it uses to return values.

3. Don't try to get everything straight
the first time. Get as much as you feel you can, then rerun the disassembler with the new information. You will be surprised at how much easier the second examination will be since the new listing will have some descriptive labels.

Instead of trying to remember that "sub_23" reads a sector and "sub_15" writes a sector, you will have changed the labels to "read_sector" and "write_sector." If you can make the disassembler put comments in its listing, do so. You want as much information as possible on the listing.

4. As you examine the listing, be sure that the data is really data, and that the instructions are really instructions. Ideally you could check this before the detail exam, but unfortunately the exam is the only real way to know.

These three goals, deciphering program logic, adding comments and mnemonic labels, and ensuring accurate translation have a synergistic effect. As you improve one, it's easier to improve the others. Conversely you cannot complete one until the others are nearly complete. Working on these three simultaneously is the secret to disassembly.

Sourcer

Now that we have thoroughly examined general disassembly, let's look at Sourcer. In the rest of this article, I'll explain how Sourcer works, how to use it, and some things about it not mentioned in the operators manual. If you haven't read Sourcer's user manual, the last section may be hard to follow.

How Sourcer Works

Sourcer is a multipass disassembler where you set the number of passes from 2 to 5. On the first pass Sourcer assumes all the code in the target program is made up of instructions. As it goes through the code, it stores the symbol addresses for use in the next pass. It maintains three symbol tables; one each for data, locations, and subroutines. On the next pass it looks at the symbol tables and makes sure the item referenced is valid.

With each additional pass, Sourcer refines the symbol tables and its ability to distinguish instructions from data. The number of entries in each table may decrease as invalid entries are recognized and removed. On the last pass it writes the listing to disk.

Sourcer works through the program sequentially. It does not take any jumps or subroutine calls, and as it works through the code it maintains a set of simulation registers which it tries to keep consistent.

Continued from page 11

Figure 4 — Original Program Simple.asm

codeseg segment
assume cs:codeseg, ds:codeseg
org 100h
start: jmp begin
msgl: db 0Ah, 0Dh, 'Hello, please enter your name: $'
meg2a: db 0Ah, 0Dh, 'Well, $'
meg2b: db ', how do you like computers?', 0Ah, 0Dh, '$'
nname_buf: db 21
 db 0 :buffer for name input
 db 20 dup (' ') begin: lea dx,msgl :ask for users name
 mov ah,9
 int 21h
 mov dx, offset name_buf :get name via string inp
 mov ah,0AH
 int 21h
 mov dx, offset meg2a :write 1st part of response
 mov ah,9
 int 21h
 mov cl, byte ptr name_buf + 1 :get char count to cl
 xor ch,ch :count in cx
 mov si, offset name_buf + 2 :get buffer addr
 mov ah,2 :set up for char disp
 jcxz done :check for null string
 disp_char: lodsb :get char from buffer

/* */
Figure 5 — Example of Range Overlap

Range Section fragment

```plaintext
sega:0100 0102 sega sega code
sega:0100 0100 0000 0000 data
```

Range Section fragment

```plaintext
start: jmp short loc_1
```

Listing fragment

```plaintext
81BF:0100 start: jmp short loc_1
81BF:0102 EB 61 db 0Ah
81BF:0102 EB 61 db 0Ah
81BF:0103 0A db 0Ah
```

updated. These registers are used in producing comments or finding the address of jump tables.

Since Sourcer does not jump or loop, the simulation register will be incorrect if the program changes a register and then jumps to a previous location.

You control Sourcer via a command menu or a definition file.

**Definition File**

The definition file has three sections.

The first is for control information and duplicates all of the commands available from the menu.

Section two is for range definition and it specifies how Sourcer processes a range of bytes.

The third section is for reference definition. It specifies how a particular address is to be treated. Figure 3 has sample definition files.

The range definition tells Sourcer which portions of the target program are data, which are instructions, which are unknown, and which are ROM. Sourcer will read these directions, and it will skip the data areas during the first pass. This prevents bogus labels from being entered into the symbol tables.

Reference definition allows finer control. You specify individual addresses and direct how Sourcer is to treat those addresses. For data you can specify a byte, word, dword, ASCII, or even a structure. If you tell Sourcer that the data item is an offset, it will add the offset directive and target label.

Sometimes Sourcer will override your range definitions. This can happen if you miss a data area, and Sourcer erroneously translates the data as a jump to the area you defined as data. Sourcer will then believe the bogus jump and label the area (you previously defined as data) as instructions.

**Example Disassembly Using Sourcer**

Let's look at an example disassembly process using Sourcer. Figure 2 is a DEBUG dump of the program "simple.com." This is the same code as in Figure 1. Note the possible data areas. Figure 3 is the define file derived from looking at the DEBUG dump and Sourcer's output based on that definition file. Compare the disassembled listing to the original program in Figure 4.

**Sourcer Secrets And Tips**

(1) Some of the definition file control commands are sequence sensitive. For instance, the "Header" command must be after the "Input filename" command or you'll get the default header. The "keep segments" command must be before the "Input filename" command.

(2) Sourcer will accept almost any
string as a label, but it's a good bet your assembler won't. If you plan on reassembling the code, save yourself some pain later and use only labels acceptable to your assembler.

(3) If the target program contains an INT 20H, Sourcer does not recognize this as the end of the procedure and does not put the required "endp" directive into the listing.

(4) Sourcer has a number of built-in labels which it uses when it feels the need. They include: "copywrt" for a copyright statement, "start" for the program's entry point, "int_XXh_ext" for external interrupt entries, and "strategy" and "interrupt" for device drivers. You have no control over these labels.

(5) If you use the "repeat" option to define a data area and you miscount the number to be repeated, Sourcer will add bogus bytes as needed. This can rear its ugly head if you have defined ASCII text and it contains a copyright. The built-in copywrt label will be applied, and your repeat count will be wrong.

(6) When you define segments via the range definition section, you must explicitly specify the last byte's address. There is no end of code operator. The best way to find the address of the last byte is to load the program with Sourcer then note the last address as shown on the main menu. Since Sourcer will process all the addresses you specify, it will stop where you told it regardless of the length of the program.

(7) The addresses in the range definition section should be contiguous. If there is a gap, Sourcer will skip over the gapped code and generate an org statement. This ensures the subsequent addresses are in their proper place, but the gapped code is removed.

(8) If two range definitions overlap, Sourcer will process the overlapped code twice. If the first range is a data definition, and the second is code definition, Sourcer will output both the data and code. Figure 5 illustrates this.

(9) The "keep segments" command is not explained well. It tells Sourcer which of two sets of range definition to use. The first is internal to Sourcer and is created as Sourcer processes the binary code. The second is the one you write in the definition file and is optional. The "keep segments" command tells Sourcer to use either: its internal definition (FILE), the one from the definition file (DEF), or to use both (BOTH).

If you haven't written a range definition, use the default FILE option. If you have written a range definition that includes the target program's address, use the DEF option. If your range definition contains only addresses outside the program's areas (such as the BIOS data area), use the BOTH option. Sourcer will process the code twice if your range definition covers the program's address space and you specify BOTH.

(10) Labels can be added to the disassembly listing at will. However you should restrict their use so that they are present only when they are the target of an instruction. For instance, if you add extraneous labels to instructions, it will be unclear which labels are decorations and which are the destinations of jumps and calls. Unfortunately Sourcer requires you to specify a label in order to add a comment. I recommend you use a sequential label notation so that each label is unique and is distinguished from functional labels.

(11) If Sourcer's analyzer makes a mistake and translates data as instructions, these bogus instructions can cause trouble farther down the line. Any addresses the bogus instructions reference as data will be marked as data, even if they are really instructions. Bogus jumps or calls will mark the destinations as instructions and these second generation bogus instructions can cause additional trouble. If you find that Sourcer made a mistake, don't just fix the part you found, backtrack until you find the root cause. The cause will always be bogus instructions.

(12) The more passes you specify, the better the analyzer can separate data from instructions. Five passes will produce the best resolution, and you can use the cross reference option. By using cross references, you can backtrack to the source error.

Sometimes a label will not have a cross reference. This indicates the label was created by a bogus instruction that the analyzer has converted back to data. You can find the cause by reducing the number of passes and then using the range definition to fix the problem.

Sourcer
V Communications
3031 Tisch Way Suite 905
San Jose, CA 95128
(408) 296-4224
### ERAC CO.
8280 Clairemont Mesa Blvd., Suite 117
San Diego, California 92111
(619) 569-1884

---

### KAYPRO EQUIPMENT BARGAINS

<table>
<thead>
<tr>
<th>Item</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>9&quot; Green Monitor</td>
<td>$50</td>
</tr>
<tr>
<td>9&quot; Amber CRT.</td>
<td>$45</td>
</tr>
<tr>
<td>PRO-8 Mod. to your board</td>
<td>$149</td>
</tr>
<tr>
<td>Host Interface Board</td>
<td>$15</td>
</tr>
<tr>
<td>Replacement Power Supply</td>
<td>$50</td>
</tr>
</tbody>
</table>

### ELGAR UNINTERRUPTIBLE POWER SUPPLIES

- **560 Watt MODEL IP5560 $650**
  - Sinewave, 560W complete w/batteries.
- **400 Watt MODEL SPR401 $149**
  - Supplies may have minor cosmetic damage, but are electrically sound. Squarewave output. Run on internal or external 24VDC battery when line goes down. Typical transfer time = 12MS. Battery supplied, not guaranteed.

**NEW 24V INTERNAL BATTERY $75**

### CPU & SUPPORT CHIPS

- **MC68000-8 CPU** $8.00
- **280 CPU** $7.50
- **280A CPU** $1.50
- **280 CTC** $1.50
- **280A PIO** $2.00
- **280A SIO** $5.00
- **8088** $6.50
- **8089-3** $6.50
- **D8284A** $2.50
- **SIP DRAM 256-12** $7.00
- **4164-10** $2.50
- **4164-12** $2.10
- **1793** $6.00
- **1797** $7.00
- **6845** $5.00
- **VC3524 Switching Regulators** $5.00
- **1458 Dual Op-AMP** $7.00
- **LM2877P 4W Stereo Amp Dual** $2.50
- **MB1464-15** $2.75
- **2716** $3.00
- **2732** $3.25
- **2764** $3.50
- **27C128-1** $9.00
- **74HC00** $0.38
- **74LS125** $0.30
- **74LS373** $0.50
- **74LS174** $0.30

### SWITCHERS

- **5V/9.5A, 12V/8A, -12V/8A** $39.00
- **5V/3A, 12V/2A, -12V/4A** $19.50
- **5V/6A, 12V/2A, -12V/1A** $29.00
- **5V/6A, 24V/1.1A, 12V/6A, -12V/6A** $29.00
- **5V/10A** $19.00
- **5V/20A** $24.00
- **5V/30A** $39.00
- **5V/100A** $100.00
- **5V/120A** $110.00

### TEST EQUIPMENT

- **OSCILLOSCOPES**
  - **TEK 7403N/7A18N/7B50A 60 MHz** $650
  - **TEK 465 Dual Trace 100 MHz** $1000

### ANALYZERS

- **TEK 491 10MHz - 40 Ghz** $4500
- **Nicolet 500A 1 Hz-0-100 KHz** $1800
- **Biomation 805 Waveform Rcrdr** $259
- **Biomation 8100 2 Channel Waveform Recorder** $795
- **HP1800A Logic Analyzer** $800
- **HP1800A/1607A Logic Analyzer** $1000

### FREQUENCY COUNTER

- **Optronics 550 MHz** $100

### BASE BOOK OF BUSINESS APPLICATIONS by Michael J. Clifford

Reg. $19.95

**NOW ONLY $3.00**

---

### TERMS

- **MINIMUM ORDER** $15.00
- **VISA, MasterCard, Certified Checks, Money Order, NO COD. Visa and MasterCard add 3%. Personal checks must clear BEFORE we ship. Include shipping charges. California residents add 8.5% Sales Tax. For more information please call.
Handling Interrupts With Any C
A Detailed Look At Supporting Interrupts

Micro C is famous for cramming lots of useful information into very little space. This article may be too long to qualify as “little space” but it’s so crammed with information on implementing interrupt service routines (ISRs) that you’ve got to read on. (Find a spot where you won’t be interrupted.) This is a great one, folks.

One idea which Borland added to Turbo C really bowled me over when I first saw it. It’s their “interrupt” type modifier! Hooray for Borland, but not everyone uses Turbo when working in C.

I found another simple but impressive idea hiding in a listing for a low level driver. (I don’t remember where I saw the listing.) The listing contained assignments to CPU registers! What incredibly nice code to read — no structure references full of obscure object names. Very neat!

I decided to combine these two ideas into support headers and library functions. The CPU registers were easy. A long list of #defines did most of the job. Adding support for high level interrupt handlers wasn’t too tricky either, and it’s turning out to be more useful than I expected.

In this article, I’ll describe two main modules, each module a group of functions. The first group supports existing system interrupt and far call functions. The second group contains support routines to allow standard C functions to service interrupt requests or be called by external programs.

Everything was written to be compiled with Manx Aztec C 86, version 4.10a. Other compilers might run into minor problems handling the assembly code, but it shouldn’t be anything serious.

CALLs & INTs

The functions in this group are super simple. They do software interrupts and far calls. The work is done by two functions in the CALL.C module, SEG86 and CALL_80860. (See Figure 1.) They’re written in a mix of assembly level spaghetti and C. The third function, INTN_80860, and fourth, regs_dump, I’ll describe shortly.

#include <call.h>

unsigned int far call_8086( ri, ro )
  _regs far *ri, *ro;

unsigned int far intn_8086( ri, ro )
  _regs far *ri, *ro;

unsigned int far seg86( r )
  _regs far *r;

void _regs_dump( r )
  _regs far *r;

ri and ro are pointers to _regs structures; *ri contains the register values passed and *ro contains the values returned.

CALL_80860 does a far call to the far function at location ri->CS:IP. intn_80860 assumes that the desired interrupt number is in ri->IP, and does a software interrupt to that handler. Both functions return the result in the PSW register.

seg86 copies the DS, ES, CS, SS and PSW register values into the corresponding objects in the _regs struct at *r. _regs_dump0 displays the contents of the _regs structure *r on STDOUT.

The _regs structure is typed in the CALL.H header (see Figure 2). It’s a complicated structure full of unions of structures of pointers and unsigned chars and ints, representing the CPU registers. The main difference between this structure and the ones included with most 8086 compilers is the organization of the registers. You can directly access some of the more commonly used register pairs: ES:BX, DS:DX and DX:AX. I also used bit fields to fully define the PSW (Processor Status Word).

You’ll see the list of #defines following the structure definition in CALL.H. The list names each CPU register, its lower and upper halves, plus some register pairs and the FSW flags. Each register/flag name expands into the notation required to get at it in a _regs structure.

An almost identical list follows, only this time the names expand into a reference to an object within the _regs struct _REGS. Nothing called _REGS is declared in the header, so you’ll have to declare a structure or macro by that name before using any of the macros.

The point of the second list is to eliminate completely all structure/union notations so you can work directly with the desired registers, register pairs and CPU status flags.

The last group of #defines expands into function calls. call86(pc) does a far call to the function at void (far *pc), int86(i) does a software interrupt to interrupt i, and pcdos(f) loads the DOS function number f into AH, then calls DOS. Each macro expects to use the _REGS structure for both the “input” and “output” registers.

ISRhs

When an ordinary C function gets called, it expects the proper values in the segment registers, expects parameters on the stack, and expects the stack to be big enough for automatic storage.

An ISR Handler can’t expect anything. To get a C function to handle an interrupt, you’ll need to add some support code between the interrupt and the function call (and before the return-from-interrupt).

I figured the easiest solution would be to write a “straight” version of the support code, then have another routine copy it into a data, structure for each ISR handler. The file ISRHT.C contains this “ ISR Handler template.” (See Figure 3.) The ISRH data structure contains space for the code, and variables are positioned so you can modify the code.

This way, the function isr_install() can install a C function as an interrupt handler, then isr_restore() can put the previous vector back into the interrupt table when the program’s done. It’s that simple!
Figure 1 — CALL.C far calls and software interrupts from C

```c
#include <call.h>
/*
   SEG86
   This function copies the 8086 segment register contents
   into the __regs register structure pointed at by ri.
*/
#pragma vppindex seg86
unsigned int far seg86( ri )
__regs far *ri;
{ -
  asm
  push es
  push es
  les bx, _RI
  pop ax
  mov _ES, ax
  mov _OS, ds
  mov _55, ss
  mov _CS, cs
  pushf
  pop ax
  mov _PSW, ax
  pop es
#pragma endasm
  }
/*
   CALL_8086()
   This function calls the code at PC in *ri. It doesn't
   check for messed up stacks, but is reentrant.
*/
#pragma vppindex call_8086
unsigned int far call_8086( ri, ro )
__regs far *ri, *ro;
{ -
  asm
  push si ;save C regs
  push di
  push ds
  push es
  les bx, _RI ;get pointer to input regs
  push bp ;we're going to need this later...
  push cs ;here's where the routine that we'll call
  push ax, offset retaddr
  push ax ;will return
  push _CS ;put address to call on the stack
  push _IP
  do_it: push _ES ;can't touch these now...
  push _BX ; so put them on the stack
  mov ax, _AX ;load up the cpu regs
  mov cx, _CX
  mov dx, _DX
  mov si, _SI
  mov di, _DI
  mov bp, _BP
  mov ds, _DS
  pop bx
  }
```

The file ISR.H contains the typedef for the ISR.H data structure, the prototypes for the functions in ISR.C, and some support macros (see Figures 4 and 5). ISR.C contains all the support routines. The listings are reasonably straightforward, so I'll try to be brief here.

The ISR functions are:

```c
#include <isr.h>

void *isr_get_vector( inum )
int inum

void isr_set_vector( inum, farfuncp )
void far (*farfuncp)()
int inum;

void isr_install( isr, inum, handler )
isrh *isr;
int inum;
void far (*handler)();

void isr_restore( isr )
isrh *isr;

isr_get_vector() returns the vector for interrupt inum from the 8086 interrupt vector table. isr_set_vector() puts the vector farfuncp into the table.

isr_install() copies the "handler template" code into the isrh structure *isr. It then modifies the code in that structure. The far function *handler is installed as the function that will handle the interrupt. The vector for interrupt inum is saved in isr->prev_hndlr.

If that vector is NULL, *isr will do an IRET when *handler returns; otherwise, *isr jumps to *isr->prev_hndlr on return from *handler. The last thing isr_install() does is point the CPU's interrupt vector to interrupt inum at *isr; at which point *handler will be called if an interrupt occurs.

isr_restore() restores isr->prev_hndlr to the interrupt vector table.

The template code does the following:
- Stacks the CPU registers.
- Loads the correct data segment value into the DS and ES registers. (You'll have to check your start-up routine to find out what
values you need in DS and ES.)

- Stacks a far pointer to the ISRH data structure (i.e., this routine.)
- Does a CALL FAR to your C handler function.
- On return, pops the far pointer to the ISRH structure off the stack, discarding it.
- Restores the CPU registers.
- If your C handler used the isr_pass() macro before returning, then an IRET instruction is executed here. If it used the isr_pass() macro, or neither of the two macros, the default procedure at this point is to jump to the previously installed handler; or in the absence of one, to execute an IRET.

The isr_retn() macro can also be used by functions that are called by external programs; it causes the handler to execute a far return. This can be handy, for example, when writing handlers for mouse driver interrupts. The mouse driver will intercept the interrupt, then do a far call to the user's handler.

### Using The ISR Functions

Since the CPU registers are pushed on the stack used by the C handler function, you can access anything that's on that stack as a parameter. A handler would therefore be declared like this:

```c
void far handler( *isr, bx, es, cx,
    ax, dx, ds, di, si, bp, ip, cs, psw )

isrh far *isr;
```

Or, if your compiler supports passing structures by value (most do)

```c
void far handler( *isr, _REGS )

isrh far *isr;
isr_regs _REGS;
```

Again, CS:IP and PSW were pushed by the CPU prior to calling *isrh, which stacks isr and BX through BP, then calls handler(). On return from handler, *isr pops BP through BX and isr. CS:IP and PSW get popped off when an IRET is executed.

If, for example, you install an INT 0x23 Ctrl-Break exit handler, it can either clear the caller's Carry flag to indicate that you want your program to continue executing, or set the carry flag to indicate that you'd like to crash. An INT 0x23 handler to ignore Ctrl-Break interrupts would be set up like the following:

```c
main()
{
    void far int_23();

    isrh isr_23;

    isr_install(&isr_23,Ox23,int_23);

    /* your program */

    isr_restore( &isr_23);
}

void far int_23( isr, _REGS )
isrh far *isr;
isr_regs _REGS;

    _CARRY = 0;
isr_iret(isr);
```
would have a parameter list like this: (isr, r1, r2). r1 was pushed by DOS before it issued the INT 0x24. r2 was pushed by DOS when it received control from your program. To force DOS to retry the operation, you'd have to set r1.AX to 1, then do an IRET (return after an isr_iret(isr)).

A couple of side notes: See the IBM Disk Operating System Technical Reference, chapter 5, if you need more info on the Critical Error handler (and others).

The isr pointer is passed by the template code to allow the interrupt handler to identify the source of the interrupt. I did that to allow a general handler to intercept and identify illegal interrupts. If you find you need the stack space, or aren’t suffering from spurious interrupts, you can get rid of it by deleting the associated code in the template, in isr_install() and in the ISRH structure definition in the ISRH header. (Spurious interrupts aren’t a problem most of the time.)

The Listings

CPU.H contains CPU related constants and the definition of the _flags structure. REGSDUMP.C contains the entire _regs_dump() routine, which is kept separate to ensure that it’s only linked in if it’s used (see Figures 6 and 7).

At the end of ISRC is a test routine that gets compiled if MAIN is #defined. As you can see, it installs handlers for the keyboard, timer and CTRL-BREAK interrupts. Essentially, it displays the last scan code received from the keyboard, and the elapsed time since the program started. It’s not very easy to break out of since it traps all attempts from the keyboard, and tells DOS to ignore anything that might come down a COM line if the console is redirected.

Two other files, CALLER.C and CALLER.H, contain improved versions of code originally written as part of a big project two years ago (see Figures 8 and 9). They served as the prototypes for CALL.C and CALL.H. They’re based on a routine, caller0, that’s faster, smaller and uses less stack space than the new functions. It also saves and restores SS:SP to recover from routines that don’t behave properly.

I didn’t bother to discuss it here because it’s not ROMable or reentrant. For some applications, though, caller0 would do just fine, so those two files are included with the others in the file ISRARC. Look for it on the Micro C BBS or the Issue #46 disk.

ISRH Notes

The main thing to bear in mind when
DRIVE TESTS—Complete diagnostics for Hard and Floppy drives, including controller cards. Tests read, write, and format capability as well as seek timings, hysteresis and rotation timings.

I/O PORTS—For both parallel and serial ports, confirms internal and external loopback capabilities at all baud rates and configurations.

MEMORY—Performs over eight different tests to check standard, extended, and expanded memory.

KEYBOARD—Verifies that all keys send correct key codes, including shift, CNTRL, and ALT modes.

CPU & NUMERIC COPROCESSOR—Verifies that all single and multiple instructions perform correctly and accurately, as well as testing all internal registers.

VIDEO DISPLAY—Checks video controller cards. Confirms attributes, graphics, colors (if applicable), and CRT alignment patterns.

REAL TIME CLOCK—Verifies correct timing, all internal registers, and battery backed-up RAM.

...and many more features to insure the integrity of your computer.

Figure 2 — CALL.H

/* CALL.h - header to allow low level routines to look like they work directly with CPU registers and do call/inta "naturally!" Works with the functions in CALL.C/ISR.*
*/

#ifndef OP_RETF
#include <cpu.h>
#endif

/* These typedefs describe the types of structures that are found in the final register structure at the end. These aren't intended to be used by humans... */

typedef union __reg {  
    struct {  
        unsigned char l, h;
    } b;
    unsigned int w;
} __reg;

typedef union __regdp {  
    struct {  
        unsigned char l, h;
    } b;
    unsigned int w;
} __regdp;

typedef union __regpsw {  
    struct {  
        unsigned int w;
    } __regpsw;
} __regpsw;

typedef union __reg3r {  
    struct {  
        reg ax, dx;
    } r1;
    struct {  
        reg ax;
    } r2;
    struct {  
        reg dx, ax;
    } r3;
} __reg3r;

typedef struct REGS {  
    regpsw ebx;
} __regpsw;

/* This is the main register file structure. The interrupt handlers all push and pop registers in this order, and the functions in CALL.C work with this "register file" organization. Note that the stack pointer isn't actually used. That object was included in the structure for "completeness" and/or "future expansion," etc. */

typedef struct __REGS {  
    __regdp esbx;
    __reg ax;
    __reg dxax;
} __REG3R;

PC/XT System Diagnostic Software $29
PC/XT Disk Diagnostics (w/test diskettes) $29
PC/XT I/O Loopback Test Plugs $19
COMPLETE PC/XT DIAGNOSTICS SET (save $28) $49

PC/AT System Diagnostic Software $29
PC/AT Disk Diagnostics (w/test diskettes) $29
PC/AT I/O Loopback Test Plugs $19
COMPLETE PC/AT DIAGNOSTICS SET (save $28) $49

BOTH PC/XT and PC/AT SETS (save $75) $79

Capital Software presents the definitive disk-based diagnostics package for the IBM PC AT and XT. A technical tool detailed enough for the repair technician. A friendly interface that places problem-solving skills in the hands of the end user. An uncompromising solution.

SEND CHECK OR MONEY ORDER TO CAPITAI SOFTWARE 951-2 OLD COUNTY ROAD SUITE 224 BELMONT, CALIFORNIA 94002 FOR INFORMATION CALL: (415) 592-9076
writing an ISRH is that your function will be using whatever's left of the stack frame that the interrupted code was using (i.e., don't expect to have any stack space available). Better still, don't use automatic storage at all — if possible, declare your local variables as static.

Make sure that your handler isn't being interrupted, then called reentrantly. If that's going to happen, be careful. You might also have to implement some form of mutual exclusion lock, or modify the template code to use an array of stack frames.

Often, you can avoid these hassles by keeping a handler short and sweet. Let it collect whatever information the main program might need, execute a procedure, or whatever; but keep it simple, and get out fast!

Remember that the compiler makes assumptions about the structure of the memory spaces that it works with. In particular, it may assume that your stack frame is in its data segment (especially in the small data models), so it's wise to be careful about pointers while servicing an interrupt. If you've got a bug somewhere and don't see any problem with your source, check the assembly level output from the compiler. Make sure that the proper segment values are used in pointer operations.

Not all ISRHs are going to be tricky, but many will. In most cases, it won't be possible to run them through a debugger, and library functions like printf() won't work from within them. On the other hand, they're useful enough to be worth the extra effort!

---

Continued from page 20

```c
unsigned int di;
#define REGDP bpdi;
#define PCCREG bp;
#define REGPWR pw;
#define REGDP stk;
#define _regs;
/* These macros are provided to make it easier to access objects in the above structure. Use either p-_AX or s._AX as appropriate to keep the code from looking too silly... */
#define AX dadax.rl.ax.x
#define AH dadax.rl.ax.b.h
#define AL dadax.rl.ax.b.l
#define BX esbx.i.r.x
#define BH esbx.i.r.b.h
#define BL esbx.i.r.b.l
#define CX c.x
#define CH c.b.h
#define CL c.b.l
#define DX dadax.rl.dx.x
#define DH dadax.rl.dx.b.h
#define DL dadax.rl.dx.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define CS c.x
#define CH c.b.h
#define CL c.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
#define DI di
#define SI bpsi.i.r.x
#define SP bpsi.i.s
#define ES esbx.i.r.b.h
#define SS esbx.i.r.b.l
#define DS dadax.rl.ds
#define ES esbx.i.r.b.h
#define BP esbx.i.r.b.l
```
The low-level format of your hard disk drive is probably the last thing you want to think about, but like the foundation of your home, you depend upon it every day without ever giving it a second thought... until something goes wrong.

Every byte of data stored in your hard disk rests upon the drive's low-level format foundation. When that foundation weakens, DOS begins reporting errors:

- **BOOT FAILURE**
- **SECTOR NOT FOUND**
- **BAD SECTOR ERROR**
- **GENERAL FAILURE READING DRIVE**
- **ABORT, RETRY, IGNORE**

That's how your vital data becomes hard to recover. That's how your computer hard disk drives the least reliable components in our computers.

Today you have two choices: Sit around worrying about the safety of your data, backing up the drive continually to minimize the extent of the loss when it occurs...

**Our users' experiences has shown:**

If you use SpinRite® every two or three months, you'll never have ANY PROBLEMS with your hard disk.

Or cure the problem at its source by preventing your drive's low-level foundation from ever weakening and crumbling.

SpinRite completely eliminates the problem of gradual low-level format deterioration by quickly low-level reformating any DOS hard disk while leaving all its data in place...

**But SpinRite goes FAR BEYOND JUST THAT!**

SpinRite's Main Features:

SpinRite is an all-in-one, total low-level, format maintenance, repair optimization utility.

- Non-destructively low-level reformats any DOS hard disk drive in minutes. Backup & restore are not required.
- Fully automated surface defect management utilizing the industry's most extensive worst-case data pattern analysis.
- "On-the-fly" instant sector interleave optimization establishes the maximum possible drive data transfer rate.
- Recovery and repair of correctable and completely uncorrectable (unreadable) data.
- Identification, diagnosis and repair of every form of data and format damage.
- MFM, RLL, APLL and EPLL compatible.

- In a matter of minutes it gives any DOS drive a completely new, clean, stable and solid low-level format WITHOUT requiring a tedious backup & restore operation.
- It detects and eliminates all data-threatening hard disk errors (which DOS can't see) long before they become data-damaging.
- It instantly optimizes and resets the drive's sector interleave, guaranteeing maximum possible data transfer rates.
- It locates and isolates all data-threatening surface defects. (Two to three times more than ANY other surface testing software)

SpinRite is offered with a 30-day money-back satisfaction guarantee. It is extremely easy to use with a simple user-interface, on-line help, on-line index and a short 40-page owner's guide. SpinRite is immediately available from:

Gibson Research Corporation
22891 LaCadena, Dept-MC
Laguna Hills, CA 92653
(714) 830-2200

Credit card orders, personal checks and COD orders welcome. Send: $59 plus $2 shipping and handling. California residents please include 6% state sales tax.

---

**unsigned int far seg86( _regs far *ri );**

**unsigned int far call_8086( _regs far *ri, _regs far *ro );**

**void _regs_dump( _regs far *rp);**

/*
 These macros do what they seem to do...

NOTE: the cast to (void * (x)) for _PC in the call166() macro doesn't work unless x appears before 1,h in the typedef for PCREG (Manx C 86, V. 4.10a.) */

```c
#define call166(pc) ( _PC = (pc), call_8086( _REGSP, _REGSP ))
#define int86(i) ( _IP = (i), int86_8086( _REGSP, _REGSP ) )
#define pados(f) ( _AH = ( f), int86( 0x21 ) )
```

This code is a typical example of how to call a 8086 function from inside a C program.

---

Note: You'll find figures 3 - 9 on pages 74-79.
Pioneer Laser Disc Player
These attractive system chassis were manufactured by Televideo for the Pioneer Laser Disc Computer System. They are brand new and include the following features:
- 12-mmTm He-Ne Laser Tube
- Laser Power Supply
- 2 Front Surface Mirrors
- Two 1/2" Voice Coil Actuated Oscillating Mirrors
- One Beam Splitter
- Two Optical Lenses
- One Optical Detector
- Mini Gear Reduction Motor
- All Controlling Electronics
- Assorted Switches, Fan, Solenoid

These chassis are not available from HSC. They cannot be manually controlled. They are as-is basis.

$99.00

IBM / APPLE SOFTWARE BLOWOUT!

Mitsumi UVEX-AW51P
UHF / VHF VARACTOR TUNER
- 3 1/2" x 2" x 1/2"
- Pinout on Case
- Two for $17.95

APPLE TITLES
- FlashCalc
- VisiCalc
- VisiLink
- VisiDex
- VisiTutor for VisiCalc
- VisiCalc (Advanced)
- VisiFile

IBM TITLES
- Visi On Graph
- Visi On Calc
- Visi On Word
- VisiWord
- VisiSpread
- VisiTutor for VisiWord
- StretchCalc

Price from $6.95 to $12.95
* Mouse and Basic required for some programs

ENGINEERING / TEST / DEVELOPMENT
AT/XT 3-SLOT MOTHERBOARD EXTENDER
- Fused Extender Card
- One 16-Bit Slot
- Two 8 Slot Switch
- Test Points for All Bus Points
- Power Connector
- Cables Incldued
- Not an expansion chassis

$89.95

HALF HEIGHT EXTERNAL DRIVE ENCLOSURE
- Attractive Low Profile Case
- 15" x 15" x 3" D.D.
- Fits nicely directly under PC
- Standard IBM Colors
- Bezel fits one 5 1/4" and one 3 1/2" Drive Only

$29.95

DELTEC AC LINE CONDITIONER
400 WATT
- Ideal for you PC or Any Equipment
- Eliminates AC Line Noise and Regulates to a Constant 120VAC
- Reliable Ferroresonant Transformer
- Attractive Case with Power Cord, Outlets, and Switch/Circuit Breaker

$149.00

WE SHIP C.O.D.!
HSC of Sacramento
5549 Hamlock St.
Sacramento, CA 94928
(916) 338-2545
HSC of Santa Rosa
6819 Redwood Drive
Costa, CA 95412
(707) 792-2277

Call California Residents 408-732-1573

Reader Service Number 11

TERMS: Minimum order $10. California Residents and 7% sales tax. Prepaid orders are freight C.O.D. or call for charges. Shipping will be added to credit card and C.O.D. orders. $2 Handling charge on orders less than $25. Prepaid order over $100 use money order or certified check. Please no test cash. Some items limited to stock on hand. Prices subject to change.
Turning A PC Into An Embedded Control System:
Phase 1 — Interfacing EEPROMs To The PC Bus

Okay, guys, Bruce is hacking hardware again. He's taken on a project that'll keep him embedded in our hearts and minds for some time. It's an unforgettable design project using EEPROM.

In the coming months, I'll be working (piece by piece) on an ambitious project—turning a PC motherboard into an embedded control system.

In this issue, I'll start by showing you how to use nonvolatile memory (or EEPROM: Electrically-Erasable Programmable Read Only Memory) in your PC.

Embedded Controllers

Embedded controllers are standalone systems generally dedicated to a single task. They usually don't include monitors or disk drives, but sometimes do have a very simple keyboard or display.

You hear a lot about the microcomputers in telephones, toasters, toilets, and televisions. These embedded controllers on a single chip have been optimized for cost. Single-chip microcontrollers are ideal for situations where you make thousands of products, because the design costs can be spread out. These costs are (of course) significant.

While I was at Fluke (in Seattle), I programmed a 4-bit microcontroller designed by NEC to drive a vacuum-fluorescent display directly (like the one on your VCR). This involved learning the hardware characteristics of the chip and a very arcane assembly language. We had to develop and debug the assembler while writing the code (with little in the way of examples or English documentation).

All this work was expensive. However, since it was going into a meter with projected sales of 15,000 to 25,000 per year, the costs were insignificant compared to the price of using, for instance, a Z80 with the necessary peripheral support chips.

Many companies can't justify the cost of building everything from scratch, and there's a rapidly growing industry which supplies various types of PC-environment embedded systems. These include normal PC hardware with a special BIOS (which doesn't look for disk drives, etc.) and (often) a ROMed version of DOS.

Systems like this allow you to develop your embedded systems using the powerful compilers and tools available on the PC (you can, for instance, use C or C++).

Commercial PC-environment embedded systems tend to be expensive; not as expensive as most other alternatives, but often as expensive as an XT with a hard disk. The prices drop dramatically if you get into volume.

My favorite example is Intel's "Wildcard 88," which has the core of an XT motherboard on a card 2" x 4". Unfortunately, there isn't any RAM (an Intel spokesman said, "we don't make RAM, so why should we put any on the board?"). But these boards are only $50 apiece, in lots of 1000!

Since XT motherboards are under $100 (and you don't need the usual hefty power supply since there won't be disk drives), why can't we build our own embedded controller (in quantities of one) with the power and development environment of a PC for under $200?

We can, if we can negotiate the tricky parts—

1. Figuring out how to modify (or write from scratch) the ROM BIOS.
2. Using RAM — probably static RAM, since it uses less power and doesn't need refreshing.
3. Establishing nonvolatile memory, since we don't have disk drives (the subject of this article).
4. Writing C or C++ embedded applications and burning them into EPROM.

This is a big tricky ticket, and it's going to take us several issues before we have all the nooks and crannies explored and worked out. But that's where we're headed.

EEPROM

EEPROM allows you to store information not lost when you power down. It's like an EPROM which doesn't require an EPROM burner — it gets "burned" right in the circuit. EEPROMs take much longer to write than they do to read. The read cycle is on the same order as normal memory.

You can only write to a memory location of an EEPROM a limited number of times (for the chip I'll talk about here, it's 10,000). However, there's no restriction on the number of times you can read it. Normally you erase an EEPROM by writing over old data, but new crops of "flash erasable" chips are showing up.

Embedded systems often use EEPROMs as long-term storage for state information (since there are no disks) or for storage of information which can't rely on the presence of a disk.

As EEPROMs get cheaper, it becomes feasible to begin using them to store large amounts of data. Several companies make cards containing EPROM with ROMed DOS and large amounts of EEPROM.

These cards look like bootable DOS disks. You simply load your program onto the EEPROM and install it in your target system as drive A, so the computer boots from the card. This looks like it might be an alternative way to create an embedded system from a PC. However, the price of these cards has always been too high for me.

The Chip

I selected the chip for this project by going to the back pages of BYTE and looking for EEPROMs. I always try to use easy-to-find parts, but sometimes people still have trouble, so I felt this was the safest approach.
This part is a Samsung KM2816A-25 (compatible with Xicor X2816A) 2K by 8 bit (i.e., 2K bytes) TTL-compatible 5-volt-only EEPROM. The "25" in the part designation refers to the 250 nanosecond read time. It's available through Jameco for $6.25, but they do have a minimum order of $20. I suggest you get the data sheet ($0.50 extra).

Jameco Electronics
1355 Shoreway Rd.
Belmont, CA 94002
(415) 592-8121

If you need a denser part, the 2865A-30 is 8K bytes for only $9.95; it should be easy to modify my design for that part.

Many EEPROMs require special hardware support for the extra-long write cycle. Or they may have other interfacing requirements (higher write voltages, for example).

The 2816A chip was designed to interface directly to a microprocessor bus. It only requires 5V for reading and writing, and has internal hardware to latch the write value.

You use the same kind of bus cycle to write to the EEPROM as you do to a RAM. (The pinout is the same as a common 6116 static RAM; you can just drop it into the same socket if the bus timing is within specs.) The only caveat is you must wait 10 milliseconds between writes.

In an embedded application with hardware dedicated to a specific task, it would be appropriate to take over some of the PC's memory space for EEPROM, some for static RAM (lower power, fast — no wait states, and no refreshing necessary), and some for the program EPROM.

On a general-purpose PC, however,
we can't dedicate any of the memory space to EEPROM so I've designed it to use the I/O space. Even then, there isn't enough free I/O space to support 2K bytes (only 10 lines, or 1 K, of I/O is brought out to the PC bus, with much of that dedicated to other purposes).

To test the EEPROM, I've designed a card which occupies 8 I/O addresses. (For an introduction to designing PC adapter cards, see my article in Micro C #43, Sept./Oct. 1988, or chapter 11 of my book, Computer Interfacing with Pascal & C.)

The card only uses three of the addresses. You write address and control line information to BASE and BASE+1. (These are write-only.) You write data to and read data from BASE+2.

Once again, I've taken a chip designed to interface directly to a bus and made it independent of the bus through a combination of hardware and software (see Micro C #42, July/Aug. 1988, or chapter 10 of my book). Maybe it's a step backward, but I find it a very useful technique for testing hardware.

Figure 1 shows the complete circuit for the chip. Most of the circuit is simply the PC adapter card interface described in issue #43. Two latches were added to hold the address and control lines to the EEPROM; the rest is done in software.

Figure 2 is the assignment of the bits in BASE & BASE+1 (BASE+2 is simply the data bus, so I didn't show it here). Notice I've combined address and control functions in the two bytes. I could put the control lines in a separate byte, but that would take more hardware. If you modify this design to add more EEPROMs, however, you might want to separate the control lines.

Figure 3 shows the two functions —

EEPROM_write()
EEPROM_read()

and a very simple program which uses these functions.

If you program in another language, you should be able to find similar functions to substitute for import() and output().

Cycling The Lines

Reading the chip is a two-step process. First, you write the address to BASE, latching it onto the 2816's pins, and force CE (chip enable) and OE (output enable) low by writing to BASE+1. (These lines are active low.)

Then you can read data from BASE+2.

You can save power by raising the CE line (again a write to BASE+1) following the read. While unnecessary in our system, this happens automatically if the chip is interfaced to a bus.

For a write cycle, you send the address (BASE) and data (BASE+2) to the chip. Then you enable the chip by dropping CE. Here, OE must be high (since we're WRITING to the chip, outputs are not enabled). Finally, you lower and raise the WE line to generate a write pulse.

After the write, you raise the CE line to disable the chip and then twiddle your thumbs for 10 msec to make sure another write doesn't happen too quickly. (This isn't necessary if you have some other insurance that your program won't write a byte more often than once every 10 msec.) To write large quantities of data while performing other tasks, you might want to use an interrupt routine tied to a clock tick.

In a system where the EEPROM ties directly to the microprocessor bus, the microprocessor and the decoding logic generate signals to the chip, so they look other than they do here. In that situation, you must examine the timing on your bus and compare it to the spec sheets for the 2816A.

Circuit Notes

You'll notice I left off the power and ground pins on the chips. I drew the diagram with OrCAD (what a wonderful program!) and the parts came from the library this way, which is very common on circuit diagrams. Fortunately, there's a pattern. If you look down at the top of the chip with pin one in the upper left corner, you'll see the lower left pin as ground and the upper right pin as power.

Another "hidden feature": every two or three TTL chips, you should put a capacitor with a value from 0.01 to 0.1 μF between power and ground to keep TTL noise from propagating through the supply lines (called "bypass capacitors").

Editor's note: Bypasses should be low-impedance ceramic or plastic capacitors and you should keep their leads as short as possible. In fact, it doesn't hurt to have one tied between +5V and ground at every IC. Also add three or four 10 to 50 μF tantalum capacitors between the 5V line (and any other supply lines) and ground.

Tantalums are polarized (like common electrolytics) so be sure someone shows you which lead is plus. Also, tantalums have a voltage rating. Those rated at 20V or greater will work fine for 5V and 12V supplies.

The DIP switch controls address selection. I've shown it as "optional" since you can simply wire the lines to ground or leave them open. To select an address for BASE of 0x220 (generally unused), ground all lines except A5 SEL and A9 SEL. The 74LS682 has internal pullup resistors so any lines left open will be forced to "1."

The backslashes ("\") after signal names denote "not" or "active low." OrCAD won't let me draw bars above signal labels (which would have been prettier). At least, I haven't figured out how to do it.

For quick bread-boarding, you can get a superstrip card that plugs into the PC bus from Global Specialties. The part number is PB88 or PB88-2. They also have a new proto-development card with all the decoding and buffering done for you (PCL-750). Call them for a catalog.

Global Specialties
P.O. Box 1405
Newhaven, CT 06505
(800) 345-6251
Figure 3 — 2816 EEPROM Reads And Writes

/* Turbo C program to read and write data on the 2816 EEPROM by Bruce Eckel, Eisys Consulting */

#define BASE 0x220 /* address set on DIP switch */

void EEPROM_write(int address, unsigned char value) {
    /* See figures 1 and 2 to understand this function */
    outportb(BASE, address & 0xff); /* Low 8 bits */
    /* High 3 bits and CE forced low */
    outportb(BASE+1, (address >> 8) & 0x07);
    /* Write the data: */
    outportb(BASE + 2, value & 0xff);
    /* 'delay()' is a function from Turbo C 1.5 to wait for a number of milliseconds. If you have version 1.0, write a simple "for" loop. */
    delay(10); /* Wait 10 mS for EEPROM to finish writing */
}

int EEPROM_read(int address) {
    /* Place the address at the 2816 pins and force CE low: */
    outportb(BASE, address & 0xff);
    outportb(BASE+1, (address >> 8) & 0x07);
    /* Read the data: */
    return inportb(BASE + 2);
}

main() {
    int i, address, value;
    /* First, dump the contents of the EEPROM: */
    for (i = 0; i < 2048; i++) {
        /* Generate breaks & line numbers: */
        if (i % 20 == 0) printf("%d: ",i);
        printf("%d
",EEPROM_read(i-1));
    }
    while(1) {
        printf("Read, Write, or Quit?\n"); /* Generate a simple menu: */
        switch(getch()) {
            case 'R' : case 'r' :
                printf("address to read? ");
                scanf("%d",&address);
                printf("%d\n", EEPROM_read(address));
                break;
            case 'W' : case 'w' :
                printf("address to write? ");
                scanf("%d",&address);
                printf("%d\n");
                printf("value to write? ");
                scanf("%d",&value);
                printf("\n");
                EEPROM_write(address,value);
                break;
            case 'Q' : case 'q' :
                exit(0);
                default:
                    break;
        }
    }
}

Next Time

I may get distracted from embedded systems sooner than I expected. I was cruising Radio Shack the other day and saw a pair of nifty LSI chips which, combined with some hardware, create speech.

I know speech projects are popular, and it seems like a PC adapter card to generate speech could be a lot of fun. Embedded systems, C++, speech synthesis, PCB cards ... there are just too many fascinating projects waiting to be created!

Editor's note: Bruce often refers to articles in previous issues. You can get a book of his hardware articles, Computer Interfacing with Pascal & C, and a disk of source code. Send a check for $30 to Micro Cornucopia, P.O. Box 223, Bend, OR 97709.

If you want to learn more about C++ programming, you can get disk #1 of the C++ Source Code Library, "Windows, Matrices & MicroCAD," by sending a check for $15 to the Bruce Eckel, Eisys Consulting, 501 N. 36th St., Ste. 163, Seattle, WA 98103.

$54.95 MASTERCARD & VISA ACCEPTED
Add $3.00 Shipping & Handling. Georgia Residents Include 4% Sales Tax.

To Order:
SEMWARE, 730 Elk Cove Court
Kennesaw, Georgia 30144
Phone Orders: (404) 428-6416
Bringing Up A Surplus 68000 Board
A New Father Describes The Experience

Last issue Karl talked about purchasing boards on the surplus market. This issue he describes the process of bringing up a surplus system. It’s a fascinating process, one you’ll miss if you limit your building to plugging together clone boards.

I wanted a fast 68000 system with lots of RAM, capable of running the SK*DOS operating system, and I wanted it for less than $200. This article tells how I did it.

I built my system around a surplus Convergent Technologies Mini-Frame 68010 board running at 10 MHz. The board boasts 512K of RAM, a floppy disk controller, a hard disk controller, two serial ports, a parallel printer port, and a 16-channel HDLC cluster controller. I added a Panasonic JU-455 floppy drive, high-current power supply and a TeleVideo 910 terminal. Total cost of the components (all surplus) — about $180.

Getting a surplus floppy drive to work seldom presents a problem; bringing up a full mini-computer without user’s manuals, technical docs, or even a schematic is another story. When I first brought my $40 board home from the surplus shop, all I had for information was what was silk-screened on the board.

There Are Easier Systems

If you want to try your hand at bringing up a surplus 68000 system, you can make your job easier by starting with a better documented board. Schematics and documentation for something like the Motorola M68KVM02 VERSAmodule family are easy to get. The boards turn up often in the surplus markets.

Or, you might stumble onto a used Amiga or Atari 68000 system, in which case there should be no problem with documentation. Of course, you can also buy these systems new, or go with the Peripheral Technology PT-68K. The latter is a good system that can be purchased for as little as $200 in kit form; drop PT a line for more info.

Checking Out The Hardware ... 

A quick look at the Mini-Frame revealed enough information to get started. The board has a single large LED near one edge, a row of five smaller LEDs (apparently a status display) down another edge, two socketed 2764 EPROMs and some 1488/1489 RS-232 drivers for the serial ports. Since the 1488s use +12V, +5V and -12V, I knew my board would need (at least) these voltages. I also found a large reset button below the serial connectors.

After perusing the board, I figured it was time to see if it worked. Using a VOM (and a pin-out of the 1488s), I located the pins on the nine-pin power plug for +5, +12 and -12 volts. When I connected my $30 surplus power supply and hit the board’s reset button, the large LED lit red and stayed on. Not encouraging.

Looking over the board more carefully, I noticed that one of the two system EPROMs had been plugged in backwards. I reversed the chip and tried power again. This time I got ten seconds worth of dancing LEDs. After the LEDs stopped flashing, the large LED went out and only a single green status LED remained lit.

Encouraged, I connected a floppy disk drive, inserted a floppy disk and powered up the board. The drive came on and the heads stepped across the disk as the board attempted to boot the alien disk. I had a working board!

Writing A New Monitor

But this board wanted to boot and run UNIX — I wanted to rewrite the code in the EPROMs to do something more interesting. This meant figuring out where all the peripheral chips, EPROMs, and RAM resided in the CPU’s 16 MB address space.

A partial disassembly of the EPROMs, provided by a friend who had also purchased a Mini-Frame board, revealed that the EPROM was addressed to start at $800000.

Since the 68010 typically uses address zero as the starting point for its exception table, it was a safe bet that the RAM ran from $0 to $7FFFFFF. (It turned out that wasn’t the case, but I’ll get to that in a bit.) But where were the peripheral chips? I had to find the address of the Intel 8274 serial chip before I could talk to my board.

There was only one way into the machine — through the ROMs where the boot code resides. I would have to burn a test routine into a pair of EPROMs, install them in the board and run the program.

My routine would scan through memory, reading each address. When it hit the address containing the serial chip, the chip’s select line would go low. Once I found the block which contained the port, I tested smaller and smaller areas within that block to locate the address.

The Host, The Target, And The Tools

Obviously, the Mini-Frame wasn’t ready to be the development system, so I needed a host and some tools. I used a PC-clone running the X68K cross- assembler from 2500 A.D. Software. To burn the EPROMs, I spent weekends with my company’s DATA-I/O Model 19 PROM programmer.

I knew I was going to be reprogramming and inserting the Mini-Frame’s ROMs many times, so I plugged machined-pin AUGAT sockets into the board’s original ROM sockets. That way I could use the upper, removable sockets as much as I wanted, without damaging the lower, soldered-in sockets.

For hardware tools, I had only a simple LED logic probe. When touched to a circuit point, the probe would show a logic one, a logic zero, or a pulse. I used
the logic probe mainly to track down the addresses of the different peripheral chips, by looking for a pulse on the chip select line.

Most of all, a large library of technical data made this project possible. I obtained data books on all the major chips in the Mini-Frame, knowing I would need the information at least twice. I used the pinout data as I hunted for the chip's address in the memory map. I later used the programming information when I wrote new code for my own monitor and programs.

Finding The Serial Port

Now begun the hunt for the address of the 8274 serial port chip. What followed was a long, tedious cycle of coding, programming the EPROMs, running the code, noting the results and trying again. The program repeatedly read every address in a selected range, then started over again. As the program ran (if it ran), I would hold a logic probe on the serial port's chip select, looking for a select strobe.

The board crashed many times, leaving just the large LED on. This meant that I had read a forbidden address. Unfortunately, there seemed to be several such areas of memory.

Finally, I tracked down the serial port at address $C30000. Armed with the proper Intel manual, I wrote a routine which sent a stream of asterisks to the terminal. Then I installed the new code in the EPROMs and fired up the board.

Unfortunately the big LED stayed on, telling me the code had crashed. I rewrote that code a couple of times, to be sure I got it right, but the result was always the same — the board crashed.

The code was correct; there had to be a problem elsewhere. Since I used subroutines to do the initialization of the serial port, the RAM had to be working for the code to work. Perhaps the RAM wasn't there after all.

Activating The RAM

Figuring I had overlooked something, I went back to the disassembly of the original EPROMs. The code started out by writing a pattern of words to particular addresses. After I added this code to the beginning of my routine, I got my stream of asterisks on the terminal. Apparently, writing just those words to just those addresses activated RAM.

Later, I found that system RAM could be moved about in the CPU's logical address space by changing the data written into the mapping RAMs. This was verified later still when a friend acquired a set of schematics for the Mini-Frame. For now, it was enough to know how I could turn on the RAM.

The Beginnings Of A Monitor

By now, I had learned enough about the Mini-Frame to begin writing a small, EPROM-based monitor. I built some simple diagnostics into the first section of the monitor's code, using the five status LEDs. As the monitor went through its startup sequence, it would light different LEDs. A failure of any test crashed the system, leaving a distinctive status pattern. If all the tests passed, only a single green LED remained lit.

I included the system testing because several friends had also purchased surplus Mini-Frame boards and the tests would help them get their boards running.

Finally, before the monitor prompted for a command, it would activate all 512 KB of RAM and test it, displaying on the terminal the results of the test. The first time that baby came up and announced that all 512 KB tested good, I felt great.

Dumping, Scanning, And The RAM Loader

One of the first commands I installed in the monitor was a simple ASCII dump routine. This allowed me to quickly explore areas of the address space, looking for goodies. Considering the 64 KB memory of my 6809 systems, it felt strange to look at addresses above $10000; I wasn't used to numbers that large!

I tried using the dump command to locate the addresses of the other main chips, such as the WD2797 floppy disk controller (FDC). That got boring quickly; it was too much work to type in a dump command, hold the logic probe in place, and press RETURN just to read 256 bytes. I needed a way to check a lot of addresses quickly and repeatedly. So, I added the sweep command to the monitor.

The sweep command would read whatever size block I wanted, repeating the task until I interrupted it. Using this tool, I quickly found the locations of the FDC, the WD1010 hard disk controller (HDC), and the printer port. I also stumbled across at least one address that appeared to force a hard reset of the system. Earlier, this had caused me so much grief when I had to burn my searches into EPROM.

Which brings up a point — each time I changed the monitor or tested a program, I still had to burn a new set of EPROMs. This had two major drawbacks. First, it takes time to erase, program, test and install EPROMs. Second, since the EPROM programmer belongs to my company, I only had access to it on the weekends, limiting the time I could hack on my machine. I needed a way to download 68000 code directly from the host system.

Since the linker on the host system could output Motorola 68000 object records, I wrote a simple 68000 loader and burned it into the monitor EPROMs. This RAM loader command removed the last barrier to rapid progress on the Mini-Frame. I could now develop programs on the weekends, then use the EPROM programmer on the weekends to upgrade the monitor.
Tracking Down The Disk DMA

I had made a lot of progress so far. I had full control of the serial ports, access to 512 KB of RAM, and a monitor strong enough to develop simple programs. But the biggest challenge remained. How did the floppy disk system work, and could I get SK*DOS running?

It was about this time that my friend acquired the set of schematics. The diagrams showed the DMA section, and my sweep routine located the DMA controller. Using the disassembly of the original EPROM code, I put together a simple set of routines to read data from a PC-formatted floppy disk.

With a little bit more work, I also wrote data to the disks. After all this, I figured it wouldn’t be too difficult to generate code which would read and write SK*DOS disks.

Cleaning Up Little Things

What remained? I wanted to change the serial baud rate at will. A little re-sweep routine located the DMA controller, mainly because I don’t have a spare hard drive. That will make a good rainy day project. I also want to add some battery-backed static RAM to the machine. This will let me put SK*DOS’ RAM disk in nonvolatile RAM. (Yes, I do have SK*DOS running on the Mini-Frame and it’s a terrific DOS!)

I also intend to add an adapter so I can run PC-compatible cards on the Mini-Frame. The address and data busses come out on a 100-pin connector mounted on the board. I just need to find the matching connector and wire up the interface. The idea of EGA graphics (or better) on my 68010 system gives me tingles.

What Have I Gained?

I now own a fast 68010 system, running an excellent DOS. I learned a tremendous amount about 68010 hardware and system design from exploring this machine. I certainly saved a lot of money, but mostly I had a heck of a lot of fun doing a difficult hack on a terrific piece of hardware.

And Thanks To …

Any project this difficult needs more than just one person on it; several people contributed time, talent and support to get the Mini-Frame to this point. Mark Loomis provided the disassembly of the original EPROMS and insights into the 68000 CPU. Mike Sutherlin scrounged up a great set of schematics. Bruce Beyerler offered encouragement and technical support.

Particular thanks are due Brian Bierer for getting me started. He found the Mini-Frame boards at the local surplus shop. He also added hardware expertise and enthusiasm to the project. A special “thank you” to Bill Lieske, Jr., for the support, encouragement and patience he has shown for the group of hackers who congregate each month at EMR Co.’s metal shop in north Phoenix.

About The Vendors

PT has been a leader in 6800/6809 technology for years. People whose judgement I trust give their PT68K 68000 single-board computer a very high rating for design and performance. Considering the basic kit only costs $200, it would be tough to find a cheaper way to enter the 68000 world.

Peripheral Technology
480 Terrell Mill Rd., Suite 870
Marietta, GA 30067
(404) 984-0742

Peter Stark of Star-K Software has developed first-class software for Motorola chips clear back to the 6800. He’s turned out some superb programs over the years, but SK*DOS for the 68000 is something. Lately, he has worked with Radio-Electronics magazine (in the Computer Digest section) on a series of articles dealing with the PT68K board and SK*DOS. Check out back issues of the mag for solid information on both the 68000 and the PT68K.

Star-K Software Systems Corporation
Box 209
Mt. Kisco, NY 10549
(914) 241-0287

2500 AD Software wrote the 68000 cross assembler for MS-DOS. It sells for $299.50. They’re also working on a C compiler that runs on a PC but compiles ROMable 68000 code. It should be available around February 1. Price should be around $600.

2500 A.D. Software, Inc.
P.O. Box 480
Bueno Vista, CO 81211
(800) 843-8144
$299.50 MS-DOS

Editor’s note: Karl had to search through memory for I/O addresses because Motorola maps ports and RAM in the same space. Intel, on the other hand, has separate I/O read and write lines so port addresses can overlay RAM addresses without conflict. Both systems have their proponents, but what’s important is that the search for ports differs with different processors.

+++
ED is an editor that makes the very conception of other editors obsolete.

ED is an object-oriented open architecture system for editing programs and manuscripts, manipulating data files and creating snazzy demos. ED's commands respond with no perceptable delays. Programmers have complete simultaneous control over multiple source files. Functions for creating manuscripts are part of ED's design, with stream blocks, word-wrap, tabs, intelligent paragraph formatting, block justification, and pagination that behave correctly and execute instantly. ED can be used to create sophisticated sorts and filters. Macros, search and replace, column blocks, and block sorts allow records to be sorted and selectively removed, and fields to be added, rearranged, and removed.

Flexibility, extensibility, and programmability are realized by providing programmers access to the objects which ED itself manipulates. All aspects of ED's appearance and operation can be controlled by the programmer. ED, a restricted or enhanced form of ED, or any of ED's objects, can be embedded in the programmer's applications with no royalties. Objects such as dynamic arrays, windows, data entry windows, macros, menus, browsers, popup directories, and regular expressions are manipulated through normal C function calls.

ED

- Number of Buffers Limited Only by Available Ram
- Number of Windows Limited Only by Screen Size
- Popup Directory Facility
- View FileSpec/Subdirectories
- Find File on Disk
- Search Disk Files for Text
- Copy
- Rename
- Delete
- Browse
- File Status
- Edit
- Execute
- Sort
- Search and Replace
- Forward/Backward
- Case Sensitive/Insensitive
- Fully Parenthesized Regular Expressions
- Incremental/Global Search and Replace
- Across All Buffers
- Block Commands
- Column/Line/Stream Blocks
- Save
- Cut
- Delete
- Blank
- Search For
- Format
- Justify Left/Right/Center
- Uppercase/Lower Case
- Move
- Overlay
- Reactivate
- Tab
- Drag
- Sorts
- Spreadsheets
- Style
- Math
- Add/Sub/Mult/Div/Dag
- Macros
- Menu Driven
- View
- Auto-Execute
- Time-Delay
- Nested
- Recursive
- Interactive
- 1 or 2 Keystroke
- Screen Display Control
- Size Limited by Available Memory
- Automatic Macro Menu Creation
- More Functions
- Pulldown User Menu System
- Reconfigure Color and Window Style
- View/Manipulate Buffers Through Menu
- Or Keystroke
- Intelligent Diagram/Box Draw
- Adjustable Scroll Value
- Transpose Lines/Characters/Words
- Move-to-Position Stack
- Visual Tab Creation
- Popup Ruler
- Column/Line/Stream Tabs
- Time and Date Stamp
- Restore
- Typed-Over Characters
- Change Directory
- Bracket/Brace/Parenthesis Matching
- Enter Graphics Characters
- Execute Dos Commands
- Popup Ascii Table
- User Defined Popup Files
- Keyboard Configuration
- Print Buffer With Page Numbers
- Adjustable Spacing and Margins
- Interactive Tutorial
- Manual and Tutorial On-Line
- Written in C and Assembler
- External Utilities
- Intelligent Change Directory Command
- Popup Directory Facility
- String Translator
- Hardware Requirements
- Dos
- Pcx/Tat
- Ps2
- 386
- CGA, MDA, EGA
- Hercules
- Wyse 700
- Amdek 1280
- Runs In All Video Modes, No Flags, No Drivers
- 256K
- For Literature and Demo:

Call 201-450-4545

Version 1.1
Price: $265.00

Mastercard, Visa, C.O.D., and P.O.'s

Reader Service Number 107
Practical Fractals
And Other Flights Of Fractal Fancy

Larry obviously didn’t understand the true import of last issue’s Culture Corner (he thought I was encouraging his fractilious schemes). And now, of all things, he thinks he’s found a practical application for these twisted notions. (We’re trying to humor him, really we are.)

Kicked back in my chair, feet up on the desk, I watch yet another Mandelbrot plot crawl across the screen. A sarcastic voice behind me says, “Fractals? Again? When are you going to do something useful around here?”

An annoying and utterly unfair remark. I do plenty of useful work at Micro C. In summer I mow the lawn (well, once anyway), and in winter I shovel snow from the walk. What more do they want?

As it turns out, plenty of folks are working on practical applications of fractals. But, before we get into that, let’s go over some fractal theory from a less mathematical perspective — one that will make clear the potential for use of fractals in the real world.

Empirical Theory

Enough obscure esoterica has been written on fractal theory to supply the Micro C outhouse with paper products for years to come. (Indoor plumbing? In Central Oregon? Be serious.) Let’s take the concept of fractal dimension and come up with a gut-level definition.

Many objects and processes in nature exhibit “self-similarity” at different scales. That is, they look the same under varying magnifications. England’s coastline is a classic example.

From space we see an irregular line made up of bays, peninsulas, and amusement parks. A walk along the beach shows similar detail, but on a much smaller scale. And the view through a magnifying glass includes miniature baylettes, peninsulettes, and parkletteles — again looking similar to the view from space.

This “scale invariance” of the coastline’s appearance is a major characteristic of fractals. It makes the length of the coastline hard to pin down. A measurement taken from space misses much of the small detail and therefore will be smaller than one taken by a man with a yardstick. But we can measure the rate of increase of the coastline length as we zoom in.

It seems reasonable that more complex coastlines, like England’s, would show a greater increase in length than, say, Southern California with its long, straight beaches. So this rate of increase directly measures the complexity of the coastline. It’s also related to the coastline’s fractal dimension.

In more specific terms, determination of the fractal dimension of England’s coastline goes like this: Measure its length at various scales and draw a graph of the log of the length vs. the log of the unit measure. “Unit measure” means the smallest measurable unit at a given resolution — large scales have large unit measures.

Obviously, the slope of the plotted line will be negative since, as the scale increases, the length decreases. The fractal dimension then equals one minus the slope of the plotted line — a value greater than one since the slope is negative. It turns out that England’s coastline has a fractal dimension of about 1.2.

We can test this notion with a geometrically straight line segment. Since it has no irregularity, measurements of its length will give the same value at all scales. Plotting length and unit measure as before shows a line of slope zero — no change in measured length. And (1-0) gives a dimension of one, like all good Euclidean lines should have.

Despite my good intentions, a bit of unexplained math has crept into the discussion. The faint of heart may want to skip to the next section, but for those with a penchant for completeness, the general case follows:

$$P(E) \propto e^{e^{d}}$$

where $P(E)$ represents the value measured (here, the length of the coastline), $e$ is the unit measure, $d$ the topological dimension, and $D$ the fractal, or Hausdorf-Besicovitch dimension. Think of $d$ in the more traditional way — a dimension of integer value. (Coastlines have a topological dimension of one, and for planetary surfaces, $d=2$.)

By the way, one definition of fractals says that a fractal has $D>d$. Again, this makes sense: $D=d$ gives us straight lines and flat planes — definitely not fractals.

Fractals In Oregon?

One of the nice things about living in Bend is having a wild river running through town. At least, the mink, otter, and beaver seem to like it. (Our own governor once refused to visit Bend — “The middle of nowhere” as he called it.) What better place to try my hand at fractalizing than on a nice crooked river.

No, I didn’t actually pace the banks of the Deschutes for a small scale measure: I let my dividers do the walking on a map. An afternoon of diligent dividing in the Central Oregon Community College map room generated the data I needed. It’s hard to come up with more than one order of magnitude in scale variation this way, but it does serve as an example of the technique. (See Figure 1.)

By my approximation, the wanderings of the Deschutes earned it a fractal dimension of 1.25 — not an unreasonable value. I’m sure the otters (and Governor Goldschmidt) will be glad to know.

Finding The Fractal Dimension

(I know — sounds like an episode of Flash Gordon: Flash Explores the Fractal Dimension.)
Given an object in the "3-D" real world, The Real World for example, we can look at its diameter at different scales. Or we could consider the same scaling effect on its surface area. Again, the difference between mountains on the grand scale and molehills on a lesser scale gives us the information necessary to find the fractal dimension of the Earth's surface.

Gathering this data poses a difficult task, but researchers have developed methods for approximating the fractal dimension of a surface.

One possibility involves taking a horizontal slice of the surface (a contour line), finding its fractal dimension, and adding one. A fairly coarse approximation, and one that may depend on the elevation of the contour.

Or you could look at individual points on the surface and investigate their elevation difference vs. the distance between them. Or you can choose from other computationally intensive methods.

K.C. Clarke came up with a simpler method. Overlay the surface with the largest square that will fit. Take the average of the elevations of the four corners and assign it to the center of the square. Now find the total area of the four surfaces of the resulting pyramid.

This gives a first large scale measure analogous to measuring a coastline from space. Next, overlay four smaller squares on the same surface and measure their areas in the same way. The smaller scale will yield a larger value for the total surface area since it will catch more of the surface's details.

Continue this process to the lowest limit of scaling (two pixels by two pixels).

---

**Figure 1 — Finding The Fractal Dimension (D) Of The Deschutes River**

\[
\text{Slope} = \frac{0.25}{-1.0} = -0.25
\]

\[
D = 1 - (-0.25) = 1.25
\]
for a digitized image). Then plot the log of total area vs. the log of the area of an overlaid square, for each size of unit square. Very similar to the coastline method described above, Clarke’s method saves a bunch of computer time over other fractal surface dimension techniques.

A Practical Example

Professor Philip J. Hopke (of the University of Illinois at Urbana-Champaign Environmental Research Laboratory) and his associates have used Clarke’s method in an effort to characterize particles by their fractal surface dimension. They hope to use the fractal dimension as a tool to identify the processes that form airborne particles.

Hopke captures the particle’s image from a scanning electron microscope (SEM). The SEM bombards the surface of the particle with a tightly focused beam of electrons. The surface, in turn, reemits secondary electrons whose energy depends on the topology and composition of the surface. This digitized secondary electron image provides the data necessary to find the fractal dimension.

Particles of sodium chloride, sodium sulfate, ammonium sulfate, and one unknown compound have been subjected to this technique. Like most of the research I’ve seen, this application seems to work well over a somewhat limited range of scales. Hopke reports very consistent results at lower magnifications.

However, high magnification (>100K) yields lower values for the fractal dimension. Nonetheless, a judicious choice of magnification may yield useful information.

To quote Professor Hopke, “If we can work out the remaining kinks in our methods, fractals could prove to be a valuable tool in investigating atmospheric particles.”

Other Applications

Fractals have shown promise in a wide variety of fields, often in finding a quantitative measure of the irregularity of some property, as in Professor Hopke’s research. Other areas are:

- Characterization of the roughness of the ocean floor.
- Distribution of earthquakes along fault systems.
- Optical diffraction analysis of particle clusters formed by diffusion limited aggregation (DLA).
- Clustering of stars.
- Spreading of diseases.
- AC behavior.
- Growth of DLA clusters.
- Simulation of lightning.

Data compression, linguistics, turbulent flow, population growth, and most anything else you can imagine have been explored with fractal techniques. The list is truly endless.

.CO.Mplexity

I wanted to do something keen like a fractal measure of program complexity. But this points out a danger: Not all objects and processes are fractal in nature. If experimentation shows scale invariance of some property, then by all means explore its fractal possibilities.

I’ll keep thinking about it, but it doesn’t look like executable programs are fractals. (Except, of course, fractal programs.)

The Bottom Line

A few years back, John Maddox wrote, “...this is an interesting case, which seems to happen more and more often, of how an explanation in search of a phenomenon may lay claim to far more territory than it can handle.”

This still holds true for the most part. But while fractals remain the darling of a large portion of the scientific community, a healthy case of skepticism has set in. A myriad of applications tease researchers with promising results, but no one seems willing to claim absolute success in modeling, characterization, or any other application of fractals (except graphics).

Still, it won’t surprise me in the least to see fractals become an increasingly important tool in the future.

Commercial Fractal Software

This here’s a computer rag so let’s finish up with a look at some software available for interdimensional explorations. I know of only two commercial fractal packages for the PC: Mandelbrot Explorer (ME), and FractalMagic (FM). Both do a fine job of tying up your computer at a bargain basement price.

According to the Micro C Fractal Accessory Review Team (Allan Chambers, Dave (that Thompson guy), and myself), ME and FM do just about anything you’d want with Mandelbrot and Julia sets.

Both packages allow storing work in progress for resumption at a later time. (Handy if you occasionally use your system for such mundane tasks as word processing.) They also let you define parameters for new plots by moving a window around an existing plot, or by entering parameters directly.

While both programs let you “animate” or dynamically change the color map of an image, FM gives you more control over the process. I could watch an animated image for hours.

Finally, both support math coprocessors — a definite must for serious fractal work.

On to the differences. FM has mouse support, ME doesn’t. I don’t consider this a deciding factor — the cursor keys work just fine.

If you have a CGA system, get FM since ME won’t run on CGA. Better yet, buy an EGA or VGA setup. I used a VGA/386 system at MicroSphere to test these programs and I’m hooked. I’ve upgraded the recommended minimum system to EGA. CGA and Hercules fractals just don’t cut it.

Mandelbrot Explorer

Mandelbrot Explorer sports online help, very powerful control over color assignment, some picture editing capability, and tips on photographing screens.

In the PC tradition, ME accepts command line parameters and operates with keystroke commands. The command line capability means that you can easily set up a batch file directing ME to do any number of fractals unattended. Wonderful! Head off on a month-long vacation while your computer sits home alone scratching out fractals.

Another nice feature called Escape lets you watch the dance of a point as it goes through the iterative process.

I do not like ME’s method of screen boundary input. Instead of taking maximum and minimum values, ME wants to see the coordinates of the center of the region and a “magnification.” Very unnatural.

Fractal Magic

Written in Turbo Pascal, FractalMagic comes in two flavors — regular, and 80x87. And if you want to create your very own fractal set, Advanced FractalMagic ($25) lets you write new calculation routines that can be linked in with the FractalMagic code. (You’ll also need Turbo Pascal v.4.0.)

FM uses pull-down menus to create a very nice user interface. Screen parameter input follows the more standard method of specifying maxima and minima. During operation FM shows a bright dot at the point of calculation — a very nice touch that makes it easy to see where the heck you are.

The makers of FM also sell a screen dump utility, T-shirts, and another inter-
estimating sounding graphics program called KaleidoScope.

Which One?
A rough comparison using the full Mandelbrot set gave FM about a 15% faster rating. I didn’t get a chance to compare the two on an 80x87 system.

I’d be hard pressed to make a choice between the two programs. I prefer the user interface of FractalMagic, and it has the edge in speed. But Mandelbrot Explorer’s batch ability and its slick Escape feature tip the balance back towards even.

Shoot, it’s only money — buy ’em both.

Mandelbrot Explorer $30
Peter Garrison
1613 Alvito Way
Los Angeles, CA 90026

FractalMagic $35
Sintar Software
P.O. Box 3746
Bellevue, WA 98009

References


MICRO CORNUCOPIA, #46, Mar-Apr 1989 35
Hacking Sprint: Creating Display Drivers

Brett was one of the original architects of the IEEE 802.5 Token Ring LAN, and coauthored Living Videotext’s ThinkTank 2.0. He’s also an instigator of the annual Hackers’ Conference, which makes him a natural for hacking Sprint, Borland’s flexible new word processor.

Life in the Silicon Valley isn’t quite as idyllic as it is in Central Oregon, but it has its pleasures: the challenge of identifying smog-shrouded friends from 6 feet and the romance of the regional car crawl during rush hour.

My desire to do something other than fight the gridlock one Friday evening led me to crack the imposing manuals I’d received with Borland’s Sprint (the word processor).

As it turned out, the result justified the effort. If you’re picky about your editing environment, learning how to hack Sprint will give you unprecedented control over the way you view or manipulate text.

A Complete Word Processor

Sprint is the progeny of a long and distinguished line of text editors and formatters that began at MIT and CMU with Richard Stallman’s EMACS (Editing MACroS) editor and Brian Reid’s Scribe text formatter.

Versions of these two programs, mainstays of the academic computing community, first appeared on microcomputers as MINCE (“MINCE Is Not Complete EMACS”) and Scribble. These were the foundations for products such as Perfect Writer, Amethyst, and Final Word.

Sprint, derived from Final Word II, provides almost every feature a user could ask for. Imagine a text editor that comes with nine user interfaces, four built-in “languages,” a text formatter, a spelling checker, a thesaurus, a file format converter, a mail merge, an automatic save feature, and support for dozens of printers, screens, and terminals.

Sound intimidating? It might be, had Borland not invested a great deal of time and effort to hide much of the complexity of this rich environment from the novice.

Sprint emerges from the box accompanied by a menu-driven setup, pop-up menus within the editor, and well-chosen defaults. Many of you will never need to delve any deeper into the mysteries of Sprint.

However, if you do get into the “advanced” portions of the documentation (the Reference Manual and the Advanced User’s Guide), you’ll discover that Sprint has unmatched facilities for user customization and enhancements.

Figure 1 shows an overview of the Sprint system. Note that, at every place where there’s an arrow in the diagram, a user-accessible “language” can create new commands and menu text.

A C-like macro language controls the interpretation of keystrokes and allows redefinition of EVERY KEY COMBINATION. So the underlying EMACS-like editor can emulate the user interfaces of WordPerfect, Microsoft Word, and SideKick.

A configuration “language” controls how Sprint interacts with the screen and printer. The Scribe-like formatting “language” helps to organize your output. And a record-oriented “language” lets you control the SprintMerge mail merge. This configurability makes Sprint a hacker’s editor par excellence.

Adapting Sprint To A New Screen

Each of the languages and interfaces in Sprint deserves a small book of its own. In this article I’ll focus on one of Sprint’s more unusual features — its ability to run nearly any video display adapter or terminal.

During the course of my experimentation with Sprint, I’ve sent characters to the screen via direct memory mapping, software interrupts, the IBM PC BIOS, MS-DOS, a serial port, and I/O ports — all without major problems. Because you can load specific processor registers with selected values and call a software interrupt, you can write a TSR that does almost anything with Sprint’s screen output.

To provide a real-world example, Earl Hinrichs of PC Tech was kind enough to let me borrow a high resolution monochrome video board and monitor to test Sprint’s configurability.

The PC Tech video board, which uses a TMS34010 graphics CPU, produces a 736x1024 “portrait” display with up to 66 lines by 80 characters. (It looks as if it’s possible to get still more characters on the screen, but this is the largest number currently supported without a special driver.)
I started out by reading Appendix F of the Sprint User's Guide ("Build Your Own Screen and Printer Drivers"). This section describes the format of a Sprint "library file" — a file which describes screens and printers to the Sprint configuration utility (SP-SETUP).

A library file consists of ASCII "records" that define the characteristics of printers, screens, and I/O ports. If you’re defining a printer device, there's also a construct called a "subrecord" that defines fonts, character widths, attributes, and character translation for a printing device.

The Library Record Format

A record in a library file must begin at the left margin and start with the word "printer," "screen," or "port." Each record consists of the name of a device, followed by a comma and one or more fields. Each field gives information about the device, and consists of a field name, a space, and a value (if required) for the field. Commas separate the fields.

A record can span more than one line; if it does, lines after the first must be indented. Comments, which begin with double semicolons, can appear in a record; if they appear in the first column, the setup program displays them as you configure Sprint.

Library records for screens have fields that specify a large number of options. Figure 2 shows an example of a library record (taken directly out of Borland's library of screen definitions) for a screen accessed via the IBM PC BIOS.

Let's examine this record bit by bit. First, the word "screen" in the first column indicates we’re defining a screen device; the name of the device, "IBM PC BIOS", follows immediately thereafter.

The fact that the name of the device begins with a period has special significance. Devices whose names begin this way won't appear on the setup program's menu, but you can create devices that are visible on the menu with a record like —

```
screen IBM PC BIOS, as .BIOS
```

This record will cause an entry called "IBM PC BIOS" to appear on the screen setup menu, and the special "as" field

![Figure 1 — Sprint System Overview](image-url)
will invoke all the same field settings as the hidden "BIOS" device would.

The first three fields of the .BIOS record, "cur," "up," and "down," are set to the capital letter "B." This setting has a special meaning: It causes Sprint to use the IBM PC BIOS to position the cursor and move it up or down. It's also possible to place strings in these fields which convert the row and column into a cursor positioning string to be sent to a terminal.

The next group of fields, which includes "help," "infobox," "menufirst," "menu," "error," "status," "select," "plain" and a series of control characters, selects the attributes used to paint the screen. You can modify these hex numbers for maximum visibility, or simply to suit your tastes.

Finally, the "reset" field tells Sprint what to do when you exit the editor. The string in our example manipulates the BIOS to clear the screen.

Then, the sequence —

```c
O>DX>BH 2 int 10h
```

puts 0 in DX and BH, and 6 in AH, then invokes software interrupt 10h. This causes the BIOS to clear the screen.

Then, the sequence —

```c
O>DX>BH 6 int 10h
```

puts 0 in AL, 0 in CX, 1850 hex in DX, 7 in BH, and 6 in AH, then invokes software interrupt 10h. This causes the BIOS to clear the screen.

The PC Tech Monochrome Display

I installed the PC Tech Monochrome Display in my PC almost without glancing at the instructions. The monitor has an automatic on/off feature that causes it to turn on when your PC boots. It doesn't require a separate power switch — a very nice touch.

I turned my PC on and the board came up emulating an IBM CGA card. The emulation was good enough, by the way, that even IBM's OS/2 Standard Edition 1.0 — known to be sensitive to minute hardware incompatibilities — didn't notice the difference.

The display, which uses two bits per pixel, maps colors to combinations of black, dark grey, light grey, and white, allowing most CGA programs to produce readable output despite the lack of color. However, because the display was acting like a CGA, it limited the text to 25 lines at the top of the screen — a terrible waste of high-resolution display space.

Fortunately, you can expand the PC Tech display to use more of the screen by loading a special TSR, called TSR10, that communicates with the TMS34010 graphics processor. Once you've loaded the TSR, the command —

```c
MOVECRN CS 80 66
```

maps more of the display area into the PC's memory, yielding an 80 by 66 display.

However, at this point Sprint still wouldn't have known that the display wasn't a CGA. So I created a new screen record (Figure 3) and inserted it into the MAIN.SPL file on my Sprint program disk.

This record contains several tricks I'd picked up by looking at other device descriptions in the MAIN.SPL file.

First, I wanted the menu entry on Sprint's configuration menu to show the string "PC Tech, 66 lines." However, in order to keep the setup program from mistaking the comma for the one that comes at the end of the device name, I had to prefix it with a backslash.
When I invoked the SP-SETUP program, lo and behold, the PC Tech display was on the menu — and after selecting it I was able to edit a whole typewritten page of text at one time.

Even nicer, I could open eight files on the screen at once and move back and forth between them — excellent for editing programs that span multiple files. (Then, of course, I wrote a simple macro that did a “make” on all the files I’d edited. Sprint can be addicting.)

Special Fonts And More

Just displaying the fonts as different shades of grey, black, and white (as in this configuration) is still barely scratching the surface of what can be done with the TMS34010 processor on the PC Tech display board.

The graphics CPU is capable of displaying boldface, underlined, subscript, superscript, strike-through, and other fonts on the screen pretty much as they’re going to be printed, given only a few enhancements to the software on the board. (The Hercules RAMFont card already performs some of these functions with Sprint, but since it does them in hardware it may not be as flexible.)

At this writing, I haven’t yet figured out how to modify the software on the coprocessor to display all the variations on the fonts, but it’s certainly possible — and I hope the folks at PC Tech will lend a hand.

Macros And Other Business

Sprint’s configurability in one small area (the development of display drivers) gives a hint of what it’s capable of doing in others. I’ve seen Sprint editing macros that play tunes on the PC’s speaker, edit your directory (veteran EMACS users will recognize this as an adaptation of a macro called “DIRED” on older EMACS systems), and perform other functions you might not usually see inside an editor.

The sheer size and scope of Sprint has bewildered a few reviewers. But there’s no question that at least one class of user (the ingenious hacker) will revel in the possibilities. If you fall into this category, or even if you don’t, you may want to put Sprint through its paces and see what you can create.
Great C Comparison Of 1989!

By Scott Robert Ladd
P.O. Box 61425
Denver, CO 80206
(303) 322-7294

Here we go again. This is Version 2.0 of Scott's “everything you ever wanted to know about C compilers.” In this updated version you'll get faster compiles, larger libraries, newer debuggers, and fancier manuals.

Yes, it's that time of year again ... when my family wonders about my sanity, my friends think I've disappeared, and I find myself buried in boxes, disks, and manuals. Once again, I'm reviewing the great hoard of MS-DOS C compilers.

The lineup has changed somewhat. Last year, I looked at 11 compilers; this year it's 12. Nearly all of last year's compilers (the ones still on the market) have undergone major upgrades. There are two significant newcomers—WATCOM C and MIX Power C. Zortech now sells what used to be the Datalight compiler. Mark Williams' Let's C, a fine product I recommended in '88, is undergoing a major upgrade to full ANSI compatibility. The people at Mark Williams did not want their old compiler reviewed with a new product in the wings. As soon as I get the new version, I'll cover it in the regular column.

My methods have changed a bit. I've spent the past year honing the process, and at SOG VII several of you suggested things you'd like to see.

The Latest Tests

Figure 1 summarizes the results of my latest test. For the major benchmarks, I've listed the top five finishers. Note that, as in the coprocessor run-time for GRIND, the compilers at the top are only a few tenths of a second apart. A compiler which did not make the top five may still have been very close to the leaders.

All times are shown in seconds; EXE file sizes are in K (the size of the executable image divided by 1024). All times are the average of five tests. For QuickC and Turbo C, I used the command-line versions of the compilers. Compile and link times were done with optimizations turned off; after all, an optimizer usually only runs in the final compile. I compared run times after compiling for maximum speed.

"Em" means I used the software floating-point emulator, while "87" means I used the math coprocessor. All tests ran under MS-DOS v 3.21 on a 16 MHz 80386 PC with a 16 MHz 80387 math coprocessor and a 28 msec hard drive. I packed the drive so all compiler components and source files were contiguous. I used Microsoft LINK v 3.69 if the package didn't provide its own linker.

Figure 2 summarizes the compilers' features. It lists such things as ANSI compatibility, library features, memory models, and tools.

The Benchmarks

To begin with, the benchmarks have undergone major surgery. Where there were nine benchmark programs last year, there are now five. Only two of last year's benchmarks survived. The seven deleted benchmarks ran under fifty lines in length, and supposedly tested individual features of the compilers. In reality, they tested very little.

Real-world programs do something, whereas these tiny benchmarks were simple loops containing a few statements. Also, optimizing compilers cannot show their mettle on a tiny program. This year's benchmark suite consists of longer, more complex programs — a better test for optimizers.

DHRYSTONE 2 is a revision of the classic Dhrystone benchmark. The original author re-wrote the program in C (the previous version was a C translation from Ada). He also cleaned up its logic and corrected some bugs. DHRYSTONE 2 contains a mix of statements designed to simulate a theoretically average program. Originally designed for comparing different hardware architectures, it's also very useful as a compiler benchmark.

FXREF was here last year. (See Figure 3.) This MS-DOS filter program creates a cross-reference listing of a text file. FXREF reads a file from standard input, and breaks it into text tokens. It then stores the tokens (along with line number references) in a binary tree. FXREF tests a compiler's I/O and dynamic memory allocation. The input to FXREF was its own source code.

GRIND has undergone very few changes. (See Figure 4.) It simulates a typical report pro-
Figure 1 — Benchmark Results

<table>
<thead>
<tr>
<th>Dhrystone Compile Time</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Dhrystone/Sec</th>
<th>Em EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>2.6</td>
<td>20.3</td>
<td>2463</td>
<td>9.5K</td>
<td></td>
</tr>
<tr>
<td>7.3</td>
<td>5.0</td>
<td>23.6</td>
<td>2119</td>
<td>18.5K</td>
<td></td>
</tr>
<tr>
<td>55.5</td>
<td>13.1</td>
<td>20.7</td>
<td>2415</td>
<td>16.8K</td>
<td></td>
</tr>
<tr>
<td>10.5</td>
<td>5.2</td>
<td>23.4</td>
<td>2137</td>
<td>14.8K</td>
<td></td>
</tr>
<tr>
<td>15.4</td>
<td>7.5</td>
<td>23.7</td>
<td>2110</td>
<td>12.8K</td>
<td></td>
</tr>
<tr>
<td>13.7</td>
<td>2.4</td>
<td>16.9</td>
<td>2959</td>
<td>7.7K</td>
<td></td>
</tr>
<tr>
<td>24.5</td>
<td>5.6</td>
<td>19.3</td>
<td>2951</td>
<td>26.1K</td>
<td></td>
</tr>
<tr>
<td>14.7</td>
<td>3.4</td>
<td>16.6</td>
<td>3012</td>
<td>10.7K</td>
<td></td>
</tr>
<tr>
<td>6.6</td>
<td>3.3</td>
<td>19.6</td>
<td>2551</td>
<td>10.9K</td>
<td></td>
</tr>
<tr>
<td>15.6</td>
<td>3.3</td>
<td>21.3</td>
<td>2347</td>
<td>18.6K</td>
<td></td>
</tr>
<tr>
<td>19.7</td>
<td>3.3</td>
<td>17.9</td>
<td>2793</td>
<td>9.5K</td>
<td></td>
</tr>
<tr>
<td>6.4</td>
<td>3.3</td>
<td>17.5</td>
<td>2857</td>
<td>10.0K</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compile Time Rankings</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Dhrystone/Sec</th>
<th>Em EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>fxRef Compile Time</td>
<td>4.9</td>
<td>2.5</td>
<td>41.3</td>
<td>8.4K</td>
<td></td>
</tr>
<tr>
<td>Link Time</td>
<td>6.5</td>
<td>4.8</td>
<td>42.6</td>
<td>13.5K</td>
<td></td>
</tr>
<tr>
<td>Em Run Time</td>
<td>34.0</td>
<td>20.6</td>
<td>38.7</td>
<td>11.6K</td>
<td></td>
</tr>
<tr>
<td>Dhrystone/Sec</td>
<td>8.2</td>
<td>9.0</td>
<td>42.8</td>
<td>15.6K</td>
<td></td>
</tr>
<tr>
<td>Em EXE Size</td>
<td>9.1</td>
<td>2.5</td>
<td>42.7</td>
<td>8.6K</td>
<td></td>
</tr>
<tr>
<td>fxRef</td>
<td>7.1</td>
<td>9.0</td>
<td>39.4</td>
<td>21.2K</td>
<td></td>
</tr>
<tr>
<td>Link Time</td>
<td>22.5</td>
<td>6.9</td>
<td>38.6</td>
<td>9.0K</td>
<td></td>
</tr>
<tr>
<td>Em Run Time</td>
<td>9.8</td>
<td>4.9</td>
<td>38.6</td>
<td>12.2K</td>
<td></td>
</tr>
<tr>
<td>Dhrystone/Sec</td>
<td>6.0</td>
<td>5.5</td>
<td>39.7</td>
<td>7.9K</td>
<td></td>
</tr>
<tr>
<td>Em EXE Size</td>
<td>10.6</td>
<td>5.0</td>
<td>39.1</td>
<td>12.9K</td>
<td></td>
</tr>
</tbody>
</table>

| Sines Compile Time      | 2.8   | 3.0       | 138.4       | 25.4K         |
| Link Time               | 4.0   | 4.8       | 20.0        | 16.0K         |
| Em Run Time             | 13.7  | 9.0       | 10.5        | 18.1K         |
| Sines EXE Size          | 4.8   | 2.5       | 15.4        | 9.8K          |
| Compile Time            | 5.5   | 6.9       | 10.6        | 18.4K         |
| Em EXE Size             | 4.1   | 2.5       | 10.8        | 14.4K         |

| Empty Compile Time      | 2.4   | 3.7       | 23.6        | 14.0K         |
| Link Time               | 3.7   | 8.5       | 10.5        | 16.0K         |
| Em EXE Size             | 4.1   | 4.2       | 5.3         | 5.0K          |

**Emulator EXE Size Rankings**

<table>
<thead>
<tr>
<th>Rank</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Sines EXE Size</th>
<th>Empty EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manx</td>
<td>Zortech</td>
<td>Manx</td>
<td>Ecosoft</td>
<td>Ecosoft</td>
</tr>
<tr>
<td>2</td>
<td>Borland Turbo C</td>
<td>Borland Turbo C</td>
<td>WATCOM</td>
<td>MIX</td>
<td>WATCOM</td>
</tr>
<tr>
<td>3</td>
<td>WATCOM</td>
<td>Borland C Ware</td>
<td>Ecosoft</td>
<td>Manx</td>
<td>Manx</td>
</tr>
<tr>
<td>4</td>
<td>Zortech</td>
<td>C Ware</td>
<td>Zortech</td>
<td>MIX</td>
<td>Zortech</td>
</tr>
<tr>
<td>5</td>
<td>Microsoft</td>
<td>Microsoft</td>
<td>Zortech</td>
<td>C Ware</td>
<td>Microsoft</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Compile Time Rankings</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Sines EXE Size</th>
<th>Empty EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>fxRef Compile Time</td>
<td>4.9</td>
<td>2.5</td>
<td>41.3</td>
<td>8.4K</td>
<td></td>
</tr>
<tr>
<td>Link Time</td>
<td>6.5</td>
<td>4.8</td>
<td>42.6</td>
<td>13.5K</td>
<td></td>
</tr>
<tr>
<td>Em Run Time</td>
<td>34.0</td>
<td>20.6</td>
<td>38.7</td>
<td>11.6K</td>
<td></td>
</tr>
<tr>
<td>Compile Time</td>
<td>8.2</td>
<td>9.0</td>
<td>42.8</td>
<td>15.6K</td>
<td></td>
</tr>
<tr>
<td>Em EXE Size</td>
<td>9.1</td>
<td>2.5</td>
<td>39.4</td>
<td>21.2K</td>
<td></td>
</tr>
</tbody>
</table>

**Coprocessor Run Time Rankings**

<table>
<thead>
<tr>
<th>Rank</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Sines EXE Size</th>
<th>Empty EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zortech Turbo C</td>
<td>Zortech Turbo C</td>
<td>Microsoft</td>
<td>MIX</td>
<td>WATCOM</td>
</tr>
<tr>
<td>2</td>
<td>Comp In</td>
<td>Borland Turbo C</td>
<td>MIX</td>
<td>WATCOM</td>
<td>WATCOM</td>
</tr>
<tr>
<td>3</td>
<td>WATCOM</td>
<td>Borland C Ware</td>
<td>Manx</td>
<td>Manx</td>
<td>Manx</td>
</tr>
<tr>
<td>4</td>
<td>Microsoft</td>
<td>Microsoft</td>
<td>Zortech</td>
<td>MIX</td>
<td>Zortech</td>
</tr>
<tr>
<td>5</td>
<td>Metaware</td>
<td>Microsoft</td>
<td>C Ware</td>
<td>Microsoft</td>
<td>Zortech</td>
</tr>
</tbody>
</table>

**Coprocessor EXE Size Rankings**

<table>
<thead>
<tr>
<th>Rank</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Sines EXE Size</th>
<th>Empty EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manx</td>
<td>Zortech</td>
<td>Manx</td>
<td>Ecosoft</td>
<td>Ecosoft</td>
</tr>
<tr>
<td>2</td>
<td>Borland Turbo C</td>
<td>Borland Turbo C</td>
<td>WATCOM</td>
<td>MIX</td>
<td>WATCOM</td>
</tr>
<tr>
<td>3</td>
<td>WATCOM</td>
<td>Borland C Ware</td>
<td>Ecosoft</td>
<td>Manx</td>
<td>Manx</td>
</tr>
<tr>
<td>4</td>
<td>Zortech</td>
<td>C Ware</td>
<td>Zortech</td>
<td>MIX</td>
<td>Zortech</td>
</tr>
<tr>
<td>5</td>
<td>Microsoft</td>
<td>Microsoft</td>
<td>Zortech</td>
<td>C Ware</td>
<td>Microsoft</td>
</tr>
</tbody>
</table>

**Emulator Run Time Rankings**

<table>
<thead>
<tr>
<th>Rank</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Sines EXE Size</th>
<th>Empty EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Microsoft</td>
<td>Zortech</td>
<td>Manx</td>
<td>Ecosoft</td>
<td>Ecosoft</td>
</tr>
<tr>
<td>2</td>
<td>Borland Turbo C</td>
<td>Borland Turbo C</td>
<td>WATCOM</td>
<td>MIX</td>
<td>WATCOM</td>
</tr>
<tr>
<td>3</td>
<td>WATCOM</td>
<td>Borland C Ware</td>
<td>Manx</td>
<td>Manx</td>
<td>Manx</td>
</tr>
<tr>
<td>4</td>
<td>Zortech</td>
<td>C Ware</td>
<td>Zortech</td>
<td>MIX</td>
<td>Zortech</td>
</tr>
<tr>
<td>5</td>
<td>Microsoft</td>
<td>Microsoft</td>
<td>Zortech</td>
<td>C Ware</td>
<td>Microsoft</td>
</tr>
</tbody>
</table>

**Coprocessor Run Time Rankings**

<table>
<thead>
<tr>
<th>Rank</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Sines EXE Size</th>
<th>Empty EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zortech Turbo C</td>
<td>Zortech Turbo C</td>
<td>Microsoft</td>
<td>MIX</td>
<td>WATCOM</td>
</tr>
<tr>
<td>2</td>
<td>Comp In</td>
<td>Borland Turbo C</td>
<td>MIX</td>
<td>WATCOM</td>
<td>WATCOM</td>
</tr>
<tr>
<td>3</td>
<td>WATCOM</td>
<td>Borland C Ware</td>
<td>Manx</td>
<td>Manx</td>
<td>Manx</td>
</tr>
<tr>
<td>4</td>
<td>Microsoft</td>
<td>Microsoft</td>
<td>Zortech</td>
<td>MIX</td>
<td>Zortech</td>
</tr>
<tr>
<td>5</td>
<td>Metaware</td>
<td>Microsoft</td>
<td>C Ware</td>
<td>Microsoft</td>
<td>Zortech</td>
</tr>
</tbody>
</table>

**Coprocessor EXE Size Rankings**

<table>
<thead>
<tr>
<th>Rank</th>
<th>fxRef</th>
<th>Link Time</th>
<th>Em Run Time</th>
<th>Sines EXE Size</th>
<th>Empty EXE Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Manx</td>
<td>Zortech</td>
<td>Manx</td>
<td>Ecosoft</td>
<td>Ecosoft</td>
</tr>
<tr>
<td>2</td>
<td>Borland Turbo C</td>
<td>Borland Turbo C</td>
<td>WATCOM</td>
<td>MIX</td>
<td>WATCOM</td>
</tr>
<tr>
<td>3</td>
<td>WATCOM</td>
<td>Borland C Ware</td>
<td>Manx</td>
<td>Manx</td>
<td>Manx</td>
</tr>
<tr>
<td>4</td>
<td>Zortech</td>
<td>C Ware</td>
<td>Zortech</td>
<td>MIX</td>
<td>Zortech</td>
</tr>
<tr>
<td>5</td>
<td>Microsoft</td>
<td>Microsoft</td>
<td>Zortech</td>
<td>C Ware</td>
<td>Microsoft</td>
</tr>
</tbody>
</table>
This test generates some surprising results. The entire source for it is:

```
SMALL

First, it reads a text file of 1000 floating point numbers. It then sorts values and performs calculations. Finally, it writes the calculated values to a disk file. GRIND tests 1/0 speed, loop optimization, and floating-point library speed.

SINES is a new benchmark. (See Figure 5.) It calculates the sine of every angle between 0 and 360 degrees. SINES is unique in many ways. First, the only variables in it are doubles. Second, it doesn’t use a single library function. It primarily tests the speed of the floating-point code generated by a compiler. It’s interesting that some compilers which have very fast math libraries create very slow standalone floating-point code. This test generates some surprising numbers.

EMPTY is not a practical program. The entire source for it is: main(). It was designed to show the size of the compiler start-up code, and how quickly a compiler could compile nothing. Like SINES, this test generated some unusual results.

Ah, now on to the meat of this review! First, I’ll look at each compiler. Then, I’ll discuss overall compiler performance, along with some recommendations and suggestions.

**Borland Turbo C v2.0B**

In two years, Borland has come from nowhere to take a significant portion of the MS-DOS C compiler market. While it has had problems, version 2.0 is a solid, complete product.

Borland gives you a lot of bang for your buck.

For $150, the Standard package gives you an integrated compiler/editor, a command-line compiler, and several programming utilities.

For $250, the Professional package adds a command-line debugger and macro assembler. As a beginner, a hobbyist, or someone on a budget, the Standard package is all you need. You can always add the debugger/assembler package for $150.

Installation is simple and the package doesn’t take a lot of room. In fact, it’s quite easy to use on a dual-floppy laptop.

The environment is nice with everything accessed via pull-down menus or hot-keys. Its biggest drawback is lack of mouse support. You can easily customize the editor, a Wordstar clone (like all Borland editors).

**Debugger** The built-in debugger, while not as sophisticated as its stand-alone brother, certainly proves adequate for most debugging tasks.

The debugger can execute source-line by source-line. You can set breakpoints and breakpoints and you can change or examine variables. Once you solve a problem, you merely edit the source and recompile — all within the environment. It all works very easily and quickly.

**Help** The package includes a memory-resident help facility. It loads into memory and can be accessed from within any editor via a hot key. The help is somewhat context-sensitive.

**Errors** Turbo C does a better-than-average job of warning you about problems. For instance, it will yell if you use a variable before assigning it a value. It warns you about unused local variables, but not about unused formal parameters. You can control many of the warnings via compile options.

Documentation: The two paperback books are reasonably well-written and the tutorial section, friendly. You get lots of examples and lots of information about each function.

Unfortunately, Borland still puts function descriptions one-after-the-other, instead of beginning each on a new page. The habit of cross-referencing detailed function information to another, similar function can lead to lots of page flipping.

**Library** The Turbo C library is large and complete. It supports most of the ANSI standard features and has all the MS-DOS extensions we’ve come to expect in a PC C compiler.

Borland’s Graphic Interface (BGI) is one of the best graphics packages included with a compiler. It supports a wide variety of adapters and (at least to my mind-set) appears very well thought out and organized. I especially like the hardware-sensing option which lets you (with some care) write a very portable graphics program.

**Finally** Turbo C is the fastest compiler in the review, although several others come close. None of its execution time or size performances were particularly exciting. On the SINES test, its emulator came out third from last.

**Ware DeSmet C v3.1e**

This compiler has changed significantly since last year. Previously, DeSmet followed the K&R standard, ignoring many of the new ANSI features. Now, it supports the most commonly used ANSI features, including function prototypes.

This is the smallest compiler in the group; a minimum configuration will fit on a 360K floppy disk with some room left over! The complete package requires only 600K. While there is no installation program, the manual clearly describes the copying process.

**Package** You get everything you need in the basic package — an editor, a linker, a compiler, an assembler, and a librarian. (The optimizer is extra.) The basic package only compiles to the small model. The large model option costs extra.

Environment: You can run the compiler from the editor and the editor will point out compile errors in the source. You can reconfigure the editor to work with non-IBM compatible MS-DOS machines.

The assembler is not MASM compatible, but should be useful for creating small routines and functions.

**Debugger** You get a source-level symbolic debugger. Although not as fancy or sophisticated as some others, it does everything you need to track most bugs.

Documentation: The manual comes in a single three-ring binder. With it you get a sheath of update pages, which you must integrate. The manual has one big deficiency — no index.

Finally: While this compiler contains some of the ANSI features, many are still missing. For instance, it does not include any of the standard time functions so I had to write a special version of the DHRYSTONE 2 benchmark. On the plus side, DeSmet C has graphics and direct-video-display functions.

This compiler runs fast and produces very small .EXE files. The performance of resulting programs, however, ranged from average to below average.

**Computer Innovations C86Plus v1.20D**

C86Plus arrives with two three-ring binders. I wish I could recommend this compiler. It has the most enjoyable documentation to read, filled with anecdotes and humor. Unfortunately, compiler performance is the worst of the bunch.

You can tell that this compiler is a little different by the names they give the tools. For instance, they’ve named their linker program “Carol.” Their object-module librarian is “Marion.” The package has all the important tools except an editor and a debugger (an empty chapter in the manual waits for a future debugger).

**Installation** You get an easy to use installation program which lets you set the locations of the compiler and tools.
## Figure 2 — Compiler Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Borland Turbo C v. 2.0</th>
<th>Comp. Inn. C-878plus v. 1.20D</th>
<th>C-Ware DeSmet v. 3.1e</th>
<th>EcoSoft Eco-C88 v. 4.0.16</th>
<th>Lattice Manx v. 3.31</th>
<th>Manx Aztec C v. 1.4C</th>
<th>Metasware High-C v. 1.4</th>
<th>Microsoft Quick C v. 5.1</th>
<th>Microsoft Power C v. 1.01</th>
<th>MIX v. 1.2.0</th>
<th>WATCOM C v. 6.5</th>
<th>Zortech C v. 1.07</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Utilities</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>environment(1)</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>linker</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>librarian</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>debugger</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>(3)</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>editor</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>(3)</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>assembler</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>make</td>
<td>Y</td>
<td>Y</td>
<td>(6)</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>grep</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>profiler</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><strong>Models (4)</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tiny (COM)</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Small</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Medium</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Compact</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Large</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Huge</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Mixed</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Code Output</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>.OBJ files</td>
<td>Y</td>
<td>Y</td>
<td>(5)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>OS/2 support</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>80x87 support</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>8087 sensing</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>186 support</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>286 support(2)</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>386 support(2)</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>ROMable</td>
<td>N</td>
<td>(3)</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Assembler</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Source</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>library</td>
<td>$150</td>
<td>$89</td>
<td>$50</td>
<td>$500</td>
<td>Y</td>
<td>$2,000</td>
<td>$150</td>
<td>$150</td>
<td>$10</td>
<td>$225</td>
<td>$50</td>
<td></td>
</tr>
<tr>
<td>start-up</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>tutorial</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>lang. ref.</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><strong>Support</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BBS</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Commercial</td>
<td>CIS</td>
<td>BIX</td>
<td>N</td>
<td>N</td>
<td>BIX</td>
<td>N</td>
<td>CIS</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td><strong>Other</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Req's HD</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Notes:**
(1) Borland, Microsoft, and WATCOM provide integrated environments with the compiler, editor, and debugger built-in. The other products provide an editor which can also run the compiler (and possibly the debugger) while editing a program.

(2) Most of the compilers reviewed support protected-mode programming - only support for the special instructions available in real mode on these chips.

(3) Available at extra cost.

(4) Models
- Tiny - code and data both in 64K (.COM model)
- Small - small (up to 64K) code, small data
- Medium - large (up to 1MB) code, small data
- Compact - small code, large data
- Large - large code, large data
- Huge - large code, large data, individual data items can be 64k
- Mixed - near and far items can be declared in one program unit

(5) Producing .OBJ files requires the use of a special utility.

(6) The compiler has limited, built-in make-like facilities

(7) 80386 protected mode support is available in a separate product

(8) The large Microsoft Optimizing C compiler package includes Quick C. For example, the editor included with Microsoft C is actually the one from the Quick C environment. Also, although MS C cannot be placed on floppies, Quick C can

(9) WATCOM C comes with Express C, a integrated editor/compiler designed for the educational market.
C86Plus takes up a significant amount of disk space, making it difficult to use on a floppy-based system.

Library: C86Plus has a large, robust library. The language conforms to most of the early drafts of the ANSI standard. And you get typical MS-DOS extensions such as near and far pointers and direct access to video memory. The function, zork(), returns a weird string (apparently) significant to the popular computer adventure game.

Finally: Running C86Plus is simple; it supports the Microsoft C compiler switches. The product seems to be very stable.

Performance left much to be desired. Compiles were so slow I occasionally wondered if the program had locked up. The linker was no better. Programs ran in average times although they did very well on the coprocessor and FXREF execution times. The C86Plus floating-point emulator is very slow, often twice as slow as any other emulator.

Ecosoft Eco-C88 v4.16
Ecosoft has produced C compilers since the days of CP/M. Their MS-DOS offering was one of the first compilers to implement some of the ANSI features such as function prototypes and void. There have been few outward changes in this compiler since last year; most of the changes have been minor bug-fixes.

Documentation: This package's biggest weakness is the manual. Contained in a three-ring binder, it's haphazardly organized and doesn't contain much of the information I looked for. For instance, the package will link several different libraries with Ecosoft programmers, but the manual provides little explanation of when to use the libraries.

Installation: The installation program lets you select language components and their location. It also lets you place the package on floppies.

This compiler uses more environment variables than any other. The environment variables show the locations and names of libraries, compiler components, and header files. One environment variable, EPICK, sets the default value for the "picky" flag (described below).

Package: Eco-C88 includes the compiler, a simple source-level debugger, and an editor. An object-module librarian is available separately. The package does not include a linker, which can be a problem if your version of MS-DOS doesn't include Microsoft's LINK. The debugger is nothing fancy — it doesn't have the windowed interfaces of some of the newer debuggers. Still, it does what it needs to — it tracks down bugs.

The compiler contains several interesting features. It can use "make" logic on the files specified, so that it will recompile any source files newer than their object modules. This was provided in lieu of a standalone make program.

Another compiler switch of interest, the so-called "picky" flag, behaves like the warning level switch found in other compilers. Eco-C88 includes several lint-like capabilities, and the picky flag determines how picky the compiler is about your code.

On the upside, Eco-C88 has a feature-rich library. It misses some of the ANSI functions but contains other interesting routines, including functions to manipulate binary trees and video displays.

It doesn't have a graphics library and some "standard" extended functions, such as the spawn() family, are conspicuously absent. Only Lattice provides less support for the proposed ANSI standard than Ecosoft.

Performance-wise, Eco-C88 runs about average. Though it generated smaller than average code, it took about an average amount of time to do it. The programs were reasonably fast, and Eco-C88 did come in fourth in the run time of the SINES benchmark (using the emulator.)

Lattice C v3.31
Lattice used to be king of the C compilers, but Microsoft passed them several years ago and Lattice never recovered. At the rate they're going, they never will.

Documentation: The manuals have seen the biggest improvement from last year. They're attractively typeset, well organized, and bound in two three-ring binders.

Installation: Installation is simple. The installation program lets you select which components to load to disk. The choices lie in a documented configuration file.

Package: This is a complete package with several tools, including the editor, debugger, and some unusual items like an OS/2 binder. Lattice is one of only two C compilers supporting OS/2 and Microsoft Windows (Microsoft, of course, being the other.) Lattice does not provide a linker. You can compile programs within the editor which will trap and track errors.

Debugger: The debugger is very primitive, basically a souped-up version of DEBUG. Lattice promised a CodeView-like symbolic debugger at COMDEX in 1987, but no one's seen it yet.

Lattice C probably supports more compiler switches than any other compiler in this review. Some of them are unique. For example, the -R switch automatically inserts the object modules into a library.

Finally: On the box they state that Lattice is an "ANSI" compiler. Unfortunately, nothing could be further from the truth. Of all the compilers reviewed, Lattice is the farthest from compliance. It doesn't even support simple features, such as the new Pascal-like function headers. And it lacks many of the ANSI header files.

Lattice C's performance is nothing to brag about. Its only claim to fame in the benchmark tables was the performance of its floating-point emulator on the SINES run-time benchmark. Otherwise, it compiled slowly and produced average-sized .EXE files.

Manx Aztec C v4.10c
Manx puts out compilers for everything from Apple IIs to 68000-based machines. Its 80x86 compiler for PC compatibles comes in several flavors, ranging in price from $200 to $500. As the price goes up, so does capability. This review looked at the Commercial Package, the most expensive (and extensive) version available. This compiler has changed very little from last year.

Installation: Installation directions are sparse and there's no installation program, so you'll find it interesting figuring out which directories to create and which files to load. There's no excuse for not having at least a section discussing the recommended way to install the product.

Documentation: The manual still has two of the problems it had last year — a too small binder and poor organization. Manx adds additional documentation for each release level. Unfortunately, they don't integrate this information into the manual. The inserts have grown to the point that the three-ring binder will not completely close. It's often difficult to find information when it's divided among two updates and a two-part manual.

Package: This is one of the most complete packages I've seen. It includes every important utility. Though the editor won't call the compiler, the symbolic debugger, SDB, is nearly as capable as CodeView (or the Turbo Debugger). Unlike CodeView, it allows an un-
C CODE FOR THE PC
source code, of course

MS-DOS File Compatibility Package (create, read, & write MS-DOS files; run non-MS-DOS computers) .................................................. $500
Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation) ................................................................. $400
COE Security System (UNIX; 3D, 4D plus windows) .......................................................................................................................... $325
GraphiC 4.1 (high-resolution, DISPLA-style scientific plots in color & hardcopy) ................................................................. $325
PC Curses (Aspen, Software, System V compatible, extensive documentation) ................................................................. $250
Greenleaf Business Mathlib (exact decimal math, formatting, depreciation, interest, cash flow, statistics) .......................... $225
NEW! Greenleaf Functions (296 useful C functions, all interrupt mode, macros, error checking) .......................................................... $175
NEW! TurboFX (TRIP certified; HP, PS; dot drivers; CM fonts; LaTeX) ............................................................................. $170
NEW! Asheville Kit (by John Zarrella; includes listing generator & loader; requires signed license agreement) ....................... $175
NEW! Communications Utility Library (all popular line interfaces) ................................................................................... $125
NEW! Turbo Graphics Library (all popular line interfaces, full line removal) ................................................................. $135
TurboGeometry (library of routines for computational geometry) ................................................................................ $125
NEW! LOC (multiple-line prompting, graphics, multiple windows) ................................................................. $115
Minix Operating System (UNIX-like operating system, includes manual) ................................................................. $105
PC/IP (CMU/MIT TCP/IP implementation for PC) ........................................................................................................... $100
B (for C package; has module utilities by Softsource) ........................................................................................................... $100
NEW! PC Tele Operating System (TeleKernel, TeleWindows, TeleFile, & TeleTerm by Ken Berry) .......................................................... $100
The Profiler (program execution profile tool) ........................................................................................................................ $90
QCB (small model, no longs, floats or bit fields, 80+ function library) ........................................................................... $90
Wendin Operating System Construction Kit or FCNX, FCVMS O/S Shells ......................................................................... $80
C Windows Toolbox (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules) ................................................................. $80
JA's Library (includes file transfer and many utilities) ........................................................................................................... $80
Polyglot Lisp-to-C Translator (includes Lisp interpreter, Prolog, and simple calculus prover) ................................................................. $80
NEW! MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores) ................................................................. $80
XT BIOS Kit (for basic XT/PS with this complete set of basic XT utility and PCL virtual printer functions for XT) ................................................ $75
TE Editor Developer's Kit (full screen editor, undo command, multiple windows) ............................................................ $75
Professional C Windows (lean & mean window and keyboard handler) ............................................................................. $75
NEW! Heap Expander (use LIM-standard expanded memory as an extension of the heap) ................................................................. $75
lp (flexible printer driver; most popular printers supported) ................................................................................................ $65
Quincy (interactive C interpreter) ................................................................................................................................. $60
Plant (parse tree management) ........................................................................................................................................... $60
OBJASM (convert obj files to .asm files; output is MASM compatible) ................................................................................ $50
NEW! ICON-Tools (full-featured icon display and editing system) .................................................................................. $60
NEW! Polyglot TSR Package (includes reminder, bookmark, cache manager, & speech generator) .................................................. $50
HELP! (pop-up help system builder) ........................................................................................................................................... $50
Multi-User BBS (chat, mail, menus, sxsp, displays; uses Galacticomm modem card) ................................................................. $50
MacFree, all languages, build-in rules) ........................................................................................................................................... $50
Coder's Prolog (inference engine for use with C programs) .................................................................................................. $45
Virtual Memory System (least recently used swapping) ........................................................................................................ $40
C-Notes (pop-up help for C programmers ... add your own notes) ............................................................................. $40
Heap I/O (treat all or part of a disk file as heap storage) ........................................................................................................ $40
Biggergen's System Tools (multitasking window manager kit) ........................................................................................ $40
NEW! CP (for Berkeley curses packages) ................................................................................................................................ $40
TINX (Comer's XINU operating system for PC) .................................................................................................................. $35
CLIPS (rule-based expert system generator, Version 4.2) .......................................................................................... $35
NEW! Polyscript (change disk size on the fly; includes utilities) ........................................................................................ $30
SP (spelling dictionary with extensive documentation) ........................................................................................................... $30
Clip (4 function compression & expansion programs) ........................................................................................................ $30
NEW! Install (automatic installation program; user-selected partial installation; CRC checking) ................................................................. $25
Personal Compiler & Interpreter or Pascal-to-C Translator (P-codes, standard Pascal) ................................................................. $25
ICO (C language, Version 7.5) ........................................................................................................................................ $25
FLEX (fast lexical analyzer generator; new, improved LEX) ........................................................................................................... $25
LEX (lexical analyzer generator; an oldie but a goodie) ........................................................................................................ $25
Builder & PREP (YACC workspace, parser generator & attribute grammar preprocessor) ........................................................ $25
AutoTrace (program tracer and memory tracer) ................................................................................................................ $25
Data Handling Utilities in C (data entry, validation & display; specify Turbo C or Microsoft) ................................................................. $25
NEW! Array handling of arrays) ........................................................................................................................................ $25
ANSI Forms (forms manager based on ANSI codes) ........................................................................................................... $20
C Compiler torture Test (checks a C compiler against K & R) ........................................................................................ $20
BenchMark (PC hardware, and Unix systems) .................................................................................................................. $20
A86 (68000 cross-assembler) ................................................................................................................................................ $20
List-Pac (C functions for lists, stacks, and queues) ................................................................................................................ $20
XLF (Macro Processor (general purpose text translator) ........................................................................................................ $20
CREATIVITY (Eliza-based notetaker) ........................................................................................................................................ $15
Data
DNA Sequences (GenBank 55.0 including fast similarity search program) ........................................................................... $150
Protein Sequences (5,415 sequences, 1,302,966 residues, with similarity search program) ................................................................. $70
NEW! SIC Codes (each SIC code with the name of the industry to which it applies) ........................................................................... $70
Values of $600s, in alphabetical order, no duplicates) ........................................................................................................ $70
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points) ................................................................. $35
The World Digitized (100,000 longitude/latitude of world country boundaries) ........................................................................ $30
KSS (1,300 characters in 139 mixed fonts: specify IX bitmap format) ................................................................................ $30
USNO Flpps (almanac boxed, high precision moon, sun, planet & star position) ................................................................................. $30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts) ........................................................................................................... $15
U. S. Map (15,701 points of state boundaries) .................................................................................................................. $15

The Austin Code Works
Voice: (512) 228-0785
11100 Leafwood Lane acwinfo@unet.us.net
Austin, Texas 78750-3409 USA Reader Service Number 4
BBS: (512) 228-8831
PAX: (512) 228-1312

MICRO CORNUCOPIA, #46, Mar-Apr, 1989 45
limited number of breakpoints and can use keyboard macros. SDB does not support a mouse.

The Commercial Package has complete facilities for the creation of ROMable code. As the benchmark table shows, Aztec C produces very small executable files.

Finally: I found Aztec C easy to work with. The compiler has a simple syntax and only a few switches. Environment variables locate headers and libraries. My only problem came when trying to change the stack size. The manual documents three variables which must be set when there are actually four (_STKRED must be set to the minimum clearance between the stack and heap.) Again, the documentation is the weakest part of this package.

Aztec C did very well in the benchmarks. The compiler runs fast and generates small programs. While not amazing, the performance of the generated programs is at least average. Aztec C came in second in the execution speed of DHRystone 2.

Metaware High-C v1.4

This product hasn't changed from last year. High-C's ads proudly proclaim that it was used to create several popular products (including dBASE III/IV and Q&A). However, it's quickly losing ground in that arena to Microsoft, WATCOM, and others.

Installation: Metaware's installation program copies the compiler components to any directory but there's no way you can run this package from a floppy. The compiler takes up over 600K!

Documentation: The manual barely fits into its three-ring binder. Divided into three sections, each with its own index, this is one of the better manuals in the group. It contains complete, clearly written information.

Only High-C supports "profiles," files which contain compiler directives and instructions. You create specific profiles for specific projects, thereby customizing the compiler's behavior.

Library: Metaware doesn't provide many utilities. It has no librarian or linker. The only utilities included are a GREP and a source code cross-referencer. While many experienced programmers don't use built-in utilities, beginners usually find them very useful.

High-C implements many common ANSI features, but it has a long way to go before it's in full compliance. The library doesn't include any extras. This compiler has one of the most boring libraries in the group with operating system access functions, but no graphics library.

Finally: High-C's performance is mediocre. Its compile times beat only C86Plus and it produces large executables. While High-C advertises as an optimizing compiler, it doesn't match the performance levels of some non-optimizing products. High-C turned in its best performances on the coprocessor run-time tests for GRIND and SINES, where it came in fifth and second, respectively.

Microsoft C v5.10

This is the industry standard. As a leader, it has its critics (who have some valid points). However, an overall view shows that Microsoft C has not become the standard merely because of the reputation of its parent.

Installation: Microsoft has continued to improve its installation program. It's very flexible, giving you control over what's put where. You can even change libraries without reinstalling the entire package.

Documentation: The documentation package includes three two inch thick three-ring binders containing six manuals, along with a paperback reference manual for QuickC. Microsoft's documentation is very thorough and liberally indexed.

Unfortunately they haven't integrated the upgrade for version 5.10 so you have to look in at least two places to be sure you have the correct information. I don't understand why companies with looseleaf binders can't integrate their updates.

Package: Microsoft includes a cornucopia (appropriate for Micro C) of utilities and tools with their C compiler. The most extensive of these is QuickC, an environment-based compiler sold separately (see the separate discussion of QuickC below.) Recent additions include a powerful editor and a grep utility.

Debugger: CodeView is Microsoft's windowing, source-level debugger. It looks a bit long in the tooth when compared to products like Borland's Turbo Debugger, but is still one of the best source-level debuggers available. CodeView lets you track and view data throughout the execution of a program. It does line-by-line and animated (slow motion) execution. You can store debugger commands in a script file.

One CodeView feature is sadly lacking in other debuggers — mouse support. I still think of CodeView as a fine product, in spite of its reliance on the archaic DEBUG and SYMDEB command set.

Editor: The Microsoft Editor (ME) would be a much better product with usable documentation (it's better for quick reference). However, if you spend the time, you'll find ME extraordinarily powerful. In many ways, it resembles Brief.

It does, however, have some unique capabilities. Instead of a built-in programming language, ME lets you write extensions in either C or Macro Assembler. You then compile them and they're linked in at run-time. (They're lighting fast.)

Using the Microsoft compiler is easy. A large number of switches control everything from code generation to the listing format. No other compiler fully supports OS/2, including multi-thread programs and dynamic-link libraries.

I've heard through the C community that Microsoft C 5.10 is buggy. It gained this reputation with 5.00. Microsoft rushed 5.00 to market, though that package had a significant number of bugs.

When 5.10 came out, it retained the stigma of its predecessor. I've had no problems with the compiler, although there still could be some.

Of course, misunderstandings about the optimizer could also generate bug reports.

For instance, the /Ob (included in the catchall /Ox switch) optimization switch tells the compiler to ignore the possibility of aliases in the program code. You get an alias when a memory location gets modified by more than one variable. (Often occurs when using pointers heavily.)

Aliasing makes C harder to optimize than other languages. The programmer needs to be aware of what aliasing is, to be sure it's safe to use the /Ob switch.

Finally: Microsoft C performs very well. It compiles slowly, but that shouldn't be a problem with QuickC included in the package. I use QuickC for most of my preliminary compiles, unless I'm using some features available only in its big brother.

Microsoft C generates fast code — it won the DHRystone 2 benchmark by a considerable margin, though it produces larger than average .EXE files. Probably its biggest failure was the SINES benchmark, where its emulator performance was very poor. It's only fair to note that Microsoft offers an alternative math library — less accurate.
than the emulator library, but considerably faster.

**Microsoft QuickC 1.01**

Microsoft's QuickC integrates editor, compiler and debugger functions. By the time this review reaches the stands, version 2.00 will be out. Version 2 is a complete rewrite of version 1.01, with many new features. Therefore, I'll keep this discussion short.

Quick C comes with a set of paperback manuals. Like the "big" Microsoft compiler, Quick C installs easily. It doesn't work well from floppy disks — it simply requires too much disk swapping, even with the special overlay provided.

Quick C offers most of the features of the big Microsoft compiler, though it doesn't support most of the optimizations. Microsoft produced QuickC for folks learning the C language. They figure those of you interested in an optimizing compiler will buy the larger package.

The package also includes a command-line version of the compiler — merely a driver which runs the integrated environment. It also includes a complete set of command utilities, along with a make and a librarian.

Finally: Quick C really performs. It compiles very quickly (as its name would imply), and produces very fast programs. QuickC outperforms Borland's Turbo C (its most common rival) in most of the .EXE run-time benchmarks. Its worst performance came on the emulator run-time test for SINES, where it ranked second slowest. (Microsoft doesn't offer an alternative math library for QuickC.)

**MIX Power C v1.2.0**

You won't find a less expensive compiler available, for any language. At $20 (basic package), Power C costs less than most computer books.

Installation: Nothing fancy here, you just copy the compiler to floppy disks or your hard drive. Everything about this package is budget-level, except for the manual.

Documentation: MIX worked hard to create a manual useful to the novice C programmer. The tutorial contains one of the best explanations of pointers I've seen.

The manual does have problems, though. They've separated the information on compiler switches from the instructions on running the compiler. The manual includes compile options that don't work yet (the README file notes...
this). The error reference section merely lists the error messages, there's no explanation of them.

Package: For $20, you get the compiler and linker — that's it. MIX's optimizer is part of the compiler's code generator.

The compiler does support a wide variety of floating point options but has only one memory model — medium. This is why I did Power C's benchmarks with the medium model while I used the small models for other compilers.

Additional packages are available. The CTrace debugger costs $19.95, as does a database toolbox and a BCD math package. The library source code sells for $10 and contains a simple 8088 assembler as a bonus.

As with all advertising claims, the comparative performance chart in the Power C ads and on the back of its manual are a bit optimistic. It compiles fast, but nowhere near as fast as Borland and Zortech. It produces small, swift programs, but not usually the fastest. The most spectacular performance by Power C was on the emulator run time test for SINES. Only WATCOM beats Power C's 37.5 seconds on the emulator run time test for SINES.

WATCOM C v6.5

WATCOM gives Microsoft a run for its money. A spinoff from the University of Waterloo in Canada, the company is well-known for its mainframe FORTRAN compilers (WATFOR and WATFIV). When they entered the PC market last year with version 6.0 of their C compiler, they made a splash that is still sending ripples through the MS-DOS C community.

Installation: WATCOM includes a simple program for installing the compiler and its components onto a hard drive. (Don't even bother trying it on floppies.) As with most compilers, it uses environment variables to direct the compiler and utilities to needed files.

Documentation: They've divided documentation into five spiral-bound books. You also get a stack of quick reference guides, one for each major program. All the manuals have complete, and logical, indexes.

Package: This package has everything, though the editor isn't particularly exciting. It emulates the old IBM mainframe editors by being slow and clumsy. But at least it exists.

Debugger: The debugger is not as easy to use as its competitors, but it has plenty of features. For instance, I

```c
add_tree(tok, line);
}

void add_tree(char *tok, line_no line)
{
    token *temp_tok, *new_tok;
    location *temp_loc;
    short comp;
    if (root->text == NULL)
        
        if (NULL == (NOLL == (root->text =
                        (char *)malloc((unsigned)strlen(tok)+1))))
            error(1);
        strcpy(root->text, tok);
        if (NULL == (NOLL == (root->loc =
                        (location *)malloc(sizeof( location)))))
            error(2);
        root->loc->line = line;
        root->loc->next = NULL;
        root->last = root->loc;
        return;
    }
    temp_tok = find_tree(tok);
    if (temp_tok = find_tree(tok))
        /* comp is true (non-zero) if they don't match */
        
        if (temp_tok = find_tree(tok))
            /* comp is false (0), the tokens match */
            
            if (temp_tok = find_tree(tok))
                return node;
            else
                node = root;
            while (1)
            
                if (0 == (comp = strcmp(tok->text))")
                    return node;
                if (comp < 0)
                    if (node->less == NULL)
                        return node;
                    else
                        node = node->less;
                else
                    if (node->more == NULL)
                        return node;
                    else
                        node = node->more;
            }
    }

token *find_tree(char *tok)
{
    short comp;
    token *node;
    node = root;
    while (1)
        
            if (0 == (comp = strcmp(tok,node->text))")
                return node;
            if (comp < 0)
                if (node->less == NULL)
                    return node;
                else
                    node = node->less;
            else
                if (node->more == NULL)
                    return node;
                else
                    node = node->more;
        }

    void show_tree(node)
    
        token *node;
        
            location *loc;
            short pos;
            if (NULL == node) return;
            show_tree(node->less);
```
**Figure 4 — GRIND.C**

/*
Program: Grind
Version: 1.11 Date: 26-Oct-1988 Language: ANSI C

Tests all aspects of a C compiler's functions, including disk i/o, screen
i/o, floating point, recursion, prototyping, and memory allocation. It
should be a large enough program to test the advanced optimizers in some
compilers.
Developed by Scott Robert Ladd. This program is public domain.
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define MAXFLOATS 1000

struct tabent
{
    double val, vsum, vsqr, vcalc;
};

struct tabent table[MAXFLOATS];

char *errmsg[] = {
    "GRIND.TBL cannot be created",
    "GRIND.TBL cannot be closed properly",
    "GRIND.IN cannot be found",
    "GRIND.IN has been truncated",
    "GRIND.IN cannot be closed properly"
};

/* function prototypes */

short main(void);
void readfloats(void);
void sortfloats(void);
void quicksort(struct tabent *, short, short);
void maketable(void);
void writetable(void);
void error(short);

short main(void)
{
    puts("Grind (C) v1.10 -- A Benchmark Program\n");
    readfloats();
    sortfloats();
    quicksort(table, 0, MAXFLOATS - 1);
    maketable();
    writetable();
    puts("End run GRIND!!\n");
    return(0);
}

void readfloats()
{
    register short i;
    char buf[12];
    FILE *fltsin;
    fltsin = fopen("GRIND.IN","r");
    if (fltsin == NULL) { continued next page
the help hot key will bring up a subject related to the item at the current cursor location. This way you get help while working in your favorite editor.

Zortech C includes several utilities. Many of these, such as the timer and grep programs, come with source code.

Library: The library is quite complete. It includes several standard functions, such as strtok() and sleep(), not mentioned in the manual. (I also noticed they'd left out or misplaced examples in the function reference section.) Zortech has supplied a graphics library (as have other vendors) and their Flash Graphics are very fast.

Unlike Datalight, Zortech doesn’t include the source to the library functions. Source costs you an extra $50.

Finally: On the benchmark tests, Zortech C did amazingly well. It compiled very rapidly and produced small .EXE files. Zortech showed up in almost all the “top five” charts in the benchmark table. Its file I/O and floating-point emulator performed admirably. Were I to rank compilers solely by performance, Zortech would lead by a wide margin.

Considerations

If you expect me to tell you which compiler to purchase, you’re going to be disappointed. The compiler you need depends upon your resources and what you’re doing. Remember that these are only benchmarks, not real life, and a group of comparisons won’t be as thorough as individual reviews.

Microsoft C has slipped a bit (in comparison to the other products) since last year, but remains an excellent value. You can pick up this package for under $300 from many mail order houses. (Watch for object-oriented facilities. They've scheduled a new release for mid-1989.)

WATCOM C has taken the MS-DOS C world by storm. If you work with floating point, there is no other choice. This compiler's biggest liability is compile time. Otherwise, it performs amazingly well. Watch for version 7.0, due out in early 1989.

Borland’s Turbo C is blazingly fast. It doesn’t produce the fastest executables, but it’s still faster than average. Borland’s produced an amazing debugger and an excellent graphics library.

Manx Aztec C remains a good value. It comes with a great debugger and is optimized for producing ROMable code. Anyone who works with embedded systems should have Aztec
Figure 5 — SINES.C

/*
Program: Sines
Version: 1.00
Date: 29-Nov-1988
Language: ANSI C

Computes all of the sines of the angles between 0 and 360 degrees. Developed by Scott Robert Ladd. This program is public domain.
*/
#define pi2rad 57.29577951

/* prototypes */
void main(void);
double fact(double);
double power(double, double);

void main()
{
    double angle, radians, sine, worksine, temp, k;
    for (angle = 0.0; angle <= 360.0; angle += 1.0)
    {
        radians = angle / pi2rad;
        k = 0.0;
        worksine = 0.0;
        do
        {
            sine = worksine;
            temp = (2.0 * k) + 1.0;
            worksine += (power(-1.0, k) / fact(temp)) * power(radians, temp);
            k += 1.0;
        } while (sine != worksine);
    }
}

/* Note: this function is designed for speed; it ONLY works when n is integral */
double fact(double n)
{
    double res;
    res = 1.0;
    while (n > 0.0)
    {
        res *= n;
        n -= 1.0;
    }
    return res;
}

/* Note: this function is designed for speed; it ONLY works when p is integral */
double power(double n, double p)
{
    double res;
    res = 1.0;
    while (p > 0.0)
    {
        res *= n;
        p -= 1.0;
    }
    return res;
}

/***/

C. This compiler would also be a good choice in environments where code must run on different architectures.

Power C is an excellent performer at an unbelievable price. MIX has finally worked the bugs out of their product, making it one of the best values going. I recommend this compiler to students and beginners. For number crunchers on a budget, it performs nearly as well as WATCOM C.

However, with all those comments aside, one compiler stands out in my mind as an outstanding value. Zortech C has continued the reputation it earned as Datelight C, by providing excellent code quality and compile speed for a small amount of money.

Zortech appeared in the benchmark summary charts 15 out of 18 times; no other compiler matches this performance. Many programmers interested in the quality of the code they put out have chosen this compiler. Those programmers looking to the future may want to purchase the Zortech C++ package which includes this C compiler. While you use their powerful compiler, you can also learn about objects.

The Future of C

By this time next year, some vendors will have upgraded their compilers from C to C++ (or some other object-oriented variant of C). Zortech's C++ product is already on the market. Like the move to structured programming in the seventies, object-oriented programming will forever change how we write programs. In the issues ahead, I'll look at some of the reasons C is the basis for the languages of the 1990s.

Next Issue

In the May/June 1989 issue, I'll talk about the use of some of the tools and utilities commonly associated with C. I'll also discuss some of the differences between the ANSI and K&R standards. Some are more significant than you might think.

Several of the manufacturers are readying major upgrades as I write this (the last week of November). So I'll talk about these.

Micro Assembly

On Memorial Day weekend 1989 (May 27-29), we're holding a Micro Assembly here in Denver, Colorado, at the Clarion Hotel (Airport). Registration costs $35 before the convention, and $50 at the door. Get in touch if you're interested.

MICRO CORNUCOPIA, #46, Mar-Apr, 1989 51
Wow, Laine's body may be in Minnesota, but his mind certainly hasn't cleared customs. You'll hear more as he unpacks. Meanwhile, on to the body of his column.

S
o, I'm fresh off the boat, see? Well, off the plane anyway — you know, newly arrived in the country, ready to tackle some hot high tech. I've been dreaming of this for months. Yeah, get back to good ol' PC Tech and maybe do something new and exciting. Like write a multitasking kernel for a 34010 ... a video driver for a high res board ... a ROM BIOS for a 386. Maybe implement LIM 4.0 for a new version of the 16 Megger.

Anyway, I'm walking into the office, right? First stop after the airport and a night on the floor at La Guardia. Slept across the carpet from an old guy named "Jackson." He had two sweet rolls and a can of Blitz for breakfast and said someone was coming to pick him up. First chance to relax since sharing overnight baggage watch duty with two Iranian senior citizens who came in on the same flight. First real moment of non-anxiety since being approached by a street dude asking me, "What kinda stuff you got there, man? Got enough for me?" (This happened while I was taking out my contact lenses, sitting in the lounge chair right across from the Piedmont check-in counter.)

Where was I? Oh, yeah. So, I'm walking into the office, okay? Well, "Earl," I ask, "What am I gonna work on? Whaddya got planned for me?"

So Earl says, "Well, I'm finishing the software for this new 34010 mono board with Group 4 FAX image decompression built in and ..." ("Oh yeah!" I'm thinking. "This is gonna be hot!") "... and I'm still using the 9 x 15 font from the 736 x 1024 CPT monitor, but this new 2048 x 1536 Moniterm needs something larger, like about 12 x 23."

Whaatttt?? I just sold my X24/Rabbit Portable for plane fare! I've left my friends and life of sun and leisure! I've come 9,000 miles to click dots on and off with a mouse?!!??

"I don't want you to waste your time clicking dots on and off with a mouse, though." Earl continues. "What I really want is a program that converts between Windows and Gem raster fonts, and our own internal 34010 format:"

"Oh."

So, armed with the Windows Developer's Kit, the Gem Programmer's Toolkit, Zortech C++, and my favorite coffee cup (you must see it to appreciate it), I set off on my week long quest to understand binary images, discover new storage methods, to boldly figure out what the hell is going on.

So Where Was I?

Oh yeah... fonts. That's it! Well, see, it turns out that there are hundreds of fonts in the public domain, if you just know where to find them. Don't ask me; I don't know where to find them.

Anyway, knowing that there's lots of free fonts out there (seriously!) should give you the desire to read on. Please?

Here at PC Tech, we got several fonts included with "GFX Fonts," a font display library used with the GFX Graphics Library from a company called C Source. I can't tell you much about C Source or GFX Fonts, since I only wanted to translate their fonts into another format.

All I can tell you is that their font editor failed to run on a Hercules card (although it ran fine on an EGA). And their font files contained deviations from the Gem standard they were supposed to follow. But that's just first impressions, so ignore it, huh? Besides, it includes all the source code, so you can change it if you want.

Anyway, there are all kinds of fonts around. Balloon fonts, cartoon fonts, Olde English Fonts. We even put a Cyrillic font into ROM on one of the 34010 boards just for fun (we're trying to land a contract with the KGB, but don't tell anyone).

If people just had enough information to use these fonts ... (Hah! I knew I'd squeeze a topic out of this somewhere!).
Abstract
The programs I ended up with are too special-purpose (and too ugly) to put on public display in their entirety. But documentation and explanation of the font files themselves are worth talking about. So I’ll explain the format of Windows raster font files and Gem font files, and give you a couple programming examples so you can use them.

Then, as an example of “something completely different,” I will briefly describe PC Tech’s own “PCT” font file, newly created by a committee of 1.5. (I asked Earl’s opinion now and then, but mostly ignored what he said.)

First I’ll discuss a few different possible font formats.

How To Represent A Character
When asked to come up with a data structure for a graphics representation of a character set (a font), most people will whack out the definition in Figure 1. This is fine, if your characters are all equal width, and this width is 8 or less. What if your font were 9 bits wide? Do you then switch from chars to ints? You could, if you wanted to waste nearly 50% of your storage. This would only mean a loss of 256*Height bytes (3840 if your characters are 15 high). That’s acceptable if you only use one font, but some graphics applications use more than 10 fonts at a time. Quite a few of those fonts are much larger than the font we’re talking about.

Some other time I’ll talk about the pros and cons of different font storage formats. For now let’s just look at some (but I’m sure not all) of the different ways people have come up with to store font bitmaps.

Figures 2 through 5 give a graphic idea of the font formats, using a two character 6 x 6 font as an example.

Unpacked Linear
This is the format of the definition in Figure 1 (see Figure 2). The system stores all the scan lines of a character together. Each scan line starts on a byte boundary following the previous scan line of the character. Therefore, the bits for line y of character n start at the beginning of byte:

\[(n \times \text{Height}) + y\] 

This format eliminates all those nasty bit shifts required for extracting characters in packed fonts (we’ll discuss this momentarily), but wastes an average of NumChars*Height/2 bytes per font. This can add up.
Also, it takes a multiply to find the beginning of the character. While an integer multiply takes only around 30-40 clocks (on a 286), all these multiplies can add up (especially on an 8088, where a multiply is > 100 cycles). What if the font is large enough that the bitmap is greater than 64K? (Not an unreasonable idea.) However, a long multiply takes multiple instructions.

Packed Linear
In this format, the system stores the bits for a character together. Very together (see Figure 3). The bits for scan

---

**Figure 4 — Unpacked Rectangular Format**

<table>
<thead>
<tr>
<th>Bit</th>
<th>BYTE</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To get a character:

```c
char* get = bitmapstart + n << 1; /* location of line 1 */
for (int ct = 0; ct < Height; ct++)
{
    Put 16 bits from *get on screen;
    get += 512;
}
/****/
```

**Figure 5 — Packed Rectangular Format**

<table>
<thead>
<tr>
<th>Bit</th>
<th>BYTE</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
<td>1A</td>
<td>2A</td>
<td>2A</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>1B</td>
<td>2B</td>
<td>2B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
<td>1C</td>
<td>2C</td>
<td>2C</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
<td>1D</td>
<td>2D</td>
<td>2D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>1E</td>
<td>2E</td>
<td>2E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

To get a character:

```c
int RowWidth = (NumChars * CharWidth);
int ByteOffsetInRow = (n * CharWidth) / 8;
int BitOffsetInByte = (remainder of above);
char* get = BitmapStart + ByteOffsetInRow;
for (int ct = 0; ct < Height; ct++)
{
    put *get shifted left by BitOffsetInByte on screen;
    get += RowWidth;
}
/****/
```
Figure 6 — Windows FNT Font File Header Definition

typedef struct
{  
unsigned int dfVersion;  // version # of font file format (256) */  
unsigned long dfSize;  // file size in bytes */  
char dfCopyright([60]);  // vendor copyright notice */  
unsigned int dfType;  // type of font - bit 0 on if vector */  
unsigned int dfPoints;  // point size at which char looks best */  
unsigned int dfVertRes;  // dpi at which font was digitized */  
unsigned int dfHorizRes;  // see above */  
unsigned int dfaa;  // no. of rows from top to baseline */  
unsigned int dfInternalLeading;  // leading included in chars */  
unsigned int dfExternalLeading;  // recommended leading outside chars */  
unsigned char dfItalic;  // 1 if italic font, else 0 */  
unsigned char dfUnderline;  // 1 if underlined, else 0 */  
unsigned char dfStrikeOut;  // 1 if strikeout, else 0 */  
unsigned int dfWeight;  // stroke weight on 1-1000 scale */  
unsigned char dfCharSet;  // char set standard, FF for IBM PC */  
unsigned int dpixWidth;  // width of char cell, or 0 if variable */  
unsigned char dfPixHeight;  // height of char cells & bitmap */  
unsigned char dfPitchAndFamily;  // low bit set if variable pitch */  
unsigned int dfAvgWidth;  // width of character 'X' */  
unsigned int dfMaxWidth;  // widest character in var. fonts */  
unsigned char dfFirstChar;  // ASCII of first char in bitmap */  
unsigned char dfLastChar;  // ASCII of last char in bitmap */  
unsigned char dfDefaultChar;  // char to use if requested not there */  
unsigned char dfBreakChar;  // char that defines word breaks */  
unsigned int dfWidthBytes;  // # of bytes in one row of bitmap */  
unsigned long dfDevice;  // disregard */  
unsigned long dfFace;  // offset in file of font name */  
unsigned long dpBitsPointer;  // used internally by Windows */  
unsigned long dpBitsOffset;  // offset of bitmap in file */  
} WinFontHeader;
/**/*

Figure 7 — ReadWinFont

/**** Read a Windows FNT file into global arrays ********/  
WinFontHeader WinFont;  
char bitmap[32767];  // watch out you don’t read anything bigger */  
int chofstable[256+1];  
char facenama[17];

void ReadWinFont(char filename[])
{  
 /* note: for brevity I have removed all file error checking */  
FILE* fin = fopen(filename,”rb”);  
fread(&WinFont, sizeof(WinFont), 1, fin);  
if (WinFont.dpPixWidth == 0) /* indicates a variable pitch font */  
/* read character offset table */  
fread(&chofstable,  
(WinFont.dpLastChar-WinFont.dpFirstChar+2)*2, 1, fin);  
fseek(fin, WinFont.dpBitsOffset, SEEK_SET);  
int bitmapsize = WinFont.dpWidthBytes * WinFont.dpPixHeight;  
fread(bitmap, bitmapsize, 1, fin);  
fseek(fin, WinFont.dpFace, SEEK_SET);  
fread(&facenama, 17, 1, fin);  
fclose(fin);  
} /* ReadWinFont */
/**/**/  

The arithmetic for figuring out where a scan line is located gets more complicated, if you’re still talking in bytes. Addressing according to bits simplifies things a little. However, except for the 34010, most processors have a tough time dealing with bits.

The bits for line y of character n start at byte:

\[((n*Height) + y) * CharWidth) / 8\]

and bit (7 - remainder) where remainder is the remainder of the division by 8. It’s (7 - remainder) instead of just (remainder) because most PC graphics adaptors consider the “left” bit of a byte to be bit 7 (although bit 0 is the left on 34010s).

Packed Linear eliminates the waste seen in Unpacked Linear format, but makes for all kinds of unnecessarily complex bitshifts just to get a character out of the bitmap. The amount of shift is not only different for every character, but for every scan line of every character.

And, just like Unpacked Linear, you still have to do a multiply to find the beginning of a character.

Unpacked Rectangular

In this format, the system stores the first lines of all characters, followed by the second lines of all characters, the third lines, etc. Each piece of a character starts on a byte boundary. In this format, line y of character n is at

\[((n*WidthInBytes) + (y*BitmapWidth))\]

where BitmapWidth is CharWidthInBytes*NoCharsInSet. I know this sounds silly, but there are reasons for it. Really.

The advantage is that you can successfully retrieve any character without doing any multiplies (remember how much time a multiply takes). For example, Figure 4 shows the algorithm to get the nth character in an unpacked rectangular bitmap with character cells 16 bits wide.

Unfortunately, although fast, you waste just as much space in Unpacked Rectangular format as in Unpacked Linear Format.

Packed Rectangular

Windows and Gem use the packed rectangular format (see Figure 5). The system stores all the bits for a certain
row packed together (paying no attention to byte boundaries) followed by the bits for the next row of characters. The bits of scanline y of character n start somewhere in byte:

\[(y * \text{BitMapWidth}) + (n * \text{CharWidth}) / 8\]

where BitMapWidth is \((\text{CharWidth} \times \text{NoCharsInSet}) / 8\) rounded up to the next byte boundary (actually the next word boundary, for Windows and Gem).

Within that byte, you can find the bit offset of the start of the character by taking \((7 \times \text{remainder})\) where remainder is the remainder of \((n \times \text{CharWidth}) / 8\). All bits from this bit to bit 0 are part of the character. So if remainder is 0, all the bits in the byte are part of the character (unless the character is less than 8 bits wide).

Now this looks screwy, doesn’t it? It can’t be as fast as Unpacked Rectangular format (you’ve got to shift the character data around to normalize it), but it is more compact. It only wastes an average of \(\text{Height} / 2\) bytes per font (\(\text{Height} \times 1.5\) if each row rounds to a word boundary instead of a byte boundary).

Let’s investigate how we would get a character out of this type of bitmap. The algorithm in Figure 5 will get all the lines of a single character, “n”.

Realize that I’m simplifying things quite a bit when I say “get shifted left by BitOffsetInByte.” I admit I haven’t devoted enough time to this subject yet to come up with a detailed algorithm.

I wouldn’t want to make you suffer through another bit-by-bit shift of the character onto the display. Heck, Larry already did that in the last issue. (If I was being threatened with a snowball down the throat of my X24 if I didn’t show you something that worked, I would probably show you the same chunk of code.)

Back to the subject ... Well, this looks like the most speed inefficient storage method yet. Look at all those multiplies. At least the bit shift for all the rows of a character is the same.

Notice the first multiply is constant as long as you use the same font. You could do it once when you load the font and then forget about it. Both Gem and Windows include RowWidth as part of the font header record, so you never have to do it at all.

The second multiply is different for each character in the font. There are at most only 256 characters in any font, so we can easily build a “character offset” table when we load the font. Gem includes this table as part of the font file, and with Windows it’s an optional addition to the file (used if the font has variable width characters).

The divide... well, everybody knows that a divide by 8 is nothing more than a shift right by 3.

So, by fooling around a bit, and by using 516 bytes more memory (514 for the character offset table and 2 for RowWidth), we have eliminated all multiplies from the operation of putting a character on screen.

I guess that’s why the makers of the two most widely accepted graphics interfaces adopted the same format.

Windows Fonts

The Windows font files I’m talking about here are the ones that have an FNT extension, not the ones on the Windows runtime disk which have an FON extension. You produce FON files by combining several FNT files with the Windows Resource Compiler, which shuffles the information around. The FON files contain the same information, they just don’t follow the format given in the Developer’s Kit.

Also, I’m only talking about raster fonts — those represented in the file by bitmaps indicating which dots should be on and off within a character. Vector fonts (those represented by lists of vec-
Figure 10 — Font Conversion

/*-- Fill in a Windows Font Header from a Gem Font Header --*/

void ConvertG2W( )
{
    /*-- Fill in a Windows Font Header from a Gem Font Header --*/
    void ConvertW2G( )
{
    /*-- Fill in a Gem Font Header from a Windows Font Header --*/
    int numchars = (WinFont.dfLastChar - WinFont.dfFirstChar + 1);

    WinFont.dfVersion = 256;
    strnset(WinFont.dfCopyright,'c',60);
    WinFont.dfType = 0;
    WinFont.dfPoints = GemFont.gfFormHeight;
    WinFont.dfVertRes = 72;
    WinFont.dfHorizRes = 72;
    WinFont.dfAscent = GemFont.gfTopLine;
    if (WinFont.dfAscent == 0)
        WinFont.dflnternalLeading = 0;
    WinFont.dfExternalLeading = 0;
    WinFont.dftalic = 0;
    WinFont.dfUnderline = 0;
    WinFont.dfStrikeOut = 0;
    WinFont.dfWeight = 200;
    WinFont.dfCharSet = 255;
    WinFont.dfPitchAndFamily = 1; /* low bits indicates proportional */
    WinFont.dfSize = WinFont.dfFace + strlen(facename)+1;
    WinFont.dfFace = WinFont.dfbitsOffset
    + ((numchars+1)*2);
    WinFont.dfBitsPointer = 0;
    WinFont.dfChars = 0;
    WinFont.dfBitsOffset = (((sizeof(WinFont) && 1) == 1) ?
        sizeof(WinFont)+1 : sizeof(WinFont));
    WinFont.dfFace = WinFont.dfbitsOffset
    + (WinFont.dfbitsBytes * WinFont.dfPixHeight);
    /* NOTE: The following is out of sequence because it depends on
     * other computed values in the header */
    strncpy(facename,GemFont.gfName,17);
    WinFont.dfSize = WinFont.dfFace + strlen(facename)+1;
} /* ConvertG2W */

/*-- Fill in a Gem Font Header from a Windows Font Header --*/

void ConvertW2G( )
{
    /*-- Fill in a Gem Font Header from a Windows Font Header --*/
    int numchars = (WinFont.dfLastChar - WinFont.dfFirstChar + 1);

    GemFont.gfID = 2;
    GemFont.gfPointSize = WinFont.dfPoints;
    strcpy(GemFont.gfName,facename);
    GemFont.gfFirstChar = WinFont.dfFirstChar;
    GemFont.gfLastChar = WinFont.dfLastChar;
    GemFont.gfTopLine = WinFont.dfAscent;
    GemFont.gfAscentLine = WinFont.dfAscent+1;
    GemFont.gfBottomLine = WinFont.dfPixHeight-WinFont.dfAscent-1;
    GemFont.gfMaxHeight = WinFont.dfPixHeight;
    GemFont.gfMaxCellWidth = WinFont.dfPixHeight-WinFont.dfAscent-1;
    GemFont.gfRightOffset = 0;
    GemFont.gfThickening = 1;
    GemFont.gfUndisSize = 1;
    GemFont.gfGiftHeight = 0x5555;
    GemFont.gfGiftWidth = 0x5555;
    GemFont.gfflags = 0;
    GemFont.gfHorOfsTable = 0;
    GemFont.gfCharOffsTable = ((sizeof(GemFont) & 1) == 1) ?
        sizeof(GemFont)+1 : sizeof(GemFont);

continued on next page

MICRO CORNUCOPIA, #46, Mar-Apr, 1989 57
PageMakers care about internal leading and external leading. I just want to print something on the screen and don’t care about one dot this way or that, usually.

Figure 9 shows a C++ function to read a Gem font into global memory buffers similar to the function for Windows fonts. Gem uses different field names (dfWidthBytes is gfFormWidth). Also, the character offset table is not in a fixed place in Gem fonts, while the font name is.

C++ Function??

An aside here. When I say “a C++ function,” I mean “a C function that I compiled with Zortech C++,” so it probably won’t compile under normal C.” I haven’t gotten into C++ yet, so these examples don’t look too “plussy.”

Don’t think of them as examples of C++ programming. Think of them as examples of dealing with fonts in a high level language.

Partial Character Sets

Both Windows and Gem support fonts with any number of characters up to 256. (It appears Gem will support up to 65,535, but I haven’t tried it.) The lowest character in the set is character dflFirstChar (gfFirstChar for Gem) and the highest character in the set is dflLastChar (gfLastChar).

To avoid wasting space on characters that aren’t even in the font, the system stores the character “FirstChar” at the very first position in the bitmap, and the bitmap ends after LastChar.

Because of this, your character display function should always make sure the character you want to display lies somewhere in the range FirstChar – LastChar. If not, you should display DefaultChar (or nothing).

Don’t forget that you find character “n” at position “n-FirstChar” in the bitmap.

From Windows To Gem And Back

The functions ConvertG2W() and ConvertW2G() in Figure 10 show how the fields of the two font types correlate. Use them if you want to read both font types, but write all your font processing routines to use a single font format. The bitmaps and character offset tables are identical for both fonts.

Speaking of multi-compatibility, the function IsWindowsFont() does a simple check on a font file and returns true if it fits the rules of a Windows font file. It only checks that the dfSize field equals the length of the file in bytes, but that’s enough for mere mortals.

As long as you’re decently careful, you can mix the use of Windows and Gem fonts transparently. Just call IsWindowsFont() on a file before you open it, then read it into the proper buffers and do the conversions. You decide which format to use internally. I would use the Windows format, but that’s just because I figured it out two days before Gem, so I’m more familiar with it.

Even if you use Windows format internally, you should call MakeChOfsTable() for fixed width fonts in order to find characters faster. Since Gem requires a character offset table, even for fixed width fonts, you have to support character offset tables to be able to display Gem fonts, anyway.

Character Offset Table

Speaking of that, I almost forgot to explain the format of the Windows and Gem character offset tables.

The character offset table has (dfLastChar - dfFirstChar) + 2 word entries which give the offset in bits of each character from the left side of the bitmap rectangle. You can find the width of a character n by subtracting charofstable[n+1] from charofstable[n]. Hence the extra entry at the end, used to determine the width of the last character.

As with the bitmap, the offset for character n is at charofstable[n-FirstChar].

Putting It All Together

Okay, so I’m too proud to flick bits, am I? I think it makes me look stupid and old-world. I would just feel too guilty leaving you with absolutely no way of looking at those fonts you ripped off from your buddy’s copy of Ventura.

The C++ function in Figure 11 writes a character from a font onto the screen.
as a bunch of asterisks. This doesn’t do much in practical terms, but at least it verifies that you understand what’s going on. Two Brownie Points to the first one of you who figures out how to integrate this into Larry’s printer graphics library in issue 45.

If any of you have an algorithm that’s more efficient, I would be truly honored to take a peek at it. If I get any good ones, maybe I’ll print them.

PCT Font Format?

Oh, yeah. I promised to tell you about our own font format. The Unpacked Rectangular format now stores the PC Tech font format. This is because we use it only for the text mode display on the 34010.

With our 34010s in text mode, there’s loads of extra memory sitting around. (Even the smallest PC Tech 34010 has 256K of nondisplay memory and 256K of video memory.) So space isn’t a concern, but the fastest performance possible is.

In PC Tech format, as with any other semi-intelligent 34010 font format, the systems stores all the bits within a byte in reverse order from the Windows and

Gem fonts. That’s because, as mentioned before, the 34010 is a bit addressed processor, so the left bit is bit 0.

The header for PCT format fonts contains more or less the same information that’s in the Gem and Windows headers. Just less of it. The actual format doesn’t matter, since you’ll probably never see a file in this format anyway (unless you get a 34010 board). Mainly I just mentioned it to show you that no single format is best for all situations.

Oh, and by the way, the entire internal loop of the algorithm I gave for packed rectangular form is a single instruction on the 34010. Including the bit shifts and masks.

You wonder what’s kept Earl glued to his screen for the last year ...

Next Time...

... the magic of the 34010 PIXBLT instruction and other wonders of the electronic world.

P.S. Remember — no matter what they tell you, Raisin Bran is an addictive narcotic. Please take it only as your doctor advises.

***
PC-File:dB & PC-Write 3.0
Definitely New & Definitely Improved.

By Anthony Barcellos
P.O. Box 2249
Davis, CA 95617-2249
(916) 756-4866

For those of you tuning in late, Tony Barcellos is software librarian and newsletter editor for the Sacramento PC Users Group. In his spare time he teaches math and writes study guides, but it's his knowledge of shareware that has made him famous among software authors and has brought him to the pages of Micro C.

A shareware columnist has no excuse for writer's block. There's entirely too much to pick from in this corner of the computer industry. The usual problem is deciding what to write about.

A couple of giants in the shareware business made my job easier this month. ButtonWare released PC-File:dB and Quicksoft shipped version 3.0 of PC-Write. Both programs instantly jumped to the top of my topic list.

Single File
There are plenty of competing database products out there, even if one forgets the high-end products for a moment. (By "high-end" we mean in terms of price, of course.) Despite worthy competitors like ExpressWare's File Express, however, ButtonWare's PC-File continues to attract the most attention.

Rightfully so. Jim Button's famous database program was half of the dynamic duo (PC-File and PC-Talk) that established the credibility of user-supported software. While PC-Talk has faded, PC-File is still going strong.

So what has Jim Button done for us lately? Although his company's offerings have expanded over the years to include several other programs (see the May-June 1988 issue of Micro C), PC-File remains the centerpiece. Still stressing ease-of-use via menus, now enhanced by context-sensitive help screens, PC-File calls its new incarnation PC-File:dB. Guess what the "dB" stands for.

"By making PC-File dBASE compatible," says Jim Button, "I can now offer dBASE users an easy-to-use interface for their current data."

Since ease-of-use is conspicuously missing from the vocabularies of most dBASE users, PC-File:dB goes head to head with Ashton-Tate's dBASE III Plus. (Now that dBASE IV is finally shipping, Ashton-Tate can claim to having addressed the ease-of-use problem. However, since the file format remains the same, PC-File:dB should be adaptable to dBASE IV as well.)

Recent PC-File enhancements are still there, of course. PC-File:dB's predecessor, PC-File+, sported integrated graphics. Hence you can whip up pie charts or bar charts with your ButtonWare database.

The bar charts come in several different flavors, or you may choose line or scatter plots instead. The pies explode (for your enjoyment, of course) while bar and line charts offer razzle-dazzle logarithmic scales. This is heavy-duty stuff.

Keyboard macros, long a PC-File feature, can be created by example within PC-File:dB. Just turn on the macro feature and let PC-File record your keystrokes. With a little ambition, you can set up a macro that closes off data entry, sorts the data, prints a report (perhaps calling user-created report formats from disk), and puts you back where you began.

Not only can you save report formats (which PC-File has supported since much earlier versions), but now you can create them by painting the layout of your report on-screen.

The PC-File:dB laundry list of features is impressive. PC-File:dB has five data types: numeric, character, logical, date, and memo.

ButtonWare provides a few details about the program: Maximum number of records per database: 1 billion; maximum record length: 4000 characters; maximum field length: 254 (with exceptions for special fields like date fields, which have prescribed lengths); maximum memo field length: 5000 characters (even greater than the standard record length); maximum number of fields per database: 60; maxi-
Write hardly seems to be suffering from
$89.95 + $5 shipping
Bellevue, W A

ments:

warhorse word processor? After more user's guide and a special beginner's program's heritage argues that comes on three diskettes with a case doesn't seem likely to hold up of the disks.

Big and beefy, PC-Write now complete user's manual occupying one it has added over 500 features.

Here are a few of the major improvements:

• The old 60K file limit is gone. PC-Write 3.0 will use all available memory for files.
• It supports and displays multiple columns.

• You can grab rectangles of text and delete, copy, or move them.
• It has menu-selected fonts and special effects.
• New quick keys bypass menu operations for greater speed in switching files and printing.
• Bob's integrated the editing and printing modules. (And they remain in memory whenever possible.)

Besides the word processor, Quicksoft offers companion programs like the DCA utility ($29), which converts files between DCA-RFT and PC-Write formats. Font Selector ($29) chooses and downloads soft fonts to your Hewlett-Packard LaserJet+ or compatible laser printer. Finally, there's the PageMaker Import Filter ($10), which saves PC-Write files in a format PageMaker understands. (It preserves much of PC-Write's formatting.)

"PC-Write 3.0 is a giant leap in capabilities," says Bob Wallace. He's right.

PC-Write 3.0
$89
Quicksoft, Inc.
219 First Avenue North, #224
Seattle, WA 98109
(800) 888-8088 or (206) 282-0452

Archival Arch Rivals

The continuing legal dispute between System Enhancement Associates (SEA) and PKWare over SEA's ARC utility and PKWare's rival software is a dismal thing to follow. As I mentioned in my last column, the details of the lawsuit's settlement became public despite the agreement of the two parties to refrain from commenting on it.

While PKWare's Phil Katz remains the beneficiary of sympathy from most of the user community, SEA's Thom Henderson has explained his grievances in detail. He says his lawsuit was necessary to protect the integrity of his ARC program. (I thank Michael Kaufman for sending along the latest archive of bulletin board messages on the topic.)

Henderson claims that Katz copied ARC's program code in creating PKARC and PKXARC, utilities that do all ARC's archiving functions (and then some) with major improvements in speed. Katz's defenders rebut Henderson by declaring that the compression algorithms in ARC and PKARC derive from the same public domain sources.

Henderson continues his argument by asserting that Katz would be in violation of copyright law even if he hadn't copied any of SEA's code (although Henderson is convinced that he did).

Henderson argues that Katz's admission of seeing SEA's program code for ARC establishes a presumption of infringement on SEA's copyright, even if Katz then created PKARC and PKXARC from scratch.

The original court settlement was in SEA's favor, while the court of public opinion seems to be coming down on PKWare's side. Whether Henderson's efforts to explain his position to users will shift the tide in his favor remains to be seen.

I'm sick of the entire business so I'll just wait for the tragicomedy to come to a conclusion. Somebody — anybody! — get the hook!

Editor's note: See the Letters Column in this issue for another point of view on the SEA vs PK controversy. And, stay tuned for more next issue.

Cards And Letters And Disks

My thanks to all of you who have contacted me. It's a struggle to keep up with it, let alone do justice to its volume. Besides the letters and notes, there are the stacks of disks and programs that I need to look over when I find the time. (Time? Anyone have the time?)

My forays into the stacks take on the aspects of an archaeological dig. Where's that program I wanted to write about that does graphics in Encapsulated PostScript format (no "jaggies" when you scale an illustration)?

How about that critically acclaimed spreadsheet package that was withdrawn from the regular commercial channels and relaunched as shareware? (It's around here somewhere, along with the nice cover note written by the author.) Then there's the program that purports to bring true 3-D CAD features to shareware for the first time. And the programs I talked about before keep getting updated.

There's no way I'll ever catch up. But the fun is in the trying. Keep sharing.

PC-File:dB
$89.95 + $5 shipping
ButtonWare, Inc.
P.O. Box 96058
Bellevue, WA 98009-4469
(800) JBUTTON, Toll-free order line

Write On!
The other entrant in the geriatric shareware sweepstakes is PC-Write from Quicksoft. Now at version 3.0, PC-Write hardly seems to be suffering from old age. Big and beefy, PC-Write now comes on three diskettes with a complete user's manual occupying one of the disks.

What has Bob Wallace done to his warhorse word processor? After more than a year of teasing about what PC-Write 3.0 would offer, Quicksoft reports it has added over 500 features.

Here are a few of the major improvements:

•••

MICRO CORNUCOPIA, #46, Mar-Apr, 1989 61
I received a small brown package the other day and inside I found a small black disk. Nothing more. On the label someone had scribbled “Personal Relationships, an Expert System for Adults Only.”

I fired up my system, popped in the disk, and looked at the directory.

Aha, USEME.EXE.

Welcome to a new world of relationships. To take full advantage of this software you must:

1. Find someone compatible.
2. Find a bottle of bubbly and two glasses.
3. Turn up the heat.
4. Turn down the lights.
5. And hit ENTER.

I ran upstairs, grabbed Sandy, found the bottle of carbonated apple juice we keep for medicinal purposes, and headed back to the computer.

I hit ENTER.

Please enter your name.
DAVE

Please enter your partner’s name.
SANDY

Please select the type of relationship you’d like to develop.

• 1. Non-touching, mostly platonic. (e.g. husband/wife)
• 2. Good friends (cheek kisses and hugs on special occasions).
• 3. Very Good Friends (e.g. quarterback/cheerleader).
• 4. Consumate Roommates.

I selected 1. We’d already practiced that one.
I’m sorry, that selection is not available.
I cranked up my courage and selected 2.
I’m sorry, that selection is not available.
I’ve never tackled football but after 20 years of matrimonial bliss, “Very Good Friends” sounded like something we could work toward.

I selected 3.
I’m sorry, that selection is not available.
I bit my lip and selected 4.
Good choice.

From here on, I’ll take the lead. Timing is critical so please do not begin any activity before I tell you. Continue the activity until I beep. At the beep, drop what you’re doing and read this screen.

“Beep!”
First, get comfortable. Put on a quiet record, take off your shoes, loosen your tie or collar, and take a sip or two of the bubbly. The lights should be low. Remove any distractions.
I dumped the cat off my lap.
“Beep!”
Now sit together side by side. DAVE, reach over and grasp SANDY’s hand. Gently. Maintain that position. Do not speak.
“Beep!”
Now, DAVE, put your arm around SANDY. Notice the warmth, DAVE, that begins where your arm touches SANDY. Notice how that warmth travels back down your arm and spreads throughout your entire body. Notice especially how that warmth settles into your solar plexus.
I dumped the cat off my lap. Again.
“Beep!”
SANDY, notice the warmth from DAVE’s arm. Notice how it spreads, making your breathing heavier. Almost audible.
“Beep!”
Now, before we begin the unbuttoning, DAVE and SANDY, you need to enter your Personal Relationships registration code. If you don’t have a number yet, you can obtain one immediately by calling 1-800-NDR-WARE. Be sure to have your VISA or MasterCard handy.
Registration Fee:
For $100 you get the code immediately.
For $50 we call you back after you’ve cooled off for 5 minutes.
For $15 we call you back next week.

Note: Your shameless activity, DAVE and SANDY, has been recorded in the system, along with the time and date (you did, after all, select number 4). If you attempt to exit the program at this time (via reset, power switch, etc.) to avoid registering, this machine will, forever, at random, display the gory details of this tryst on the screen.

“Beep!”
Please enter your registration code:

“Beep!”
Please enter your registration code:

“Beep!”
Please enter your registration code:

“Beep!”

✦✦✦
20 MEG HARD DRIVES

SEAGATE Model ST-225
(Full 6 Mo. Warranty)less face plate $219.95
Western Digital Controller For Model ST-225 Drive $69.95 w/cable
Yes! These are for IBM compatible machines.

SHUGART #465

QUAD DENSITY, 720 K, 96 TPI
1/2 HEIGHT
BLACK FACE PLATE (NEW BUT HAVE COSMETIC BLEMISHES)
SOLD AS IS $49.95

TOSHIBA 5 1/4" FLOPPY DRIVES

DSD, 360 K
1/2 HEIGHT
BLACK FACE PLATE (90 DAY WARRANTY) $89.95

TOSHIBA 3.5" DISK DRIVES

IBM compatible
720K Byte
Double sided
Mounting kit
Power & Data cable adapter
Model # FDD-4210GOK

FULL 90 DAY WARRANTY $119.95

5.25" DISK DRIVE CABINETS
(with power supply)

For 1 full height or 2 half height $99.95 each

300 BAUD SMART MODEM (DIRECT CONNECT)

LOW COST SERIAL MODEM
7 x 10 x 1 1/2 $19.95

CABLES-CONNECTORS-SEX CHANGERS

"XT" STANDARD PARALLEL
(10 DB-25M TO DB-25M) $10-004 $12.95
AC POWER CORD
(2 CONTACTER STANDARD, NICED INGS) $10-001 2.95
RS232 MODEM ADAPTER
(CS-25 M TO F OR CS-25 F TO F) $20-001 9.95
SEX CHANGER, DB-25 F TO F $20-006 9.95
SEX CHANGER, DB-25 M TO M $20-005 9.95

COMPUTER POWER SUPERVISOR

A LOW COST INSURANCE POLICY PROVIDES SURGE/SPike PROTECTION FOR YOUR COMPUTER!

- 5 OUTLETS EACH WITH A LIGHTED SWITCH
- 1 MASTER ON OFF SWITCH
- SYSTEM CABLE PROVIDES FULL SHIELDING
- Input 15 AMP, 120 VAC, 170 WATTS
DIM 12.5" x 2.5" x 14" Regular price $69.95 Your price $49.95

PHONE: (206) 682-5025
FAX: (206) 682-5593
M-F 9-6 SAT 9-5

Distributors of Electronics Since 1968
1123 Valley Street • Seattle, WA 98109-4425

Reader Service Number 101
By Kent Peterson
Psion Interest Group
41 Greenridge Avenue
White Plains, NY 10605
(914) 761-2196

You asked for it, more personal stories from “On Your Owners.” Well, this is personal. Kent talks about his new start-up, a newsletter for a very small computer. (Definitely not the Big Board.)

Last year I landed the kind of job my mother always told me I wanted. I became a systems analyst for a fairly large corporation which gave me job security, a pension plan, full medical, and a fast AT with almost enough disk space.

Still, the hacker in me wasn’t content. I spent more of my time in meetings than I did in microcode. My jeans and Nikes didn’t look quite right with my white shirt and tie. I knew there was more to life, but I also needed the money my “real job” provided. How depressing.

As usual, my wife, Christine, knew just how to snap me out of my depression. “You need a gadget,” she said. She was right. Somewhere among the megabytes and meetings, life had stopped being fun. I had started in this business hacking on little machines and I missed the challenge of pushing a little machine to its limits.

Then, one fateful day, the boys down in accounting called up to tell me I hadn’t claimed enough of my miscellaneous moving allowance. I tried to explain that it hadn’t cost much for me to move, but they insisted that the company owed me a couple hundred dollars. Furthermore, if I didn’t take it, it would screw up their bookkeeping.

Of course I refused their offer (for about seventeen nanoseconds). I spent the money on a funky little machine called the Psion Organiser II. (Editor’s note: Psion is pronounced SIGH-on, a psemi-psimple name.) This eight ounce box, billed as the “pocket PC,” was just what I was looking for. The unit contains a 1 MHz Hitachi 6303X microprocessor (a CMOS version of the Motorola 6809) and 32K of RAM. It has a calculator-style keyboard and a two line liquid crystal screen. It runs a proprietary language called OPL.

The Organiser doesn’t use disk drives; it uses tiny packs that contain either EPROMs or battery-backed CMOS RAM. It even reads from and writes to these packs while communicating through an eight-bit (proprietary) parallel port.

Psion also manufactures a device called a CommsLink that converts this port to a smarter RS-232C. In short, the Psion offered me enough sameness along with enough weirdness to stave off boredom for awhile.

After developing a relationship with the little Psion, Christine and I wrote a few little music routines and I managed to squeeze in a text adventure. Then I got serious. I made it talk to every other computer we had — the AT, the Hewlett-Packard, and the Atari.

The Psion keyboard lacks some characters that I consider essential (like the question mark). So I dug through the tech manual until I figured out how to re-vector the keyboard lookup table into RAM to give me an alternative keyboard layout. What fun. I was having so much fun with the little machine that I decided to do something crazy.

Thoughts Of Grandeur

“I’m going to become the Peter Norton of the Psion Organiser II,” I told my wife.

“Oh no,” she said. “Does this mean you’ll start wearing pink shirts and ties and stand around with your arms folded and your shirt sleeves partially rolled up?”

I clarified my position. I wasn’t talking about a wardrobe change. I just figured I could become the acknowledged expert on the Organiser. This would require staying current on the latest developments in the Psion world, getting all the latest hardware and software, and spreading that information within the Psion community.

“Sounds expensive,” Christine said.

“Not if we do it right,” I countered. “I’m going to make this machine pay for itself.” She just smiled. I’ve said that about every computer I’ve ever owned and it’s seldom proven true.

Newsletteritis

I decided to start a users group for the Organiser II and publish a small newsletter of program tips, techniques, and reviews. I figured membership dues would cover the cost of printing and mailing the newsletter with
enough left over to keep me stocked with the latest hardware and software.

I checked, there was no Psion users group in the U.S. The British build the little computer so there’s an active group there.

Jeff Etter, who’d sold me my Psion, thought the U.S. market was rich enough to support a group and offered to help any way he could.

The plan we formed was simple. I’d write the first issue of our newsletter. Christine (a former editor) would proof the copy and correct any misspellings. When everything passed her inspection, we’d print the script on a borrowed laser printer. Then we’d take the laser printouts to a local print shop. (The first run totalled 100 copies.)

I planned to send these copies, together with subscription forms and pleas for contributions, to 100 of the people who’d bought Psions from Jeff. If we were lucky, we’d make back the money we’d spent on the first printing and be able to keep going. If not, Christine would hopefully remember she’d married me for poorer, too.

Really Cheap Desktop Publishing

Our plan depended on desktop publishing, but we had no budget. We didn’t have Ventura, a scanner, or our own laser printer. We did have PC-Write. This forced us to adopt a very simple format for the newsletter: single column, no photos, and no graphics beyond the simple box-drawing characters that PC-Write supports. This no-frills approach gave us some unexpected benefits. Since we didn’t have one of those wisi-wonderful graphics packages, I spent my time writing instead of playing around with graphics. (It seems like whenever I get hold of a mouse and a bit-mapped image, I spend more time tweaking than working.)

Another benefit was that we could print our proof sheets on the trusty old Proprinter rather than use the laser printer for everything. Finally, since PC-Write uses straight ASCII, the output could be used in a disk version of our newsletter. (Of course, most of this was just rationalization. When Ashton-Tate offered us a copy of BYLINE, we jumped at it.)

PIGs

We immediately realized our users group needed a name, so Christine and I discussed possibilities: “We can’t call it the Psion Interest Group,” I said.

“Why not?”

“The initials spell ‘PIG.’ We can’t have a group called PIG. It isn’t dignified.”

“I think it suits you,” she said sweetly.

I looked around my office at the scraps of paper covered with program fragments, the semi-random stacks of books and computer magazines, the empty Oreo bags and Pepsi cans.

“I still don’t think it’s a very good name,” I said. “Can’t we come up with something else?”

“You could always call it the Psion International Software Society,” she chuckled.

“I guess PIG isn’t such a bad name,” I said.

“Oink, Oink,” she squealed.

We set to work writing, editing, and programming. We worked ideas into programs and wrote articles to accompany the code. We pressed anything with a keyboard and an RS-232 port into service for the writing. The Psion’s small size and portability made it a natural for note taking. Anywhere inspiration struck, the Psion was there. (It’s amazing how many good ideas come to you when you’re in the bathroom.) The Psion also served as a bridge among different machines. They couldn’t agree on disk formats, so the Psion served as an ASCII Esperanto that they all could talk to.

Finally we finished our first issue, sent out the 100 copies, and waited for people to make PIGs of themselves.

They did. Not all, not a majority, but enough. And, Jeff Etter kicked in some cash to help with postage costs. PICO magazine did a review of the Psion and fortunately the review contained a technical error. I wrote a letter correcting the error and mentioned our users group. Lo and behold, more PIGs.

Greg Paul, the director of technical support for Psion, USA, learned of our efforts and directed people our way. One of our members contacted the British users group and we established a publication swap with them.

We not only gained members, we gained a few good writers and a couple of hotshot programmers. Ideas came pouring in. PIGs are mostly users, not programmers. But the users have plenty of ideas for keeping programmers busy.

I have always relied on the strange-ness of kind people and the kindness of strange people.
Problems
Of course, not everything went smoothly. When I was ready to print the second issue of our newsletter, the laser printer was nowhere to be found. It seems the guy who owns the printer lent it to someone else. (That’s the problem with relying on one of these generous types, you never know when they’re going to be generous to somebody else.)

A few frantic phone calls connected us with a laser-printer owner who let us print the 24 pages of the second issue during lunch hour. To our relief, and our subscribers’ amazement, the second issue hit the streets reasonably close to its intended date.

A Conference
Before the third issue, our editorial staff (and the kids) flew down to sunny Florida for three days of sun, fun, and what they would like from a users group. (Now I understand exactly how body else.)

It was our first chance to meet many of the Psioneers face to face and discuss what they would like from a users group. We signed up subscribers, played with gadgets, and met some amazing people.

Shortly after the conference, interesting items began appearing in our mailbox. (Now I understand exactly how Jerry Pournelle feels.) Psion programs ranging from word processors and database managers to celestial navigators began piling up in our apartment.

The most significant piece of software wasn’t for the Psion, however. It was a copy of BYLINE, Ashton-Tate’s desktop publishing software. BYLINE came to us courtesy of Ashton-Tate and Jeff Etter.

Every Silver Lining Has A Dark Cloud
BYLINE turned out to be a mixed blessing. Our third issue had pushed PC-Write to its limits. (That issue included two simple illustrations and a very simple circuit diagram.) BYLINE promised to give us many more fonts and sizes, along with the easy addition of real graphics. The package was easy to learn, very powerful, and gobs of fun. Our fourth issue was only six weeks late.

In fairness to Ashton-Tate, I have to say that the delay wasn’t BYLINE’s fault. BYLINE performed gallantly; the problems I encountered are inherent in any desktop publishing system.

First, I had to figure out what the publication should look like. Should we use Times-Roman or Bookman? Which point size? How should listings look?

But I’d poured an issue’s worth of text and programs into BYLINE when I discovered an even bigger problem. I only had half a newsletter. Our new typeface and multi-column format held a lot more information per page.

Another problem was the art. They say a picture is worth a thousand words, but I found out that it’s much easier for me to write a thousand words than to mousepaint a picture.

BYLINE could import art from a paint program or a clip art library, but the illustrations I wanted were not available as clip art. A scanner would solve the problem, of course, but my budget didn’t include a scanner.

Unexpected Features
Like any sophisticated piece of software, BYLINE has a few undocumented “features.” A personal favorite is the way it inserts a space following the hyphen in hyphenated words. After I’d tried the usual techniques of manual reading and hair-pulling, I stumbled on the solution. I manually insert an invisible space after the hyphen.

Editor’s note: Of course — an invisible space! Why didn’t I see that? (All omissor are obvious.)

I used a similar technique to get BYLINE to print a program listing containing a less-than sign followed by a quotation mark. BYLINE uses the “<” and “>” as variable delimiters for things like page numbers. It refused to print the program listing until I inserted an invisible space between the “<” and “>”.

BYLINE gave us enormous layout flexibility, but it did limit our printer options. BYLINE supports both dot-matrix and laser printers so we could still use the Proprinter for proofing.

Unfortunately a dot-matrix printer has to do a digital hand stand to print all those wonderful fonts and pictures. So we got a lot of snack breaks while the printer exercised. Of course, we did most of the proofing on the screen, but Christine insists on paper and red pen for her final pass.

After more writing, rewriting, editing, and playing, our fourth issue got shipped in PostScript form to Psion’s U.S. headquarters. They dumped it out on their PostScript laser printer. (None of our local friends had laser printers with PostScript or the font support we needed.) Finally, for better or worse, the issue hit the streets.

So How Are We Doing?
We’re now at work on our fifth bi-monthly issue and we have about 200 members in a dozen countries. Our membership fees ($25 U.S., $40 foreign, and $15 for students, senior citizens and hard-luck stories) bring in enough to cover printing, postage, and the occasional gadget.

If I figured out what I’m making per hour on this venture, it would come to somewhere around six cents. But if I factor in the people I’ve met, the lessons I’ve learned, and the fun I’m having, I’m richer than I’ve ever been.

There is one little thing that bothers me, though. I was really looking forward to being the Peter Norton of the Psion Organiser II.

I neglected one little detail, however. When Peter formed his company, he called it Peter Norton Computing. When I formed mine, I called it PIG. My wife took note of the name and the number of nights I spent at the computer and christened me “The Big Boar.” I have no idea what Peter Norton’s wife calls him (or if she calls him at all), but at least I don’t have to wear a pink shirt.

Psion Organiser II
Weight: 9 ounces
Dim: 6” long 3.8” wide 1.2” deep
Battery: 9 V transistor
Memory: 32K main memory
Language: OPL
Processor 1 MHz Hitachi 6303X
Price $250

Accessories
32K RAM pack (battery backed)
32K EPROM pack (EPROM burner built into Organiser)
128K RAM pack available soon.
Turbo Pascal 5.0
And A Rational Number Toolkit

By Michael S. Hunt
845 E. Wyeth
Pocatello, ID 83201
(208) 233-7539

Herein Michael covers Turbo Pascal 5.0 and unreal reals. I mean real integers. Or is it rational real integers? Ah well, he’ll tell you more precisely than I can.

In this issue I’ll briefly review Turbo Pascal 5.0. (This’ll be brief compared to the extensive Modula-2 compiler review coming next issue.) Then for those of you tired of numeric roundoff error, check out the rational number toolkit. It contains the basic arithmetic functions to compute with rational numbers.

After convincing the post office that my mail should be delivered to my house and not E. B. Hunt three blocks away, I received some reader mail. Remember you can write and complain, explain, applaud or just shoot the breeze. I’ll even answer. I welcome any suggestions for topics or directions for the column. I also check the Micro C BBS about once a week.

Turbo Pascal 5.0
Well, I’ve had Turbo Pascal 5.0 for about two months now. I like it, I like it. I upgraded from version 2.0 so many of the features mentioned are also in versions 3.0 and 4.0.

At last, units. I’m a fanatic about reusable code and problem solving tools. I’ve built and used modules in Modula-2, but now my 5000 line Pascal programs can be managed instead of manhandled.

Procedure passing is essential for truly flexible tools. The classic example is a sort procedure that can pass the appropriate comparison procedure in the parameter list. With a few sort procedures, several comparison procedures, and some carefully constructed selection routines, you can build a very versatile sort package. Hmmm, sounds like a possible column topic.

I have several programs in need of some assembly language speed ups. The ability to link .OBJ files and create .EXE files without overlays is welcome. (You can still do overlays, though.)

Ever since I purchased Logitech Modula-2, I’ve been addicted to debuggers. I could get along without them, but I’d hate to. Debuggers, even rudimentary ones, are tremendous time savers.

The integrated source level debugger in 5.0 is not full featured like CodeView. Borland recommends that you buy Turbo Debugger for industrial strength debugging. Scott Ladd reviewed Turbo Debugger and Turbo Assembler and took a look at what debugging is all about in his C’ing Clearly column. Look for it in Micro C’s January/February issue (#45). You can purchase Turbo Pascal 5.0, Turbo Debugger and Turbo Assembler (as the professional package) for $250.

Turbo Pascal 5.0 supports the 80x87 numeric coprocessors. Compiled programs are smart enough to use the 80x87 if present, or emulate one if not.

This package can take advantage of EMS for overlay files and the integrated editor. The overlay manager can load the overlay file into EMS memory so overlays load faster.

If you have 64K of EMS memory, the edit buffer will be placed in EMS memory. This frees up conventional memory and allows you to debug larger programs. The integrated editor is WordStar compatible with some extensions for special program editing features. The editor is fully configurable so you can customize the commands to mimic your favorite editor.

Borland’s Graphics Interface (BGI) unit contains more than 50 graphics routines. BGI supports CGA, MCGA, EGA, VGA, Hercules, AT&T 400, 3270 PC and IBM-8514. BGI will autodetect all but the 3270 PC and IBM-8514 graphics cards. You can link all the device drivers and have the program automatically choose the best resolution at runtime. BGI is quite extensive and can help give your program that polished look.

The Turbo Pascal 5.0 package also contains a make utility, a smart linker that removes unused code, an integrated environment with context sensitive help system, and well written reference and user guides. If you’re writing in Pascal for IBM compatibles, I strongly recommend Turbo Pascal 5.0.
Get Rational
I do a lot of work with floating point numbers. I get tired of having my results turn out useless because of the accumulated roundoff error after repeated multiplication. Even more subtle is the error introduced when we subtract two numbers very close to each other (i.e., \( |a-b| = 0.0000001 \)) when we have only six decimal places of accuracy.

Even though we think of computers as being very accurate, floating point numbers are just approximations. Integer and cardinal values can be represented accurately as sums of powers of two. Since the computer represents everything in binary (powers of two), we can accurately represent integers and cardinals.

The trick would be to represent real numbers with integers. This is where rational numbers come to the rescue. A rational number is any number that can be represented in the form \( \frac{a}{b} \) where \( a \) and \( b \) are integers and \( b \) is not equal to zero. Since the rational numbers are only a subset of the real numbers, they’re a bit limited. The square root of 2 and pi are not rational numbers.

If your math can be restricted to rational numbers, you can avoid round-off error. The unit Rational (Figure 1) is a set of basic tools for arithmetic with rational numbers. Figure 2 lists the tools.

The Euclidean Algorithm
The real workhorse of the unit is the GCD function, called by all the other procedures and functions in the unit except ZeroRat. Using the Euclidean Algorithm, GCD finds the greatest common divisor for any two numbers.

The Euclidean Algorithm decomposes a pair of numbers \( x \) and \( y \), where \( x \) and \( y \) are not both zero and \( x \geq y \), by calculating \( r = x \mod y \). It then lets \( x = y \) and \( y = r \) and repeats the process until \( r = 0 \). When \( r = 0 \), the previous \( r \) or the last nonzero \( r \) is the greatest com-
Figure 1 continued from page 69

```plaintext
procedure ComDenom(var a, b : ratType); (* find a common denominator *)
var m : baseType; (* for a & b and adjust them *)
begin
  m := LCM(a.d, b.d); (* find LCM of a.d and b.d *)
a.n := a.n * (m div a.d); (* set a.n to new value *)
a.d := m; (* set a.d to LCM of denominators *)
b.n := b.n * (m div b.d); (* set b.n to new value *)
b.d := m; (* set b.d to LCM of denominators *)
end; (* ComDenom *)

procedure IncRat(var a : ratType); (* increment a by 1 *)
begin
  a.n := a.n + a.d; (* add denominator to numerator *)
  LowestTerms(a) (* reduce a to lowest terms *)
end; (* IncRat *)

procedure DecRat(var a : ratType); (* decrement a by 1 *)
begin
  a.n := a.n - a.d; (* subtract numerator from denominator *)
  LowestTerms(a) (* reduce a to lowest terms *)
end; (* DecRat *)

procedure ZeroRat(var a : ratType); (* set a to 0 *)
begin
  a.n := 0; (* set numerator to zero *)
  a.d := 1 (* set denominator to smallest positive number *)
end; (* ZeroRat *)

procedure AddRat(a, b : ratType); (* add a and b and place *)
var c : ratType; (* result in c, a+b = c *)
begin
  LowestTerms(a); (* reduce a to lowest terms *)
  LowestTerms(b); (* reduce b to lowest terms *)
  ComDenom(a, b); (* convert a & b to common denominator *)
  c.n := a.n + b.n; (* in c, a+b *)
  c.d := a.d; (* c to lowest terms *)
end; (* AddRat *)

procedure SubRat(a, b : ratType); (* subtract b from a and place *)
var c : ratType; (* result in c, a-b = c *)
begin
  LowestTerms(a); (* reduce a to lowest terms *)
  LowestTerms(b); (* reduce b to lowest terms *)
  ComDenom(a, b); (* convert a & b to common denominator *)
  c.n := a.n - b.n; (* in c, a-b *)
  c.d := a.d; (* c to lowest terms *)
end; (* SubRat *)

procedure MultRat(a, b : ratType); (* multiply a and b and place *)
var c : ratType; (* result in c, a*b = c *)
begin
  LowestTerms(a); (* reduce a to lowest terms *)
  LowestTerms(b); (* reduce b to lowest terms *)
  AddRat(b, b); (* multiply numerators *)
  c.n := a.n * b.n; (* in c, a*b *)
  c.d := a.d * b.d; (* multiply denominators *)
  LowestTerms(c) (* reduce c to lowest terms *)
end; (* MultRat *)

procedure DivRat(a, b : ratType); (* divide a by b and place *)
var c : ratType; (* result in c, a/b = c *)
begin
  if b.n = 0 then (* if division by 0 then c = 0 *)
    ZeroRat(c)
  else
    begin
      LowestTerms(a); (* reduce a to lowest terms *)
      LowestTerms(b); (* reduce b to lowest terms *)
      c.n := a.n * b.d; (* invert b and multiply by *)
      c.d := a.d * b.n; (* a to get c *)
      LowestTerms(c) (* reduce c to lowest terms *)
    end;
end; (* DivRat *)

begin
end.

/****/
```
mon divisor, the greatest divisor of both a and b.

\[ 001 = 357 \cdot 2 + 287 \\
357 = 287 \cdot 1 + 70 \\
287 = 70 \cdot 4 + 7 \\
70 = 7 \cdot 10 + 0 \]

In this case, the Euclidean Algorithm applies to 1001 and 357. The last nonzero remainder is 7, so 7 is the greatest common divisor of 1001 and 357. For more on the Euclidean Algorithm, consult a book on Modern or Abstract algebra.

Two rational numbers must have the same denominator before you can add them together or find their difference. The function LCM finds the least common multiple of two numbers. The least common multiple of the denominators is the smallest number they will both divide into. This is the smallest common denominator. ComDenom uses LCM to convert two rational numbers to the same denominator.

IncRat and DecRat increment and decrement the rational number by one, respectively. ZeroRat sets the rational to zero. The numerator is zero and the denominator is one. AddRat, SubRat, MultRat and DivRat each perform the indicated operation on a and b and return the result in c.

These tools provide only the basic arithmetic for the rational numbers. Try writing a procedure such as ExpoRat that returns a rational to an exponential power. Or do operations on vectors and matrices of rational numbers. Hmm, a matrix toolkit, maybe even sparse, that’s another possible column topic.

Warning!

The unit rational does have some limitations. It doesn’t detect integer overflow or underflow. It makes an attempt to avoid these errors by calling LowestTerms before and after operations on the numbers. This unit also assumes a nonzero denominator, ratType.d, and a nonzero deviser. You can change the size and range of the rational numbers by changing the base-Type declaration. You can use the IEEE integer type comp, but all div’s must be replaced with ‘/’.

I made no provisions for output or conversion to and from real data types. As Larry Fogg says: “But I leave these as exercises for the reader. (The ultimate academic cop-out.)”

Next Time

Next issue I’ll have a Modula-2 compiler review, complete with benchmarks, compiler comparisons and programming environment summaries. So far I have one CP/M and four MS-DOS compilers, and I have a good lead on a shareware compiler. After the compiler review, I’ll cover the graphics toolkit that I’ve been wanting to work on for a long time. Then on to ... well you decide. Tune in next time for the latest episode of “Compiler Wars.” (Music fades. Lights come up.)

MODULA-2

Something about using an advanced language inspires great work. The world’s best programmers choose Modula-2. PMI publishes the best of their efforts:

* **Repertoire**: By Charles Bradford and Cole Brecheen. After five major new releases since its introduction in 1985, Repertoire is now the most mature, reliable, and widely used Modula-2 toolkit in the world. Includes unusually powerful screen design/display system patterned after MS Windows; sophisticated list-oriented DBMS; text editor; natural language analyzer; transparent EMS compatibility; and extensive string manipulation support. Object code works conveniently with any Microsoft-compatible language (protoyped headers for C included). Includes full source (over 1.2MB), and 400 pp. manual. $199.


* **Repertoire/Btrieve Toolkit**: By Gregory Higgins and SoftCraft's Brieve file manager is the standard for large business applications. R/BT is a massive support system for building Btrieve applications with Repertoire's screen system. Includes a customizable customer-tracking application. Ideal for consultants. Includes all of Repertoire's object code (but not its source) and full source for R/BT modules. $149.

* **EmsStorage**: By Charles Bradford and Cole Brecheen. Primitive EMS systems can't allocate chunks smaller than 16K; EmsStorage is a handle-oriented, high-level subsystem that manages objects as small as 1 byte. Detects and uses LIM Expanded Memory if present, or DOS memory if not. Provides automatic garbage collection. Includes full source code. $99.

* **NetMod**: By Donald Dumitru. Makes it easy to take advantage of Novell's NetWare operating system for local-area networks. Provides simple, efficient access to every important function of Advanced NetWare 2.0. Includes thorough documentation and full source code. $99.

* **Macro**: By Kurt Welgehausen. Brings the full power of C's macro preprocessor to Modula-2; provides DEFINE, UNDEFINE, IFDEF, IFNDEF, INCLUDE, etc., for parameterized macro functions, conditional compilation, etc. Includes full source. $99.


* **ModBase**: By Donald Fletcher & John McGosgle. A B-Tree DBMS that uses data file format compatible with dBase III. Release 2.0 is four times faster and includes all new manual and full source. $39.

Supported compilers: JPI TopSpeed, Logitech, Stony Brook, FST, FTL, etc.

Overseas shipping: $15.

All products available exclusively from PMI; dealer inquiries welcome.

PMI

VISA/MC
AMEX/COD/PO
Telex: 6902091013
4536 SE 50th
Portland, OR 97206
Tel: (503) 777-8844
Fax: (503) 777-0934
help, to donate their time so we’ll have another one. “I just attended my first SOG, people don’t know what they’re missing, it’s like nothing else.” .. “I’ll help, though I’m not sure what I’d do from here.”

Distribute SOG. We might make it like the SOG of old, only spread out. That way a club (or even an individual) could help by holding a smaller, more informal SOG. Ideally we’d wind up with three to five regional events .. Northwest, Southwest, Northeast, Southeast, and Middle-east (Midwest?).

If you’ve always dreamed of being surrounded by techies and you know of a wonderful campground, assembly hall, mountain top, farmer’s field, picturesque community, seacoast, or whatever — call.

By talking we might both decide your place is crazy or it’s perfect. Hopefully, we’ll come up with locations people would enjoy visiting. Places where we can gather for two or three days of laid-back babble and technology, places I can write about in Micro C.

Anyway, call with your suggestions. And call soon. (I hope it’s not too late this year to pull it off.) Anyway, we’ll announce what we come up with in Micro C: times, dates, what people need to bring (sleeping bags or credit cards), who to contact, what to expect, lodging options and prices ..

If possible, I’ll attend them all. After all, I haven’t missed a SOG yet. (The last three weeks in July, September, or November would work best for me.)

Send Us Your Tech Tips

As strange as it might seem, Tech Tips is the best read portion of Micro C. (Yep, that tiny little column that’s written by you.)

Unfortunately, each issue we scratch for good tips. I’d like the column to be the longest (okay, the second-longest) in Micro C. To do that we need your help.

The biggest question: what makes a good tech tip? Send us things you’ve discovered about: servicing PCs, putting together new hardware, locating cheap parts, surprises you’ve found in common assemblers or compilers, and things you do to make your work easier.

Send them to:

Tech Tips
Micro C Dept. Of Information
P.O. Box 223
Bend, OR 97709

Send disk, paper, or both. Or leave a file or message on the Micro C BBS, (503) 382-7643.

You’ll be glad you did, because Tech Tip authors get: three extra copies of the magazine (at least one of those goes to your mother), and an author’s T-shirt. (Non-authors have resorted to bribery in their attempts to get one of these exclusive chest covers.)

Theme Issues Can Also Be Interesting

After announcing the themes of upcoming issues, we’ve gotten wonderful response from folks working in those areas. For instance, an automated milling machine just offered to do a piece for the robotics issue (it’s a cousin of his who welds underbodies for GM). On the other hand, a number of you are waiting for a theme issue that fits your expertise.

Don’t wait.

We’re not limited to theme articles in theme issues. One of our strengths is our mix of the weird and the arcane. So, for instance, we have hardware articles in this (software tools) issue. And, of course, we ran the “Assembler Is Better Than C” article in the C issue. (Boy, did that generate mail.)

Anyway, if you have ideas, call us. Somebody’s got to make these theme issues interesting.

The Theme Issues Are Already Interesting

Robotics is going to be hot. I mean HOT. I’m talking front burner. Loren Heiny is working on an article on visual recognition, and Bob Namsel is finishing up a cheap ($300) robot complete with an ME (Mail-in EPROM). Based on the 64180 (Z80 look-alike), it’ll be a maze runner complete with IR range sensors, bumper switches, solid state compass, and so on.

He’s proposing that groups build a single robot and have members design and burn ROMS. Each group will hold competitions as members try out their own ROM programming prowess.

Then, the top ROMs from each group would be sent to Bob for the national runoff (amazing). The best ROM wins.

Official Programmer’s Diet

If you’ve been programming any length of time (i.e. since midnight, last Friday) you know the importance of maintaining your health. So you carefully plan your diet around the seven hacker food groups — twinkies, cola, coffee, chips, chocolate, cold pizza, and Rolaid.

However, I’ve got to pass along some of what I’m hearing in Aviation Consumer, a small (but expensive) newsletter on flying.

Pilots have to watch their health. They’re not supposed to have seizures, fainting spells, heart attacks, migraines, shortness of breath and such while they’re flying. Distracts from the job. Every year or two (depending on type of license), each pilot gets a physical and during the exam he gets checked for a whole list of conditions, any of which would send him to the unemployment office.

We all know about chocolate, chips, and pizza. They’re good artery pluggers. But that’s pretty slow and painless. Now, it appears, some pilots are having trouble with NutraSweet, the sugary tasting stuff residing in just about every “reduced calorie” product on the market.

The problem came to light when a pilot wrote in saying he’d spent several years and about $40,000 being tested. He’d had a whole list of problems (including seizures) that had ended his flying. He was a mess. Almost by accident he stopped drinking six bottles of diet cola a day and stopped using all other diet foods. In a week he was a new man.

Well, a NutraSweet official responded immediately, saying this person’s experience wasn’t scientific proof of a problem with their sweetener.

Then a physician followed up with a long treatise on the chemical makeup of NutraSweet and the possibility of allergic reactions. Another physician added a look at the effects of large doses of caffeine. (Many of the pilot’s complaints matched the side effects of caffeine.)

Meanwhile, another pilot who’d been having similar health
problems reported he’d stopped using dietetic gum and candy after reading the original letter. He reported that his problems, too, disappeared after a week.

So, if you’re weirder than you’d like to be (or weirder than someone close to you would like), you might want to modify your diet slightly. (Chitlins and sprouts with salsa?)

Hard Drive Relief

"Hello, Micro C?"

Yep.

"My 20 meg hard drive hasn’t been booting dependably."

Yep.

"Should I get a new drive?"

Yep.

Actually you probably don’t need a new drive — you just need to give the drive you have a little help. Most drive problems show up after a year or so, and usually it’s because the unit’s having a harder and harder time reading data. (Usually the head’s in one place, the track’s in another.) The only way to straighten things out is to do a real, low level, reformat.

Unfortunately low level formats are a pain. You back everything off the hard drive onto ten billion little floppies, run DEBUG to get into the controller’s formatter, run FDISK, run FORMAT, and then copy all that data back onto the hard drive. I’m talking hours, assuming everything works properly.

In issue #42 I wrote an article titled “Keeping Your Hard Drives Running.” In it I mentioned a package called Disk Technician. Disk Technician watches for recoverable data errors. (Errors which MS-DOS ignores.)

When the package finds a track with errors, it stores the corrected information in memory, reformats the track, and then rewrites the data. You’re supposed to run the program every day and over a year you’ll probably get a pretty complete low level reformat (as it’s needed).

I had two problems with Disk Technician. It was copy protected, and it worked best when run everyday. Also, it was designed for people who don’t want any choice about what happens (micro-power users). (I should add that I just received a new copy of Disk Technician and they’ve upgraded it a bit, made it run on more machines, and they’ve removed the copy protection.)

Anyways, in that article I also talked about H-Test and H-Format. H-Format tests your hard drive for optimum interleave and then does the optimum low-level format without wiping out data. Wonderful package.

SpinRite

A month ago I got a copy of SpinRite. It’s a pretty complete combination of Disk Technician and H-Format.

It’ll tell you the optimum interleave and do a low level reformat using the interleave of your choice (without wiping out data). It’ll thoroughly test your hard drive surface and (if you tell it to) even recover tracks that you locked out during the original format. Plus, it’ll recover data from bad sectors (if possible).

It has super displays — tells you a lot about your hard drive — is trivial to use — detects, displays, and fixes bit errors (if possible) — and it runs fast. It’s hard drive insurance and a lot more. (Plus it’s not copy protected.) Disk Technician and H-Format sell for $99.95 each so you’d pay nearly $200 for the pair. SpinRite will cost you $69.95.

One note. Use DISKCOPY to make one or more copies of the SpinRite master disk before doing anything. Then use the copy. If you ever use the master to create a working disk, you’ll only be able to use SpinRite on that system. (You’ll find the same instructions in the installation section of the manual.)

Gibson Research
22991 La Cadena
Laguna Hills, CA 92653
(714) 830-2200

Better Graphics

As long as I’m talking about things I’m excited about I might as well tell you about three new graphics packages.

Adobe Illustrator has been running on the Mac just about forever and I’d heard serious rumors (press releases) that they’d be showing their PC version at Comdex.

I looked for Adobe. No Adobe. I made a beeline for the Apple room. No Adobe. (They were there last year.) I asked the Apple staffers where I could find Adobe.

"Adobe who?"

How quickly they forget. Illustrator must have sold thousands and thousands of Macs.

However, Digital Research (you remember them, the guys who wrote CP/something) snuck in with an announcement of Artline. I don’t know Illustrator well enough to know whether Artline is better or worse, but it looks wonderful. (In the manual they show you how to draw an apple — draw your own conclusion.)

Anyway, it lets you bend type, create shadows, sketch over scanned images (you can’t edit scanned images because Artline manipulates vectors, not pixels), draw, and fill, and all those fancy things. (You have to have expanded memory to store the pixel images you’re going to trace.)

We’ve had a similar package called Designer, but Sandy and Carol find it very difficult to use.

Finally, Arts & Letters also announced their Editor package. It’s very, very similar to Artline but includes a much larger library of vector drawings.

Artline runs under Gem. Arts & Letters runs under Windows.

We’ve received both Artline and Editor and Sandy used both in this issue. Artline generated the Go board on page 96 (upper left-hand corner) and the diagram on page 37. The Arts & Letters Editor generated the books on page 95 (lower left-hand corner) and the illustrations on pages 33, 53, and 54. She had almost no trouble figuring out either program. (We have a more complete report in the works.)

(Carol used Publisher’s Paintbrush to create and edit the robot on page 95. Paintbrush works very well for editing scanned and other pixel images.)

To Your Health and Your Hard Drive,

David Thompson
Editor & Publisher
vLIB
the user interface library for C

Develop applications faster!

Use vLIB to give your programs a professional look with a sophisticated and consistent user interface!

vLIB is a comprehensive, easy to use library of over 175 custom C functions for building sophisticated PC applications.

Windows  
- overlapping, tiled, built-in window management

Menus  
- vertical, horizontal pulldown, popup, dropdown, arrays

Forms  
- full screen or popup data entry with multiple field types

Pop Ups  
- messages, prompts, selection lists, scroll bars

Mouse Support
Formatted Screen Output
Full Editing Input
High Speed Display
Color Control
... and much, much more!

Compilers supported: Microsoft C and Quick C, Borland Turbo C, WATCOM C6.5, and Lattice C. All memory models.

Library with 280 page manual . . . . . . . . . $99

Library with manual and full source code . . . . $149

Visa and MasterCard accepted. $5 shipping and handling. California residents please add 7% sales tax.

No royalties or additional fees. Site license available.

Demo disk with usable sample programs and source is available.

call or write:
Pathfinder Associates
291 Madrone Avenue
Santa Clara, CA 95051
(408) 984-2256
BBS: (408) 246-0164

Handling Interrupts With Any C, Listings

Continued from page 22

Figure 3 — ISRHT.C isr handler template.

```asm
    .segment para public 'data'
    .extrn _Dorg_:byte
    .public _ISRHT_LENGTH_, _ISRHT_PTR_
    _ISRHT_LENGTH_ dw (offset p2 - offset pl)
    _ISRHT_PTR_ dw offset pl
    dw seg pl
    .segment
    .code
    isr_template proc far
        ; flags
        ; CS
        ; IP
        push bp
        push si
        push di
        push ds
        push dx
        push ax
        push cx
        push es
        push bx
        mov ax, seg _Dorg_ ; get local data segm value
        mov ds, ax
        mov es, ax
        mov ax, 0 ; push a pointer to the ISRH
        push ax
        mov ax, 0 ; 0s will be replaced later
        push ax
        call isr_template ; isr_template also replaced
        pop ax ; restore pointer we pushed
        pop bx ; restore callers registers
        pop es
        pop cx
        pop dx
        pop di
        pop si
        pop bp
        jmp isr_template ; jump to previous handler
    isr_template endp
```

Continued on page 76
### DISK DRIVES
- **Fujitsu 360K** $69
- **Fujitsu 1.2MB** $89
- **Teac 12.4MB** $129
- **Toshiba 3.5" 720K** $89
- **Toshiba 3.5" 1.4MB** $109

### PRINTERs
- **Citizen CD 120** $149
- **Citizen CD 160** $189
- **HPLASAR Serial 1** $109
- **Epson LX-500** $219
- **Toshiba 330 XL** $519
- **20MB Hard Disk Kit** $279
- **30MB Hard Disk Kit** $329
- **ST-2095** $215
- **8425 Minicircle** $249
- **8438 30MB Minicircle** $259
- **3600 40MB Minicircle** $349
- **3675 60MB Minicircle** $379
- **ST-157 49MB 3.5"** $479

### MONITORS
- **Samsung 20"** $79
- **Samsung 21"** $89
- **NH Multisync** $59
- **NEC Multisync** $59
- **Sony Multiscan** $99
- **HGC-Monitor Card** $49
- **Color graphic card** $49
- **EGA Paradise 49"** $149
- **VGA Paradise** $279
- **Gensia Super VGA** $299

### MOUSE
- **Logitech 20"** $69
- **Logitech 21"** $69
- **Logitech 22"** $99

### PC/XT
- **McTek 286-20MHz** $499
- **Baby McTek 286-AT** $599
- **McTek 386-20MHz** $559
- **McTek 386-16MHz** $559

### MISC.
- ** scrollTop**

---

**LCD Portable**

*(Also available in PLASMA PORTABLE)*

*(Including Hard Disk Only 19 lbs.)*

-The McTek Rabbit-286 LCD Portable combines the fastest, most reliable AT motherboard available with most visible-full-size LCD portable screen on the market. Running at a switchable 8 or 10 MHz pixel rate, it includes a 20MB hard disk, 720KB 3 1/2" floppy drive, parallel & serial ports, 3.05 bios, 460k & turbo indicator LCD. The screen is a fantastically readable, electroluminescent backlit, 80-column by 25-line, high resolution 640x400 super twisted LCD with adjustable intensity and screen-angle. The screen size is 9.5"x6". It's as readable as a CRT. You can also plug in a digital or analog color monitor or a digital or composite monochrome monitor. Included also is an external 1/4" floppy port for reading and converting to 3 1/2" disks (1/4" external drive w/case: $179 when purchased with LCD Portable). The McTek Rabbit-286 LCD Portable comes fully assembled with our one-year parts & labor guarantee, and sells for an amazing complete price of only $386-20 MHz $1799! 10/19 MHz $2799!

---

**Reader Service Number 42**

MICRO CORNUCOPIA, #46, Mar-Apr 1989 75
Figure 4 — ISR.H support for interrupt handlers

```c
#ifndef uchar
#include <defs.h>
#endif

#undef OP
#include <cpu.h>
#endif

#ifndef AX
#include <call.h>
#endif

#define ISR_HDR_LOADED

/*
 * REGS - Structure of a register file.
 * A far pointer to this structure is passed to the interrupt handlers by the traps. The CPU registers are also passed in the order shown below.
 * It's therefore possible to declare a handler like:
 * void far int_24_handler( isr, r1, r2 )
 * isr 1tisr;
 * _pregs r1, r2;
 * { }
 * Which is much easier than the alternative...
 * This only works if your compiler supports passing structures by value (By now, there probably aren't many that don't...)
 */

typedef struct ISR_REGS {
    _regdp esbx;
    _reg c;
    _reg3 raxdxr;
    unsigned di;
    _regdp bpxa;
    _preg pc;
    _pregpsw pw;
} isr_regs;

/*
The code to intercept an interrupt and call a C function is copied into a structure of the following type for each handler that's needed. The required patches to the code are then made.
*isr_install() sets up the interrupt vectors, and returns. isr_restore() restores the previous handler, and isr_pass() calls the previous handler.
*The code for this trap is in ISRHT.C.
*/

typedef struct ISR {  
    uchar code[17];
    uint ishrs;
    uchar code[2];
    uint isrho;
    uchar code[2];
    void far (*hdlr)();
    uchar code[11];
    uchar codejmpf;
    void far (*prev_hdlr)();
    int isr_num;
    int count;
    void far *stack;
} isr;

#define isr_pass(1) ( (1)-codejmpf= 1-prev_hdlr ?
OP_JMDF : OP_IRET)
#define isr_iret(1) ( (1)-codejmpf= OP_IRET)
#define isr_retf(1) ( (1)-codejmpf= OP_RET)

void far *isr_get_vector(int inum);
void isr_set_vector(int inum, void far (*farfuncp)());
void isr_install(isrh *isr, int isr_num,

void far (*hdlr)());
void isr_restore(isrh *isr);

/****

Figure 5 — ISR.C support code for ISRs

#include <isr.h>

#pragma vpindex isr_get_vector
void far *isr_get_vector( inum )
int inum;
{
    return ( void far *)( (long *)( (long ) (inum * 4)));
}

#pragma vpindex isr_set_vector
void isr_set_vector( inum, farfuncp )
int inum;
{
    #asm
    mov ax,word ptr 4[bp]
    shl ax,1
    shl ax,1
    mov dx,0
    mov es,dx
    mov bx,ax
    mov dx,word ptr 8[bp]
    mov ax,word ptr 6[bp]
    cli
    mov es:word ptr 2[bx],dx
    mov es:word ptr 0[bx],ax
    sti
    #endasm
}

#pragma vpindex isr_install
void isr_install( isr, isr_num, handler )
isr *isr;
int isr_num;
void far (*hdlr)();
{
    extern int _ISRHT_LENGTH;
    extern unsigned char far _ISRHT_PTR;
    int i;

    setmem( isr, sizeof *isr, 0 );
    /* use long form of movmem() to avoid problems with the different memory models... */
    for ( i = 0; i < _ISRHT_LENGTH; i++ )
    ((char *)isr)[i] = _ISRHT_PTR[i];
    isr->hdlr = handler;
    isr->isr_num = isr_num;
    isr->isrho = ((long)((void far *)&isr)) & 0xffff;
    isr->isrho = (long)((void far *)&isr) >> 16;
    isr->prev_hdlr = isr_get_vector( isr_num );
    if ( !isr->prev_hdlr )
    isr->code jmpf = OP_IRET;

    isr_set_vector( isr_num, ( void far (*)( ))isr );
}

#pragma vpindex isr_restore
void isr_restore( isr )
isr *isr;

```
volatile long c;
volatile int k;

main()
{
  isr_isr1, isr2, isr3, isrX;
  void far int lc(), int _09(), int _23();
  int ascii;

  seg86( _REGSP );
  _regs_dump( _REGSP );
  isr_install( &isr2, &isr1, &isr3, &isrx);
  pcdos( 0x08 );
  return !_AL;
}

int conin()
{
  pcdos( 0x0b );
  return !_AL;
}

void far int_09( isr, _REGS )
isrh *isr;
isr_regs_REGS;
{
  static int ctrl;
  k = inportb( 0x60 );
  if ( k == 29 ) /* control key pressed */
    ctrl = 1;
  else if ( k == 29+0x80 ) /* control key released */
    ctrl = 0;
  else if ( ctrl 48 ( k == 46 ) || k == 70 || k == 83 )
    { /* don't allow "C", "BREAK or "DEL " */
      outportb( 0x61, inportb( 0x61 ) | 0x80 );
      outportb( 0x61, inportb( 0x61 ) & 0x7f );
      isr_int( isr );
      return;
    }
  isr_pass( isr );
}

void far int_23( isr, _REGS )
isrh *isr;
isr_regs_REGS;
{
  _carry = 0;
  isr_iret( isr );
}

/*
8086 cpu opcodes
*/
#define OP_JMPF 0xea /* far jump instruction */
#define OP_CALLF 0x9a /* far call instruction */
#define OP_RET 0xcb /* far return instruction */
#define OP_IRET 0xcf /* software int instr. */
#define OP_RETF 0x9f /* return from interrupt */

/*
CPU flag register bit masks
*/
#define FLG_RESf 0x8000 /* unused */
#define FLG_RESe 0x4000
#define FLG_RESd 0x2000
#define FLG_RESc 0x1000
#define FLG_OF 0x0800 /* overflow */
#define FLG_DIR 0x0400 /* 0=inc, 1=dec */
#define FLG_INTE 0x0200 /* Int Enable */
#define FLG_TRAP 0x0100 /* 1=one step */
#define FLG_SIGN 0x0080 /* 1=result NEG */
#define FLG_ZERO 0x0040 /* 1=result ZERO */
#define FLG_RES5 0x0020
#define FLG_AUXC 0x0010 /* Aux carry flag */
#define FLG_RES3 0x0008
#define FLG_PE 0x0004 /* 1=even # 1 bits */
#define FLG_RESE 0x0002
#define FLG_CARRY 0x0001 /* Carry Flag */

/*
Structure of the PSW register...
WARNING:

"There are a number of caveats that apply to (bit)
fields. Perhaps most significant, fields are assigned
left to right on some machines and right to left on
others, reflecting the nature of different hardware.
This means that although fields are quite useful for
maintaining internally-defined data structures, the
question of which end comes first has to be carefully
considered..."
*/
typedef struct _FLAGS {
  unsigned carry : 1;
  unsigned res1 : 1;
  unsigned pe : 1;
}
Figure 7 — REGSDUMP.C dumps a __regs structure

```c
/***/
Figure 8 — CALLER.C does far calls
```
Neural Net Models

Introductory programs with source code and the compiler used to compile them

Networkz™ A tutorial neural net model showing a simplified associative memory network. Uses DATA statements embedded in program to represent neural processing elements. "...for the price and for a flexible system, I have to recommend it." AI SIG Newsletter.

Connections: The Traveling Salesman™ A loose derivative of the classic Hopfield-Tank model with new algorithms and reduced solution time. Gives alternative approaches and bibliography. "...amazing commercially available program...Give DAIR a call." Neurocomputers.

PLD Compiler™ A fast system language compiler and its source. PLD self-compiles and has an extensive macro capability. It has been applied to writing neural net models, compilers, and an EPROM-based expert system. Review in July Computer Language.

System requirements: DOS 2.0 or above; 256K PC, AT, or compatible or PS/2. Available on 3.5 or 5.25 inch diskette. All packages include compiler object code and user manual.

To order, or for additional information, contact DAIR Computer Systems:

Reader Service Number 90

Reader Service Number 37

Networkz

Connections

PLD Compiler

Combination at Discount

Netwurkz + Connections $117.95
Netwurkz + PLD $154.95
Netwurkz + Connections + PLD $192.95

Add sales tax in California. Shipping: $5.00 in North America, $12.50 elsewhere. Orders shipped in 48 hours. Prices US funds on US bank or by credit card.

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN FOR DEC. 13, 1988

OUTSIDE OKLAHOMA: NO SALES TAX

DYNAMIC RAM

SIMM 1Mx9 80 ns $450.00
SIMM ** 1Mx9 85 ns 390.00
SIMM 256Kx9 60 ns 150.00

1Mbit 1Mx1 100 ns 33.00
12456 256Kx8 60 ns 14.95
12456 256Kx9 100 ns 25.95
51256 256Kx8 100 ns 13.50
51256 256Kx120 12.25
12456 + 64Kx4 120 ns 17.50

EPROM

27C1000 128Kx8 200 ns $29.50
27C512 64Kx8 200 ns 13.95
27256 32Kx8 150 ns 8.15
27256 16Kx8 250 ns 4.95

STATIC RAM

128K50-10 32Kx8 100 ns $22.95
1264P-12 8Kx8 120 ns 10.80

OPEN 6½ DAYS, 7:30 AM-10 PM SHIP VIA FED-EXP ON SAT.
WE EXCEPT ONLY TO CANADA, GAUM, PUERTO RICO & VIRGIN ISLANDS

SAT DELIVERY INCLUDED ON FED-EXP ORDERS RECEIVED BY:
Th 9:00 A.M. EST Fr. 1:00 P.M. PST
BEGGS, OK, 74432
(918) 267-4961

No minimum order. Please note that prices are subject to change. Shipping & insurance extra, & up to $1 for packing materials. Orders received by 9 PM CST can usually be delivered the next morning, via Federal Express Standard Air @ $5.00, or guaranteed next day Priority One @ $10.25.
Letters

might be a first. People who want a sense of perspective should read the article.

Don Taylor
13628 SE 20th Circle
Vancouver, WA 98684

SEA Defended

I normally welcome the delivery of my Micro C and read it voraciously from cover to cover. I appreciate your irreverent editorial style and make good use of the valuable information in your magazine. I feel compelled to write to you now because your irreverent style has become irresponsible. I’m talking about the Shareware column by Anthony Barcellos in issue #45.

The amount of misinformation flying around about Software Enhancement Associate’s suit against PKWare is appalling. That you would allow the type of idle, misinformed speculation that characterized Mr. Barcellos’ column is also appalling.

I don’t know all the facts surrounding the suit between SEA and PKWare, but I do know that, quite properly, SEA has not entered the fray while a court action was pending. At the same time, Mr. Katz has apparently mounted quite an effort to try his case on the bulletin boards rather than in the courts.

The issue, as I understand it, is not whether Mr. Katz has developed a more efficient algorithm — it is one of theft, pure and simple. Mr. Katz was alleged to have stolen large amounts of copyrighted code from SEA without signing a licensing agreement.

I say allege because no one will ever really know what the truth of the matter is in this case. SEA (which, incidentally, is not a “much larger” entity than PKWare, as Mr. Barcellos claims) had an expert witness examine source code from ARC and from the PK series of programs.

The expert witness testified in court that substantial portions of the PK code, portions that had not, apparently, been released to the public, appeared to be identical to SEA’s code. It was just after this witness had testified that, as I understand the situation, Mr. Katz approached SEA’s lawyers and asked for an out of court settlement.

Again, as I understand it, it was Mr. Katz who insisted on language absolving him of any wrongdoing and who also insisted that portions of the settlement be sealed. It is indeed unfortunate that SEA is not a larger company with the resources to see the suit through to its bitter end. That way I would imagine a lot of questions, both legal and otherwise, would have been settled by the public record.

All of this, of course, is speculation. I would, however, claim that SEA has performed a great service to the microcomputing community. The concept of self-compressing archives is ingenious and has taken the bulletin board scene by storm. SEA is attempting to maintain a standard — one which will, eventually, operate on many different computers and operating systems.

Is SEA hogging the market? I would say no. SEA asks for no registration or money from private users. PKWare does. SEA publishes their source code so that programmers wishing to build utilities consistent with SEA algorithms may do so. PKWare does not.

Quite rightly, SEA asks for a licensing agreement to use that code in a commercial program. I have no problems with this — it is, after all, their intellectual property. In short, I feel that PKWare is the company that has done a grave disservice to the DOS community.

Further, I feel that responsible publications such as yours and PC World should go out of their way to cover both sides of an explosive issue such as this one fairly and evenly. Put the shoe on the other foot, Mr. Thompson. If I were to take your magazine and resell it with a different cover, one which was perhaps more appealing to the mass of appliance users out there, wouldn’t I hear from your attorneys rather quickly?

I would hope in the interest of good journalism that you would cover issues more fairly in the future, or not at all.

Norman C. Saunders
The Osprey’s Nest BBS
(301) 989-9036

Editor’s note: Thank you for writing, Norm. I understand that SEA will be making a public statement now that things have been settled legally. (The statement should make the next issue of Micro C.) And, on the other side, I have contacted the PK folks for additional information on Tony’s column. So far, they haven’t responded.

Maxi/PC PCB Update

Since I wrote “Choosing a PCB Layout System” in Micro C issue #45, I’ve gotten some experience with the Racal Redac Maxi/PC PCB layout package. To fix the problems that I was having with the software protection device (key), I had to purchase a second serial I/O card and dedicate it to the key. This was a major annoyance at the time and delayed me about a week from getting started with the package.

As a test for Maxi/PC, I entered the PD32 circuit. For you newcomers, a PD32 is a 32016 coprocessor board that plugs into an XT bus and runs UNIX. The board and software were featured at SOG V and in the October/November 1986 issue of Micro C.

The PD32 was designed by George Sclaro and the firmware and software were written by Dave Rand. The hardware design was donated to the public domain; the software port and UNIX, of course, were not.

The PCB is a full size XT board with 22 ICs and 8 256x8 SIPRAMs (for a total of 2 Megabytes on-board). It’s a moderately dense layout and a good test for an autorouter.

I placed the ICs more or less in the same locations George chose, but moved a few of the smaller ICs to locations which gave a better looking ratsnest. Before autorouting, I allowed Maxi/PC to swap pins and gates. The PALs, in particular, were pin-swapped quite extensively. Therefore, the board is not wired exactly the same as George’s board, but hopefully it’s equivalent.

The results? Well, there’s good and bad. On the good side: Maxi/PC routed all but 25 of the 662 connections on the board, and accomplished this in about 20 minutes on a 10 MHz AT clone with a 10 MHz 80287 math coprocessor.

However, those last 25 connections took me about eight hours. This is because a lot of the autorouted traces had to be rerouted since the autorouter had painted itself into a corner. Perhaps a rip-up and rety router could have completed the job.

I really have to give George a lot of credit. His layout had only about 75 vias and his traces follow data and
address paths in a very regular pattern. His traces flow in vertical, horizontal and diagonal directions on both sides of the board.

The Maxi/PC autorouter, on the other hand, uses an orthogonal router. Any time it wants to change direction, it drops in a via and continues on the other side of the board. Vertical traces are on one side of the board and horizontal traces are on the other side. As a result, the autorouter generated 277 vias!

Why is this important? Mainly, the issue is cost. There is a per-hole cost when fabricating a PCB. So the more vias, the more expensive the board will be to produce. For low volume production, the extra vias may be acceptable.

The PD32 project was one which generated a lot of excitement at the time, but never really got off the ground. I think this was partly due to the cost of building a system.

The parts cost for a 2 Megabyte board was about $400 (most of the cost in the memory chips) and the UNIX single user licence was another $500. This is cheap for a UNIX system, but apparently it was too expensive for the hobbyist market at which it was aimed.

At this point I have no plans to fabricate the PCB layout described above. However, I would like to revive the spirit of the PD32 and I'd like to correspond with anyone who has any ideas for a project like this.

Scott Baker
18185 West Union Rd.
Portland, OR 97229

Editor's note: Scott and I talked for quite a while about the possibilities for a high-performance public domain UNIX system. I suspect that a 68020 or 68030-based board with space for 4 megs of RAM would make a fine multi-user UNIX platform. There are several variations of UNIX, including some free versions. Contact Scott if you're itching to do an operating system port for a new piece of hardware.

Micro Museum
I just read my first issue of Micro Cornucopia and am impressed almost beyond words! Finally, a magazine that's not afraid to talk seriously about homebrew hardware and with scarcely a word on the latest and greatest software for non-technical users. Be forewarned — most computer magazines I have liked well enough to subscribe to have failed within 90 days. May you be spared this curse.

Several months ago I counted noses and found I owned 27 computers. Shortly thereafter I realized that my self-assigned task in life is to save as much hardware and documentation as possible defining the early years (pre-8086/32000) of the "home" computer.

To date I've added a number of units and hundreds of books and periodicals. I'm sufficiently serious about this to have purchased land for a museum I hope to build in 1990/1991. I can't believe the collection will be complete without all back issues of Micro Cornucopia.

Perhaps you have readers who would like the gift of immortality. I would be most grateful for help with my collection. The more obsolete, underpowered, and obscure the computer the better.

I'm especially interested in homebrew, kitbuilt or hacked commercial systems — preferably accompanied by something of the unit's history and the owner's experiences. Obviously, I also have an interest in periodicals and books from the Golden Era. Can anyone help?

Vernon L. Goodwin
7917 Douglas Dr.
Charlotte, NC 28217

Shipping
c/o Support Systems International
8700 Suite K Red Oak Blvd.
Charlotte, NC 28217

Editor's note: Find a large lake on a windy day and look at what's tethering the boats. At the end of those waterlogged ropes you'll find 8" drives, linear power supplies, and 5-100 mother boards. Best of luck with your search.

By the way, I'm looking for a few good copies of Micro C issue #27 myself. It turns out I didn't put any aside before they sold out.

* * *

Rack & Desk PC/AT Chassis

Rack & Desk Models
Accepts PC, XT, AT Motherboards and Passive Backplanes
Doesn't Look Like IBM
Rugged, Modular Construction
Excellent Air Flow & Cooling
Optional Card Cage Fan
Designed to meet FCC
204 Watt Supply, UL Recognized
145W & 85W also available
Reasonably Priced

Integrand's new Chassis/System is not another IBM mechanical and electrical clone. An entirely fresh packaging design approach has been taken using modular construction. At present, over 40 optional stock modules allow you to customize our standard chassis to nearly any requirement. Integrand offers high quality, advanced design hardware along with applications and technical support all at prices competitive with imports. Why settle for less?
**Who’s Alive In CP/M?**

**Part II**

By Samuel Vincent

CP/M Special Interest Group
Pittsburgh Area Comp. Club
P.O. Box 6440
Pittsburgh, PA 15212

---

**Last issue we presented the first part of this list of active CP/M commercial software vendors (compiled by Samuel and his friends at PACC). Read on for the rest of that list along with public domain software suppliers and Z-Nodes that sell CP/M products.**

As before, a complete version (with prices when available) lives on the Micro CBBS and the Issue #46 disk. Please send any corrections or additions to this list by writing to the attention of Samuel Vincent at the address shown above. You can also phone at (412) 845-8613 (evenings only).

---

**Emerald Microwave**
P.O. Box 1726
Beaverton, OR 97075
(503) 641-0347
* Verified 8/31/88 *

**Everton Publishers**
P.O. Box 368
Logan, UT 84321
(800) 453-2707
(801) 752-6022
* Verified 8/15/88 *

**EZ Systems, Inc.**
P.O. Box 23190
Nashville, TN 37202
(615) 269-6428
* Verified 8/31/88 *

**Flashlite Software**
P.O. Box 3535
Daly City, CA 94015

**FreshWare Creative Software**
522 Glenpark Dr.
Nashville, TN 37217
(615) 360-6181
* Verified 8/31/88 *

**Future Communications**
128 Midway Rd.
Decatur, GA 30030-4429
(404) 373-4831

**HURD Computer Systems**
6330 Lincoln Ave.
Cypree, CA 90630
(714) 528-0879
* Verified 7/29/88 *

**Hoyle & Hoyle Software, Inc.**
111 Sparrow Dr.
Isle of Palms, SC 29451
(803) 886-5802
* Verified 8/15/88 *

**Irata Software**
2562 East Glade
Mesa, AZ 85204

**James River Group**
125 North First St.
Minneapolis, MN 55401
(612) 339-2521
* Verified 4/15/88 *

**Kamasoft**
P.O. Box 5549
Aloha, OR 97007
(503) 649-3765
* Verified 8/15/88 *

**L/Tek, Inc.**
P.O. Box 2399
Davis, CA 95617
(916) 758-3630
* Verified 8/15/88 *

**Lexisoft, Inc.**
P.O. Box 379
Alburg, VT 05440
(514) 933-4918
* Verified 4/15/88 *

**Logic Associates**
1433 W. Thome Ave.
Chicago, IL 60660
(312) 274-0531
* Verified 8/31/88 *

**HURD**

**Emerald Microwave**

**Everton Publishers**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Emerald Microwave**

**Em
<table>
<thead>
<tr>
<th>Company</th>
<th>Address/Address</th>
<th>Phone Numbers</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnolia Microsystems</td>
<td>2818 Throbyke Ave. West, Seattle, WA 98199</td>
<td>(206) 285-7266</td>
<td>Dealer-Health/Zenith software/hardware add-ons. Most CP/M software being sold out. Free brochure available</td>
</tr>
<tr>
<td>Maxx Software Systems</td>
<td>One Industrial Way, Eatontown, NJ 07724</td>
<td>(201) 542-2121, (800) 221-0440 outside NJ</td>
<td>Aztec C II compiler in educational, pro, and commercial versions. Cross compilers to or from CP/M available. Free brochure available</td>
</tr>
<tr>
<td>Mendocino Software</td>
<td>P.O. Box 154, Willits, CA 95490</td>
<td>(707) 459-9130</td>
<td>EUREKA - Disk Catalog program</td>
</tr>
<tr>
<td>MicrolSolutions Inc.</td>
<td>132 West Lincoln Hwy, DeKalb, IL 60115</td>
<td>(815) 756-3411</td>
<td>UNIFORM - Disk conversion program. UniDOS - Run CP/M programs on MS-DOS. Free brochure available</td>
</tr>
<tr>
<td>MicroPro</td>
<td>Update Order Department, P.O. Box 7079</td>
<td>(800) 227-5609 ext. 761</td>
<td>WordStar Version 4.0 CP/M Edition</td>
</tr>
<tr>
<td>MIX Software</td>
<td>1132 Commerce Dr., Richardson, TX 75081</td>
<td>(214) 763-6001</td>
<td>MIX C compiler and linker. MIX Full Screen Editor. Free brochure available</td>
</tr>
<tr>
<td>Mountain View Press, Inc.</td>
<td>P.O. Box 4656, Mountain View, CA 94040</td>
<td>(415) 961-4103</td>
<td>Forth language products Free catalog available.</td>
</tr>
<tr>
<td>NightOwl Software</td>
<td>Rt. 1 Box 7, Ft. Atkinson, WI 53538</td>
<td>(800) 648-3695, (414) 563-4013</td>
<td>MEX Plus-communication software. MEX Pack-MEX Plus with terminal emulation and remote capability. Free brochure available</td>
</tr>
<tr>
<td>O'Neill Software</td>
<td>P.O. Box 26111, San Francisco, CA 94126</td>
<td>(415) 398-2255</td>
<td>ELECTRA FIND - Text retrieval.</td>
</tr>
<tr>
<td>Paradigm Consultants</td>
<td>39243 Liberty St., Suite L, Fremont, CA 94538</td>
<td>(415) 794-8977</td>
<td></td>
</tr>
<tr>
<td>Plu*Perfect Systems</td>
<td>P.O. Box 149, Idylwild, CA 92349</td>
<td>(800) 794-8977</td>
<td></td>
</tr>
<tr>
<td>Poor Person Software</td>
<td>3721 Starr King Circle, Palo Alto, CA 94306</td>
<td>(415) 493-9735</td>
<td>WRITE HAND MAN-Resident desk accessory program Note: Doesn't work with Wordstar v4.0 - does with earlier versions.</td>
</tr>
<tr>
<td>R &amp; L Micro Services</td>
<td>P.O. Box 15955, Station F, Ottawa, Ontario, Canada, K2C 358</td>
<td>(613) 225-7904</td>
<td>BOBCAT - Disk cataloging program. Free brochure available.</td>
</tr>
<tr>
<td>SLR Systems</td>
<td>1622 N. Main St., Butler, PA 16001</td>
<td>(800) 833-3061</td>
<td>Z80 assemblers, linkers, librarian, and dis-assembler-full screen. CP/M software emulators and co-processor cards. Free catalog available.</td>
</tr>
<tr>
<td>Software Toolworks</td>
<td>13557 Ventura Boulevard, Sherman Oaks, CA 91423</td>
<td>(800) 223-8665 or (818) 907-6789 (Customer service)</td>
<td>C. Lisp, games, utilis, spreadsheet,editors etc Free catalog available.</td>
</tr>
<tr>
<td>Somogyi Software</td>
<td>P.O. Box 1009, Redondo Beach, CA 90278</td>
<td>(213) 318-2769</td>
<td>PUSH 'N' PULL - Outline processor. Version for Osborne</td>
</tr>
</tbody>
</table>

**Microcornucopia, #46, Mar-Apr, 1989**
The following companies and user groups distribute Public Domain Software.

**Boston CP/M User Group**

c/o Susan Schuckebier
Forysth Dental Center
140 Fenway
Boston, MA 02115
(617) 262-5200 ext. 342
* Verified 4/15/88 *

**The C Users' Group**
P.O. Box 97
McPherson, KS 67460
(316) 241-1065
* Verified 6/15/88 *

**Canada Remote Systems**
4198 Dundas St. West
Toronto, Ontario
Canada M8X 1Y6
(416) 231-2383 (weekdays 5pm - 9pm)
* Verified 4/15/88 *

**Comal User's Group, USA, Ltd.**
6041 Monona Drive
Madison, WI 53716
(608) 222-4432
* Verified 4/15/88 *

**FOG, International Computer User Group**
P.O. Box 3474
Daly City, CA 94015
(415) 755-2000
* Verified 8/15/88 *

**INCA**
1249 Downing St.
P.O. Box 759
Imperial Beach, CA 92032
(619) 224-1177
* Verified 4/15/88 *

**International Software Library**
c/o U.S. Computer Supply
511-104 Encinitas Blvd.
Encinitas, CA 92024
(800) 669-2699
(619) 942-1627 (Customer service)
* Verified 8/15/88 *

**M & T Books**
501 Galveston Dr.
Redwood City, CA 94063
(800) 356-2002
(800) 356-2002, in CA.
* Verified 6/15/88 *

**Micro Cornucopia**
P.O. Box 223
Bend, OR 97709
(503) 382-8048
(503) 382-9643 (BBS - 300/1200/2400 bps)
* Verified 8/15/88 *

---

**Spite Software**
4004 SW Barbur Blvd.
Portland, OR 97201
(800) 257-9111
(503) 228-6238
* Verified 8/15/88 *

**StatSoft Inc.**
2832 East 10th St., #4
Tulsa, OK 74104
(918) 583-1419
* Verified 4/15/88 *

**SWC**
P.O. Box 706
Santa Teresa, NM 88008
(505) 286-0999
(505) 286-2345 - at 2nd dial tone, dial 792
* Verified 7/29/88 *

**T/Maker Research Company**
812 Pollaid Rd, Suite 8
Los Gatos, CA 95038
(408) 666-0127
* Verified 7/29/88 *

**The Software Store**
706 Chippewa Square
Marquette, MI 49855
(906) 228-7622
* Verified 8/15/88 *

**Workman & Associates**
1925 E. Mountain St.
Pasadena, CA 91104
(618) 791-7979
* Verified 8/15/88 *

**Worwick Industries**
4898 Ronson Ct.
Suit H
San Diego, CA 92111
(619) 571-5400
* Verified 7/29/88 *

**Xpert Software**
8865 Polland Ave.
San Diego, CA 92123
(619) 268-0112
* Verified 4/15/88 *

**Zedcor Inc.**
4500 East Speedway, Ste 22
Tucson, AZ 85712
(800) 822-4967
(602) 795-3996
* Verified 8/31/88 *

---

**Desk accessory, Sideways printing,**
**MEDIA MASTER, REMBRANDT,**  
**Business planner.**

**An eclectic assortment**
**of software/hardware**
**products for CP/M.**
**Free catalog available.**

**Free brochure available.**

**T/MAKER-Integrated**
**software/word processor**
**database, spreadsheet.**
**Also avail for TRS-DOS.**
**Free brochure available.**

**Communication, disk manager, disk editor,**
**integrated data manager, program cross-reference generator.**
**Free brochure available.**

**FITL Modula-2, WRITE word processor, BBS C, MITTE-80 telecommunications,**
**POURNELLE ACCOUNTING.**
**Free catalog available.**

**Osborne software w/docs Personal Pearl,**
**SuperCalc 1 and 2, dBASE II, Forth,**
**most software by Spectre Technologies**

**XTRAKEY - Key redef**
**XTRAPRINT-Printer fonts**
**SIDE 2 - Print sideways**
**XSSCREEN - Copy screen to printer or file**
**ZBASIC interpreter - compiler.**
**Free brochure available.**

MANDELBROT EXPLORER 2.6

$30

Peter Garrison
1613 Altivo Way
Los Angeles, CA 90026
(213) 665 1397

16-color VGA/EGA to 800x600
Specify VGA or EGA, 1.2Mb
Overseas orders please add $4
Graphics Of A Better Sort

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Gary’s discovered presentation graphics, real presentation graphics. They’re not a bad sort of thing to discover if you’ve got presentations to make.

All too often we need to convince someone (a client, a boss, a research institution) that we have significant information about something. In business for example, we might collect data (numbers, that is) which correspond to some monetary function, such as sales of Dr. Wooly windshield wipers.

A researcher in pathology might want to show a correlation between unsanitary living conditions and the proliferation of the Bubonic plague. (Fortunately, this one we already know about.)

A programmer might want to compare the efficiencies of algorithms.

In short, the sample and show scenario pervades most of the “real world.” And, an excellent way to present data is to show it, as a graph or a picture.

Although computers have been very good at calculating, until recently they haven’t been very good at making graphs or pictures. It’s no laughing matter to PC dealers that potential customers often choose Apple Macintoshes primarily for their “better graphics.”

But PC graphics are improving and have neared parity (I believe) with the Macintosh in many applications. One reason for my enthusiasm stems from a terrific program I started using this week — SlideWrite Plus, from Advanced Graphics Software in Sunnyvale, California.

Performance Analysis

Let’s suppose you and your colleagues are analyzing the potential performance of critical functions in a large program. Today you’re comparing algorithms for sorting objects.

Many smart (and shudder, “smarter”) folks have already tackled sorting, so you won’t have to figure everything out yourselves. If you’re lucky you can draw most of your conclusions from someone else’s data, and spend your time forging ahead.

Specifically, your program needs to sort strings which represent names in a database. So you face this situation —

1. You haven’t selected a specific sort nor written it;
2. You have the results of tests of sorts (gleaned from programming books and journals);
3. You don’t know exactly how many names you’ll be sorting at a time, but you know the range;
4. You don’t know what kind of order the names will be in. You can probably manipulate the order, if you become convinced there’s a need for it. That is, you can maintain the names either randomly, in order, or possibly somewhere between, but it means revising your thinking about some other parts of the program;
5. The program must be portable, which doesn’t rule out assembly language techniques, but encourages you to find a solution in a high-level language;
6. You’ve designed a prototype, but you haven’t gotten into the programming yet.

Unfortunately, you find that the data you’ve collected on sorts have a rather relative nature.

Some of the tests were written in C, some in Pascal, some in Prolog. Some use iteration; some recurse. The codes for Quicksorts, insertion sorts, bubble sorts, tree sorts, and variations of all sorts are confusing. At this point, you want to compare ideas first; then (assuming you get the approval of higher ups) you’ll give yourself a chance to write your own blazing-fast sorting functions.

What to do? What to do? One benchmark, for instance, doesn’t tell you what kind of computer he used. No PC, XT, AT mention. No clock speed.

You think you see a trend, but you’re not sure you could convince anybody. Maybe you could with a picture.

Maybe SlideWrite Plus could help.

SlideWrite Plus

SlideWrite Plus does presentation graphics. In other words, you can present data as: bar graphs; charts; scatter, line, and area plots; pies; etc.
Performance and versatility for your CP/M or MS-DOS computer

**QP/M**

QP/M by MICROCode Consulting
Fed up with the message “BDOS error: RI07”? With QP/M, you’ll never lose another file because you changed a diskette. QP/M offers full CP/M 2.2 compatibility with outstanding performance and more commands WITHOUT eating up precious program space. Get such features as automatic disk relogging, simple drive/sector selection using either a colon or semi-colon, 31 user areas, drive search path, multiple program command line, archive bit maintenance, and transparent time/date stamping; all in the same space as CP/M 2.2. Installs from a convenient customization menu, no software stage required. Bootable disks available with CBIOs for Kaypro, Xerox (8” or 5 1/4”, -1 or -2), & BBL. QP/M Operating System, bootable - specify system. ...$ 64.95 QP/M without CBIOs (installs on any Z80 system) ...$ 49.95

**Networks**

QP/M Network File System by MICROCode Consulting QP/M Network File System is an efficient local area network allowing up to seven CP/M computers to share peripherals and data resources.
- Transparent operation at speeds up to 11,000 bytes/second in synchronous mode.
- Speeds of up to 1,200 bytes/second in asynchronous mode.
- Local remote disk drive and printer support.
- Remote peripheral support for modems and printer interface.
- Provides automatic screen blanking (not avail. on 83 series).
- Full automatic disk relogging with QP/M.
- Internal time-clock support.
- No software assembler command.
- Includes manual, format, configuration, diagnostig, syngen, diskette customization utility, and hard disk utilities. Available.

**Hard Disks**

Need more speed and storage on your system? Improve the productivity of your Z80 computer with a hard disk.

**KAYPRO**

KayPRO 820 - Two hard disk drives at once.
- Mix 8" and 5 1/4" at the same time.
- Built-in drives for most serial and parallel printers.
- Get multi-monitor functions and auto-boot capability.
- 16 built-in disk formats, including Xerox and Kaypro.
- Includes disk custom format definition program.
- Banked ROM BIOS for more space in your TPA.
- Composita video adaptor on X120 board.
- Runs Empyrean floppy driveboard.
- Complete parts and repair services available.

**Xerox 820**

PLUS2 ROM and X120 Double Density Board by MICROCode Consulting and Emerald Microwave!
About had it with single density diskettes on your Xerox 820-12? Get unsurpassed versatility with our X120 Board and PLUS2 ROM package.
- Run up to four floppy drives at once.
- Mix 8" and 5 1/4" at the same time.
- Software compatible with Xerox and Kaypro.
- Drives built for most serial and parallel printers.
- Get multi-monitor functions and auto-boot capability.
- 16 built-in disk formats, including Xerox and Kaypro.
- Includes custom disk format definition program.
- Banked ROM BIOS for more space in your TPA.
- Compositor video adaptor on X120 board.
- Runs Empyrean floppy driveboard.
- Support real time clock from Z80/TC.
- Runs on the Xerox 820-1 and Big Board I.
- Both ROM and X120 board are required for operation.

**Parts and accessories for the Kaypro**

- $29.00
- $16.00
- $49.00
- $35.00
- $125.00

**Xerox 820**

- $75.00
- $65.00
- $125.00
- $75.00
- $65.00
- $25.00
- $9.00
- $25.00
- $2.50
- $35.00
- $265.00

**Frustrated because your PC can’t speak CP/M?**

UnIDOS by Micro Solutions
Run CP/M programs on your PC! Of course. UnIDOS is a memory resident program that can use the NEC 820 V20 CPU chip to actually run your favorite 8080 programs. Use UnIDOS with UnIForm-PC, and automatically switch to CP/M mode as you log on your CP/M diskette. Switch to emulation mode to run 820 code programs or for systems without a V20. UnIDOS directly converts video keyboard and emulation for Kaypro, Xerox 820, Morrow, Osborne, VT100, and eight other displays. All standard CP/M system calls are supported. Note: The NEC V20 CPU is a fast, low power, CMOS replacement for the 8086 CPU chip that includes a full 8086 instruction set as well as the standard 8086 set. Systems using an 8086 may substitute a V20 chip.

**EMERALD MICROWAVE**

P.O. Box 1726, Beaverton, OR 97075

30 day money back guarantee on all products.

**IBM PC**

CP/M, NorthStar, Macintosh, Apple II, MS-DOS, and PS/2 - Don't let incompatible diskette formats get you down, read them all with your PC.

UnIForm-PC by MicroSolutions
How often have you wished you could use your CP/M diskettes on your PC? Now you can access your CP/M disk drives and programs on your MS-DOS computer just as you would a standard MS-DOS drive. As the UnIForm-PC board is installed, you can use standard DOS commands and programs right on your original diskette without modifying or copying your files. UnIForm-PC allows you to read, write, format, and copy diskettes from over 725 CP/M and MS-DOS computers on your PC, XT, or AT. With UnIForm-PC and the CompatiCard, you can use 5¼” high density, 96TP, dual format 3½” (720/1.44 meg - PS/2), and even 8” drives.

UnIForm-PC by MicroSolutions ...$ 49.95

Uniform for Kaypro and other machines ...$ 64.95

**CompatiCard by MicroSolutions**

Meet the CompatiCard, THE universal disk drive controller card. The half inch card lets you run up to 16 disk drives (4 per CompatiCard) on your PC or XT, including standard 360K, 96H TPI, high density (1.2 meg, dual speed), 8” single or double sided (SD or DD), and dual format 3½” drives (720K/1.44 - PS/2). The combinations are almost unlimited. Comes with its own MS-DOS disk drive controller program for high capacity 3½” drives. UnIForm-PC for maximum versatility. 8” adaptor and additional cabling available.

CompatiCard Board ...$ 69.95

CompatiCard with UnIForm-PC, ...$ 82.00

CompatiCard with UnIForm-PC & high density 3½” drive ...$ 350.00

**MatchPoint-PC by MicroSolutions**

The MatchPoint-PC board for the PC XT/AT works with your standard controller card to let you read and write to NorthStar hard sector and Apple II diskettes on your PC. INCLUDES a copy of the UnIForm-PC program, as well as utilities to format disks, copy, delete, and view files on Apple DOS, PRODOS, and Apple CP/M diskettes.

MatchPoint-PC Board ...$ 169.95

**MatchMaker by MicroSolutions**

Now you can copy your Macintosh diskettes right on your PC XT/AT with the MatchMaker. Just plug your external Macintosh drive into the MatchMaker board and experience easy access to your 3½” Mac diskettes. Includes programs to read, write, initialize, and delete files on your single or double sided Mac diskettes.

MatchMaker Board ...$ 199.95

MatchMaker w/ External Mac Drive ...$ 235.00

**Frustrated because your PC can’t speak CP/M?**

UnIDOS by Micro Solutions
Run CP/M programs on your PC! Of course. UnIDOS is a memory resident program that can use the NEC V20 8080 CPU chip to actually run your favorite 8080 programs. Use UnIDOS with UnIForm-PC, and automatically switch to CP/M mode as you log on your CP/M diskette. Switch to emulation mode to run 820 code programs or for systems without a V20. UnIDOS directly converts video keyboard and emulation for Kaypro, Xerox 820, Morrow, Osborne, VT100, and eight other displays. All standard CP/M system calls are supported. Note: The NEC V20 CPU is a fast, low power, CMOS replacement for the 8086 CPU chip that includes a full 8086 instruction set as well as the standard 8086 set. Systems using an 8086 may substitute a V20 chip.

UnIDOS by MicroSolutions ...$ 64.95

UnIDOS w/UnIForm and V20-8 chip ...$ 135.00

**UnIDOS 280 ClobberBoard by MicroSolutions**

This 8 MHz Z80 half card will run your Z80 and 8060 code programs at LIGHTNING speed on your PC or AT. Functions just like the UnIDOS program, except NO V20 or emulation mode is required to run your programs. Now includes UnIForm-PC

UnIDOS 280 ClobberCard ...$ 169.95

Reader Service Number 10

MICRO CORNUCOPIA, #46, Mar-Apr 1989 87
You either enter or import data into columns, and SlideWrite shows your picture of choice. You can edit data, zoom in and out of a picture, draw on it (freehand), manipulate it, change fonts, add text and legends, and in many other ways fine tune your creation (varies, scale, orientations, rotations, and other peculiarities).

When you want hard copy, you can print, plot, or upload to a camera and shoot a roll. If you've got the right equipment (polorochrome film, a mounter), you can make overheads and slides right at your desk, or go into the overhead and slide business. In a worst case scenario, in most cities you could shoot a roll of Ektachrome and have slides the next morning.

Figure 1 shows a chart reproduced from a slide created with Image Maker (a camera) and SlideWrite Plus. Don't let the black and white reproduction disappoint you; the slide, itself, is in near-living color.

Alternately, you can export your charts to desktop publishing software (like Ventura) or to slide service bureaus. In other words, you can make a pretty impression and maybe convince a few folks.

Plots Of Sorts
Take those sorts, for example. I tested SlideWrite with the results from several sorting algorithms from two useful programming books — The C Toolbox by William Hunt and Prolog Programming In Depth by Michael Covington, et al.

Covington, et al, tested five sorts —

Insertion
Hoare's famous Quicksort (from 1962)
Difference-list Quicksort
Improved Quicksort
Treesort

Hunt tested —

Insertion
Hoare's famous Quicksort

Here's the pseudo code for the Insertion sorts —

repeat (for each element after the first)
(1) compare to sorted elements to determine where it should be inserted;
(2) move sorted elements to make room for it;
(3) insert it.
end.

and the pseudo code for Quicksort —

(1) select a test value;
(2) partition the array or list using the test value;
(3) sort the left subarray (or list);
(4) sort the right subarray (or list).

To complicate matters, I entered Hunt and Covington's data into a Paradox database (of columns), then exported it as an ASCII file. SlideWrite Plus read the ASCII file correctly and created a chart in PCX format. Finally, I zipped the PCX file via modem to Micro C where Ventura imported it directly as Figure 2. Phew.

The y-axis represents time and the x-axis the number of elements in the database or array. The greater the slope of the regression line (a power curve in each of these cases), the more time it takes the function to sort the data.

A glance at Figure 2 shows an already ordered insertion sort as by far the most efficient (the curve is far less steep). A randomly ordered insertion sort is the most inefficient with an almost vertical curve. Random or ordered quicksorts are similarly efficient (both lie between).

In this example, I tried to fit several kinds of curves (linear, polynomial, exponential, etc.) to the data before I found the best fit — a power curve. SlideWrite's statistical function (F5) informed me that the power curve was a very good fit (accounting for 99% of the variation).

You can refit and redraw very quickly until you're satisfied with a description (or picture) of the data. Then you can embellish, modify, and print the charts you like.

Details & Features
SlideWrite Plus was written in Lati-

Figure 1 — Image Maker Slide

c tice C (and utilizes a few optimized assembly language functions). It's been an ongoing project for programmer Larry Daniel (now V.P. of Research & Development) for about five years.

Larry told me that when he began the SlideWrite project, his choice of a programming language was more or less obvious. Neither Microsoft nor Borland were C contenders then.

The excellent documentation (440 ring-bound pages) and numerous features are indicative of the work that's gone into this program.

Operation —
- all functions available through menu, and most through Fkeys also

Drawing —
- object-oriented (includes a picture and symbol library)
- rotates objects
- zooms images (in and out)

Graph —
- creates bars, lines, pie, text, scatter, area, 3-D, mixed, multiple graphs on one screen, free positioning of graphs, labels, and titles on screen

Colors —
- 16

Fonts —
- 16 typefaces, scalable to any size.
- Supports laser-printer cartridge & downloadable fonts

Output —
- postscript
- HP laserjet & compatibles
- various dot matrix printer
- film recorder
- plotters

Other —
- Greek/math characters
- curve fitting
- equation plotting
- error bars
- user-controlled line thickness
- automatically senses & uses a mouse

Limits —
- 4000 data points
- 12 graphs in one drawing

Requires —
- 390K available memory

Data formats —
- TIFF
- PCX
- CGM
- ASCII
- WKS
- WK1
- WRK
- WR1

Price — $445.
It sounds like I’m impressed with this program, and that’s about right.

If you want to try out SlideWrite Plus with no commitment, call Advanced Graphics and they’ll send you a little (and very useful) manual and a SlideWrite Plus demo disk which seems to do everything the real McCoy does.

The demo is free.

For more information —

Advanced Graphics Software, Inc.
333 W. Maude Ave., Suite 105
Sunnyvale, CA 94086-4367
(408) 749-8620

A Good Place To Shop

I can’t resist a good pitch —

“Isn’t it the truth! Hardware engineers always blame the problems on software while software engineers always blame the problems on hardware. And marketing just wants to ship product.”

“Well when I told my sister (she’s a programmer) about these mugs, she said, ‘That’s exactly what they say!’ So I know these will be perfect. Buy one for yourself or give one to your favorite opponent.”

Picture two hardware and two software mugs, their handles shaped like ears, alongside mini-reviews of “The Fractal Geometry Of Nature” by Mandelbrot, and “Chaos” by the Royal Society of London.

In the same pages (28 in all), I found Escher puzzles and calendars, discussions of fuzzy math books, a consulting column, rubber stamps that reject everything, a code listing from a Build Your Own Assembler Project, OS/2, 80386, MS/DOS, and Unix programming guides.

There are “far out” opinions and things to buy.

If you want bizarre and interesting access, subscribe to John’s Picks, a “review catalogue” of books and things. Even the index is good reading! $5 per year from —

Microcomputer Applications
P.O. Box E
Suisun City, CA 94585-1050
(707) 422-1465

And that’s Tidbits.

MICRO CORNUCOPIA, #46, Mar-Apr 1989 89
Diagnosing A Slipped Disk

Disk Change Solution

Issue #45 arrived yesterday, and as usual I spent most of the evening with it. I found that the letter from John Innes of New South Wales, Australia, described some symptoms that sounded very familiar.

I'm using an AT clone with the AMI 286 BIOS. When I installed a 3.5 inch 720K drive for drive B and ran the BIOS SETUP routine, everything seemed fine. The format command worked, files could be read and written, etc. However, upon changing disks, the directory did not change. Of course, attempts to read from the disk produced garbage. I wasn't brave enough to try writing to the disk, but I would expect a write to destroy its contents.

Since I use MS-DOS 3.2 (which supports DRVVPARM in CONFIG.SYS), the solution was easy once I figured it out. It appears that the AMI SETUP utility sets up the disk parameter table to look for the door open signal. This causes a problem if the disk doesn't have a door open switch or if the switch doesn't work.

DRVVPARM is described in the DOS manual as having an option (/C) that "specifies that change-line (doorlock) support is required." What it doesn't say is that if the SETUP parameters specify change-line service, but the drive doesn't support it, then it should be disabled by using DRVVPARM without the /C option. I use:

```
DRVVPARM=/D:1
```

since the other default parameters are correct for a 720K drive. Mr. Innes should use —

```
DEVICE=DRIVER.SYS /D:0 /F:1
```

in his CONFIG.SYS file (unless his drive has a broken door-open switch, which should be fixed).

John R. Smith
609 E. Sacramento St.
Altadena, CA 91001

Editor's note: John presents a simple solution to the "phantom directory" problem. But those who just can't resist the opportunity for a little hardware hacking should read on.

... And A Hardware Version

John Innes' directory woes come from MS-DOS' method of handling disk swaps. The operating system actually reads a directory only once for a disk, storing this information in memory for subsequent DIRs. Only when a disk change is detected (through the disk-change or door-open signal from the drive) will DOS reset the drive.

Until recently, most drive controllers ignored the wishes of DOS and did a reset with each drive access. But newer controllers and drives often use signals from the drive to detect a disk change. So it's entirely possible to come up with a combination which can't detect a new disk. (^C will always reset the drives, but who wants to type endless ^Cs.)

You can easily fool the controller into resetting with each access. If it has seen any activity on the disk-change line (#2 on the ribbon cable), the controller assumes a disk change and resets accordingly. So just connect line #2 to the step line (#20) and the controller will see plenty of "disk changes." Do this by cutting line #2 and connecting the controller end to line #20.

The mod couldn't be much simpler. But if you're feeling lazy, Fujitsu and other companies sell a small adapter.

Editor's note: The preceding was gleaned from John Heilborn's "Ask Dr. John" column in Computer Currents, December, 1988, Boston Edition. The article came to us from:

Bruce S. Campbell
56 Birchwood Rd.
Windsor, CT 06095

Yet Another Hardware Fix

My 720K and 1.44M drives have XT and AT jumper options. I've found that jumpering the XT pins (even when the drive resides in an AT) solves the directory problem.

I suspect that XT disconnects pin 34. According to my documents, disconnecting pin 34 between the floppy and the controller tells the system to ignore pin 2 (the door-open line). So the system would have to reread the floppy each time it needs the directory.
Unfortunately, the disconnection might force an AT to treat the drive as a low density unit. Thus, a 1.2 meg drive would only read and write 360K. A 1.44 meg drive would only read and write 720K. (Check this on your system.)

Rob Aprato  
323 75th St. #17  
Everett WA 98203

MASM Tips  
I just read the article on Microsoft’s Macro Assembler, version 5.1. (‘‘86 World,’’ Micro C, issue # 45) I completely agree with the author’s positive comments. The new features make it a real pleasure to use. However, I’d like to mention some “gotchas.” The first problem has to do with the new LOCAL directive, which allocates stack space and generates macros for reference of local (automatic) variables. Do not try to use a DWORD as the first local variable.

That’s because the assembler always uses [BP-2] for the first local variable. This is fine for a WORD, but if you store a DWORD in [BP-2], it gets stored in [BP-2] through [BP+1], thus clobbering the saved BP at [BP]. If I need a DWORD, I declare the first local variable with the name DUMMY and the type WORD. The DWORD declaration then follows the dummy variable.

The second problem is not a bug, but a technique. In C, it’s quite easy to pass the address of a local variable to another function, so that the second function can modify the local variable.

For instance, suppose you have a routine that displays a menu. One of the first things it does is save the contents of the screen memory:

```c
show_menu() {
  /* local variable */
  char screen_buffer[4000];
  /* pass address of array */
  save_screen(screen_buffer);
  ...
}
```

I ran into problems when I translated that type of C function into assembly. Unfortunately, automatic variables are allocated in the stack segment (referred to by SS), while static variables are allocated in the data segment (referred to by DS).

If you program in the large model, there’s no problem because both segment and offset of the variable get passed. However, the small model only passes the offset and the function has no way of knowing whether the offset refers to the data segment or stack.

The only way around the problem is to make sure that SS equals DS. In high level languages, this is usually set up by the compiler. In an assembly language program, you do it yourself:

```assembly
_main:
  mov ax, @data
  mov ds, ax
  cli
  mov ss, ax
  mov sp, OFFSET STACK
 sti
  ...
```

The CLI/STI instructions should not be necessary, but a few early (buggy) versions of the 8088 failed to disable interrupts when modifying SS. Note that SS must be changed, then SP.

Richard Lamb  
3016 Waverly Dr., Apt. #311  
Los Angeles, CA 90039  
* * *
Incredibly Low Prices

1-9 Disks $29.9 ea.
10-19 Disks $24.9 ea.
20 or more $22.9 ea.

LOTTO PROPHET (2364) Best Lotto program we've seen.

LADIES FIRST 5.5 (624) Ad.?-seekers by changing spreadsheet and seeing result. (Requires Lotus)

LOTUS TUTORIAL (630) Learn Lotus (requires Lotus).

ADULTS ONLY


STRIKING POCKET (293) Pick opponent (CGA).

BAD-BAD (2904) Adventure game.

ASTRO ORION (2055) Arcade game (CGA)

X-RATED COLOR SHOW (2915) Beauties with viruses.

X-RATED PRINTSHOP (2909) Graphics for Printshop.

TELECOMMUNICATIONS

QUATTRO PRO (611, 1102, 1144) (3 disks) Powerful but easy to use. Fast.

RBS V16,1A (1107-1140, 1150) (4 disks) M.A.E. Mouser bulletin board system.


SECURITY/HACKING

COPY PROTECTION I (1219) Instructions for unprotected commercial software.

COPY PROTECTION II (1220) More software protection.

COPY PROTECTION III (1221) Adding software to unprotect.

FLUSHOT (1225) Checks software for viruses.

DATABASE PROGRAMS

PC-FILE db (601, 805, 837) (3 disks) Described as excellent and slightly better than dBase III+.

PC-GRAPH (802) Create graphics from PC FILE.

FILE EXPRESS 4.0 (803-804) Powerful system. Allows 32,000 records. Sorts up to 10 key fields.

DB III+ ROUTINES (851-852) (2 disks) Latest utilities to help you utilize dBase III+

CALL TODAY FOR FREE CATALOG

800-444-1343

Ext. 23

FOREIGN: 619-436-0130

Reader Service Number 120

MICRO CORNUCOPIA, #46, Mar-Apr, 1989 93
Is There A Gap In Your Info?

Fill in your Back Issues of Micro C today!

**Back Issues**

**ISSUE #1 (5/81)**
Power Supply RAM Protection Video Wiggle 1/2 PFM.PRN 16 pages

**ISSUE #2 (10/81)**
Parallel Print Driver Drive Motor Control Shugart Jumpers Program Storage Above PFM 1/2 PFM.PRN 16 pages

**ISSUE #3 (12/81)**
4 MHz Mods Configuring Modem 7 Safer Formatter Reverse Video Cursor VDI/VOD Words Begins 16 pages

**ISSUE #4 (2/82)**
Keyboard Translation More 4 MHz Mods Modems, Lynx, and S10s Undoing CPM ERASE Keyboard Encoder 20 pages

**ISSUE #5 (4/82)**
Word Processing Two Great Spell Two Text Editors Double Density Review Scribble, A Formatter 20 pages

**ISSUE #6 (6/82)**
BBI EPROM Programmer Customize Your Chars Double Density Update Terminal in FORTH 24 pages

**ISSUE #7 (8/82)**
6 Reviews Of C Adding 6K Of RAM Viewing 50 Hz On Your Own Begins 24 pages

**ISSUE #8 (10/82)** SOLD OUT

**ISSUE #9 (12/82)**
BBI EPROM Program Replacing Your CPM Serial Print Driver Big Board I Fixes Bringing Up WordStar Cheap RAM Disk 32 pages

**ISSUE #10 (2/83)** SOLD OUT

**ISSUE #11 (4/83)** SOLD OUT

**ISSUE #12 (6/83)**
25K for BBI Bringing Up BBI dBase II Look at WordStar Double Sided Drives for BBI Packet Radio 5 MHz for Kaypro 40 pages

**ISSUE #13 (8/83)**
CP/M Disk Directory More 256K for BBI Mini Front Panel

**Micro Cornucopia**

<table>
<thead>
<tr>
<th>Back Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ISSUE #14 (10/83)</strong> BBI Installation The Perfect Terminal Interface To Electronic Typewriter BBI Video Size Video Jitter Fix Slides Column Begins Kaypro Color Graphics Review 48 pages</td>
</tr>
<tr>
<td><strong>ISSUE #15 (12/83)</strong> Screen Dump Listing Fixing Serial Ports Playing Adventure SBASIC Column Begins Upgrading Kaypro II To 4 Upgrading Kaypro 4 To 8 48 pages</td>
</tr>
<tr>
<td><strong>ISSUE #16 (2/84)</strong> Xerox 620 Column Restarts BBI Double Density BBI 5&quot; Interface Fix Kaypro ZCPR Patch Adding Joystick To Color Graphics Recovering Text From Memory 52 pages</td>
</tr>
<tr>
<td><strong>ISSUE #17 (4/84)</strong> Voice Synthesizer 620 RAM Disk Kaypro Morse Code Interface 68000-Based System Review Inside CPM/86 56 pages</td>
</tr>
<tr>
<td><strong>ISSUE #18 (6/84)</strong> Kaypro EPROM Programmer I/O Bytes A Primer Kaypro Joystick Serial To Parallel Interface Business COBOL 60 pages</td>
</tr>
<tr>
<td><strong>ISSUE #19 (8/84)</strong> Adding Winchester To BBI 6 MHz On The BBI Bulletin Boards Tracking Buffering On Slicer 4 MHz For The 820-1 64 pages</td>
</tr>
<tr>
<td><strong>ISSUE #20 (10/84)</strong> HSC 68020 Co-Processor DynaDisk For The BBI Serial Printer On BBI Sane S10 Cheap &amp; Dirty Talker For Kaypro Extended 8&quot; Single Density 72 pages</td>
</tr>
<tr>
<td><strong>ISSUE #21 (12/84)</strong> Analog To Digital Interface Installing Turbo Pascal Low Intensity BBI Video Turbo Pascal, The Early Days 80 pages</td>
</tr>
<tr>
<td><strong>ISSUE #22 (2/85)</strong> Xerox 8010 II To A Kaypro-8 Sound Generator For the STD Bus Reviews Of 256K RAM Expansion In the Public Domain Begins 88 pages</td>
</tr>
<tr>
<td><strong>ISSUE #23 (4/85)</strong> Automatic Disk Reloading Interrupt Drive Serial Printer Low Cost EPROM Eraser Smart Video Controller Review: MicroSphere RAM Disk Future Tens Begins 86 pages</td>
</tr>
<tr>
<td><strong>ISSUE #24 (6/85)</strong> Cling Into Turbo Pascal 8&quot; Drives On The Kaypro 48 Lines On A BBI 600K Versus 80K/86 Soldering: The First Steps 88 pages</td>
</tr>
<tr>
<td><strong>ISSUE #25 (8/85)</strong> Why I Wrote A Debugger The 32-Bit Super Chips Programming The 52252 Module II RS-232C: The Interface 104 pages</td>
</tr>
<tr>
<td><strong>ISSUE #26 (10/85)</strong> Installing ZCPR Two Megabytes On DSI-32 SOC IV The Future Of Computing Graphics In Turbo Pascal 104 pages</td>
</tr>
<tr>
<td><strong>ISSUE #27 (12/85)</strong> SOLD OUT</td>
</tr>
<tr>
<td><strong>ISSUE #28 (2/86)</strong> Pascal Runoff Winners Rescuing Lost Text From Memory Introduction To Module-2 Inside The PC 104 pages</td>
</tr>
<tr>
<td><strong>ISSUE #29 (4/86)</strong> Speeding Up Your XT Importing Systems From Taiwan Prototyping In C C Interpreters Reviewed Benchmarking The PCs 104 pages</td>
</tr>
<tr>
<td><strong>ISSUE #30 (6/86)</strong> PROLOG On The PC Expert Systems Logic Programming Building Your Own Logic Analyzer 256K RAM For Your 83 Kaypro PC-DOS For Non-Clones 104 pages</td>
</tr>
<tr>
<td><strong>ISSUE #31 (8/86)</strong> RAM Resident PC Speedup Practical Programming In Module-2 Unlinking The PC's Blinkin' Cursor Game Theory In PROLOG and C 104 pages</td>
</tr>
<tr>
<td><strong>ISSUE #32 (10/86)</strong> Public Domain 32000: Hardware And Software Writing A Printer Driver for MS-DOS Recover A Directory By Reading &amp; Writing Disk Sectors 96 pages</td>
</tr>
</tbody>
</table>

**To Order:**

Phone: 1-800-888-8087
Mail: PO Box 223
Bend, Oregon 97709

**United States,**

| Issues #1-34 | $3.00 each ppd. |
| Issues #35-current | $3.95 each ppd. |

**Canada, & Mexico**

All issues | $5.00 each ppd. |

**Foreign (air mail)**

All Issues | $7.00 each ppd. |
<table>
<thead>
<tr>
<th>Advertiser</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acquired Intelligence</td>
<td>14</td>
</tr>
<tr>
<td>American Cosmotron</td>
<td>31</td>
</tr>
<tr>
<td>Austin Codeworks</td>
<td>45</td>
</tr>
<tr>
<td>Blaise Computing</td>
<td>13</td>
</tr>
<tr>
<td>Capital Software</td>
<td>20</td>
</tr>
<tr>
<td>Cascade Electronics</td>
<td>39</td>
</tr>
<tr>
<td>CC Software</td>
<td>35</td>
</tr>
<tr>
<td>C Gazette</td>
<td>59</td>
</tr>
<tr>
<td>Computerized Processing Unltd.</td>
<td>65</td>
</tr>
<tr>
<td>CompuView</td>
<td>79</td>
</tr>
<tr>
<td>Dair Computer Systems</td>
<td>90</td>
</tr>
<tr>
<td>Emerald Microware</td>
<td>87</td>
</tr>
<tr>
<td>Erac Company</td>
<td>15</td>
</tr>
<tr>
<td>Garrison, Peter</td>
<td>85</td>
</tr>
<tr>
<td>GEMS</td>
<td>67</td>
</tr>
<tr>
<td>Gibson Research</td>
<td>22</td>
</tr>
<tr>
<td>Greenleaf Software</td>
<td>7</td>
</tr>
<tr>
<td>Halted Specialties</td>
<td>23</td>
</tr>
<tr>
<td>Integrand</td>
<td>81</td>
</tr>
<tr>
<td>Manx Software</td>
<td>75</td>
</tr>
<tr>
<td>McTek Systems</td>
<td>42</td>
</tr>
<tr>
<td>Micro Cornucopia</td>
<td>94</td>
</tr>
<tr>
<td>Microprocessors Unltd.</td>
<td>79</td>
</tr>
<tr>
<td>Microsphere</td>
<td>1</td>
</tr>
<tr>
<td>Micro Star</td>
<td>93</td>
</tr>
<tr>
<td>NuMega Technologies</td>
<td>2</td>
</tr>
<tr>
<td>Pathfinder Associates</td>
<td>74</td>
</tr>
<tr>
<td>Peripheral Tech</td>
<td>35</td>
</tr>
<tr>
<td>PC Tech</td>
<td>Back Cover</td>
</tr>
<tr>
<td>PMI</td>
<td>71</td>
</tr>
<tr>
<td>Quantum Software</td>
<td>18</td>
</tr>
<tr>
<td>Research Group</td>
<td>5</td>
</tr>
<tr>
<td>RJSwantek</td>
<td>70</td>
</tr>
<tr>
<td>SemWare</td>
<td>27</td>
</tr>
<tr>
<td>SoftSolutions</td>
<td>6</td>
</tr>
<tr>
<td>Star-K</td>
<td>30</td>
</tr>
<tr>
<td>Sterling Castle</td>
<td>18</td>
</tr>
<tr>
<td>United Products</td>
<td>63</td>
</tr>
</tbody>
</table>

** Contact Advertiser Directly.

### Now Available From Micro C

**Computer Interfacing with Pascal & C**

by Bruce Eckel

- Use your PC parallel port for digital input and output
- Build an Adapter Card for your PC
- Control a stepper motor
- Design and build electronic circuits
- Control AC power
- And much, much more

"Simply the best microcomputer electronics book I've read." Larry Fogg

Only $30 ppd.
Includes Book & Disk

### Issue #47

Robotics

- Do It Yourself Maze-Running Robot
- Writing A Robotics Operating System
- Controlling Events With C++
- Alternatives In Desktop Publishing
- The Science Of Fractal Images
I remember watching the Go games in the Tektronix lunchroom. Many of the cafeteria “masters” had novices pick up their food so they could play an extra five minutes.

Often the discussion would turn to computerizing the game, but the masters always discouraged the idea. “Too complex,” they’d say. “Do something easy, like Chess.” So we’d go back to watching the boards and fetching food.

Every game has a set of rules which separates it from its surrounding reality. Most games have a simple set of rules and simple behavior or potential board configuration. But some games (the most intriguing ones) have a simple set of rules which often lead to extremely complex boards and patterns.

One of these, the Asian game “Go,” has survived 4,000 years of human, and 20 years of computer attempts to master it.

In Go, two players alternate in placing stones on a 19x19 grid. Once placed, a stone cannot be moved, but it can be removed (via capture).

You win the game by enclosing more territory than your opponent. In the process of enclosing territory, you may capture an opponent’s stones by surrounding them. Generally, the fewer stones either of you has on the board, the less territory you hold.

The rules of Go are much simpler than the rules of chess, but a Go board can be much more difficult to decipher because of the incredible number of possible board configurations (about $10^{170}$), but far less complex than in Go.

In Go, there’s only one “play” — you place a stone on an unoccupied intersection. At most (at the beginning of the game), there are 381 possible plays. Each move reduces the number of possible plays by one, while increasing the complexity of the board by a large nonlinear factor.

Although the rules of Go are simple, the staggering number of possible board configurations (or patterns) make programming a “smart” Go game very, very difficult.

Nevertheless, computer Go games are improving thanks to the joint efforts of AI researchers and “smart” programmers.

According to Walter Reitman and Bruce Wilcox, the designers and programmers of two “smart” Go games — “Skilled human Go play presumes the ability to recognize and make inferences from many different kinds of complex patterns — our Go programs use a small set of basic scanning and recognition mechanisms (rules) to deal with these patterns.”

Nemesis, Go Master

Recently, I had the pleasure of playing Bruce Wilcox’s latest version of Go — Nemesis, the Go Master (from Toyogo, Inc., in Lexington, Massachusetts).

Nemesis is a clone of POGO, created by Reitman and Wilcox on an IBM mainframe between 1972 and 1977. That Go took seven person-years and 8,000 lines of LISP to create. It required 3 megabytes of memory to run.

Bruce writes (in Reflections on Building Two Go Programs) that a single game of mainframe Go costs close to $2,000. So eventually, even on an academic budget, he had to give it up. Fortunately, a game with Nemesis on the PC isn’t so expensive.

It took Bruce one year and 13,500 lines of C to create Nemesis. Over 3,000 lines of that (almost 1/4) went into the user-interface. It takes only 146 kilobytes and a PC to play it. At $79, Nemesis is affordable (even for non-academics).

Nemesis is as good (or better) a player than its mainframe ancestor.

I’ve had a blast playing (and learning to play) this incredible game. The beauty of it is that Nemesis helps —

- Each time Nemesis places a stone, it explains its motives, associating them to a rule (or rules).
- I can ask Nemesis to help me place a stone and to explain its motives.
- I can edit the board (to a new position) and ask Nemesis to suggest moves.
- I can take back moves (and so can Nemesis).
- I can practice (or learn) by playing Nemesis on smaller boards (9 x 9 and 13 x 13).
- I can save and resume games later.

For more information —

Toyogo, Inc.
76 Bedford St., Suite #34
Lexington, MA 02173
(617) 861-0488

References

Until now, if you wanted the best Undo, the best compiler support, regular expressions and column blocks you chose BRIE®. If you wanted unlimited keystroke macros, the best configurability, "off the cuff" command language macros and blazing speed, you chose VEDIT PLUS.®

Now the Choice is Easy
The all new VEDIT PLUS 3.0 gives you the best Undo of any editor, the best compiler support, unequalled windows, true regular expressions and extensive new features. We're leading the way with easy to use pull-down menus, context sensitive help, a pop-up ASCII table, new printing options and much more. Incredibly, VEDIT PLUS 3.0 is now twice as fast as before and, at only 60K in size, it loads fast!

Completely Configurable
Change a few keys or redefine the entire keyboard, VEDIT PLUS adjusts to your editing style in minutes. You can even create new editing functions using simple keystroke macros or fine tune existing ones. VEDIT PLUS is so configurable that it easily emulates other editors and word processors (WordStar and Word Perfect emulation included). Quickly access editing functions with a single key or through the pull-down menus.

Try before You Buy
We challenge you to experience the dazzling performance and exceptional features that make VEDIT PLUS the best choice. Our evaluation disk includes the complete editor.* Learn VEDIT PLUS using our extensive "training" macro that gives instructions in one window while you experiment in another. See for yourself why no other macro language comes close.

Call for your free evaluation copy today. See why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Supports the IBM PC, XT, AT and PS/2 including DESQview, Microsoft Windows, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, CP/M-86 and FlexOS. (Yes! We support windows on CRT terminals.) $185.

Features of VEDIT PLUS 3.0
• Simultaneously edit up to 37 files of unlimited size.
• Variable sized windows; multiple windows per file.
• Execute DOS commands and other programs.
• Flexible "cut and paste" with 36 "scratch-pad" buffers.
• Block operations by line, character or column.
• Search with pattern matching or regular expressions.
• Configuration—determine your own keyboard layout, create your own editing functions, support any screen size.
• Select window colors, support 43 line EGA, 50 line VGA.

EASY TO USE
• Modern pull-down menu system. Pop-up ASCII table.
• Context sensitive on-line help is user changeable.
• Multi-level Undo (100 to 1000 levels). Undo keystroke by keystroke or line by line.
• On-line integer calculator (also algebraic expressions).
• Keystroke macros speed editing, menu function "hot keys."

FOR PROGRAMMERS
• Automatic Indent/Undent for "C," PL/I, PASCAL.
• Match/check nested parentheses, e.g. "[" and "]" for "C."
• Flexible macro runs popular compilers and automatically moves cursor to each error in your program. Easily changed to support new compilers and assemblers.

FOR WRITERS
• Word wrap, paragraph formatting and justification.
• Convert to/from Wordstar and mainframe files.
• Flexible printing; fully adjustable margins and Tab stops.

MACRO PROGRAMMING LANGUAGE
• If-then-else, looping, testing, string compare, branching, user prompts, keyboard input, 24 bit algebraic expressions.
• Flexible windowing—forms entry, select size, color, etc.
• Simplifies complex text processing, formatting, conversions.
• Extensive 400 page manual with hundreds of examples.
**VERY HIGH PERFORMANCE**

**Processors, Memory, and Display Adapters**

**The X24 High performance processor**
- 12 or 16 MHz 80286 with **NO WAIT STATES!**
- Small size ("XT" height and length) passive bus design
- 1 to 4 Mbyte 0 wait state dynamic memory
- Fully "AT" compatible Award BIOS
- Runs DOS versions 2.2 and later, Xenix and OS/2

The X24 combines the best of motherboard and backplane designs in a 100% AT compatible system. Incorporating a 16 MHz 80286, the X24 processor is designed to operate with the PC Tech Advanced System Motherboard, which contains the peripheral interfaces (hard disk, floppy disk, two serial ports and a parallel port). The X24 processor can also be used with other totally passive bus backplanes. Most critical components including the microprocessor and up to 2 megabytes of fast memory are contained on a single PC size plug-in card. This allows the processor and main system memory to be serviced or upgraded without disturbing other peripherals such as serial ports and disk drives.

**The PC Tech Advanced System Motherboard**
- Built in "IDE" interface for AT interface type hard drives
- Fully AT compatible floppy disk support for 3.5", 5.25" drives, capacities of 360k, 1.2m and 1.44m
- Two serial ports and one parallel port
- 8 total expansion slots PC/XT/AT compatible (4 slots have 32 bit bus)

The PC Tech Advanced System Motherboard is designed to complement PC Tech's X24 and X32 high performance processor cards. It contains the mass storage interfaces necessary for a complete system, plus the basic I/O required in most systems. Extra care has been given to FCC compliance by design.

**34010 Monochrome Graphics Adapter II**

```
PC Tech Mono-II
```

- Up to 384k bytes display memory
- Up to 2 Megabytes program memory
- Software is RAM based, allowing complete operating software replacement and timing re-programming from the host bus
- 34010 program loader included. Assembler, debugger, and C compiler available.
- Full hardware and software CGA, MDA and Hercules emulation
- Single bit shared memory bit-map with optional resolution up to 2048 x 1536 (736 x 1008 standard)
- Very high resolution COLOR version available
- Custom 34010 software development available

The TMS34010 is a true general purpose graphics processor. PC Tech makes the total processing power of the 34010 available to both programmers and end users. Our 34010 Monochrome Graphics Adapter is designed to allow programming from the PC/XT/AT host bus. You can completely replace our 34010 software with yours to directly harness the incredible image processing power of the TMS 34010 for your application. We make a complete set of development tools available, including an assembler, C compiler, program loader, 34010 debugger, and PC interface trace/debugger. Our standard product includes support for extended CGA, MDA and Hercules emulate as well as a host addressable graphics bit-map. We also support and recommend the DGIS graphics interface standard (from Graphic Software Systems) for applications development as an alternative to native 34010 software development. Ready to run drivers are available for most major applications software packages as well.

**Custom Designs Available**

PC Tech will license most products for non-exclusive manufacture. We will also customize any of our designs to better meet your needs on our in-house CAD systems. All of our standard products are available in private label versions.

**About PC Tech**

PC Tech has been designing, manufacturing and marketing high performance PC related products for over three years. Our standard product line includes processor, memory, and video products. All products are designed, manufactured and supported in our Lake City, Minnesota facilities.

**Designed, Sold and Serviced By:**

```
907 N. 6th St., Lake City, MN 55041
(612) 345-4555 • (612) 345-5514 (FAX)
```

PC, XT, AT, DGIS, Hercules, and Award BIOS are trademarks or registered trademarks of their respective companies.