No. 45 January-February 1989 $3.95

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Computer Aided Design

Thinking of setting up shop as a hardware designer? Schematic capture and board layout tools will not only make you more efficient, they'll really put you in demand.

And More . . .

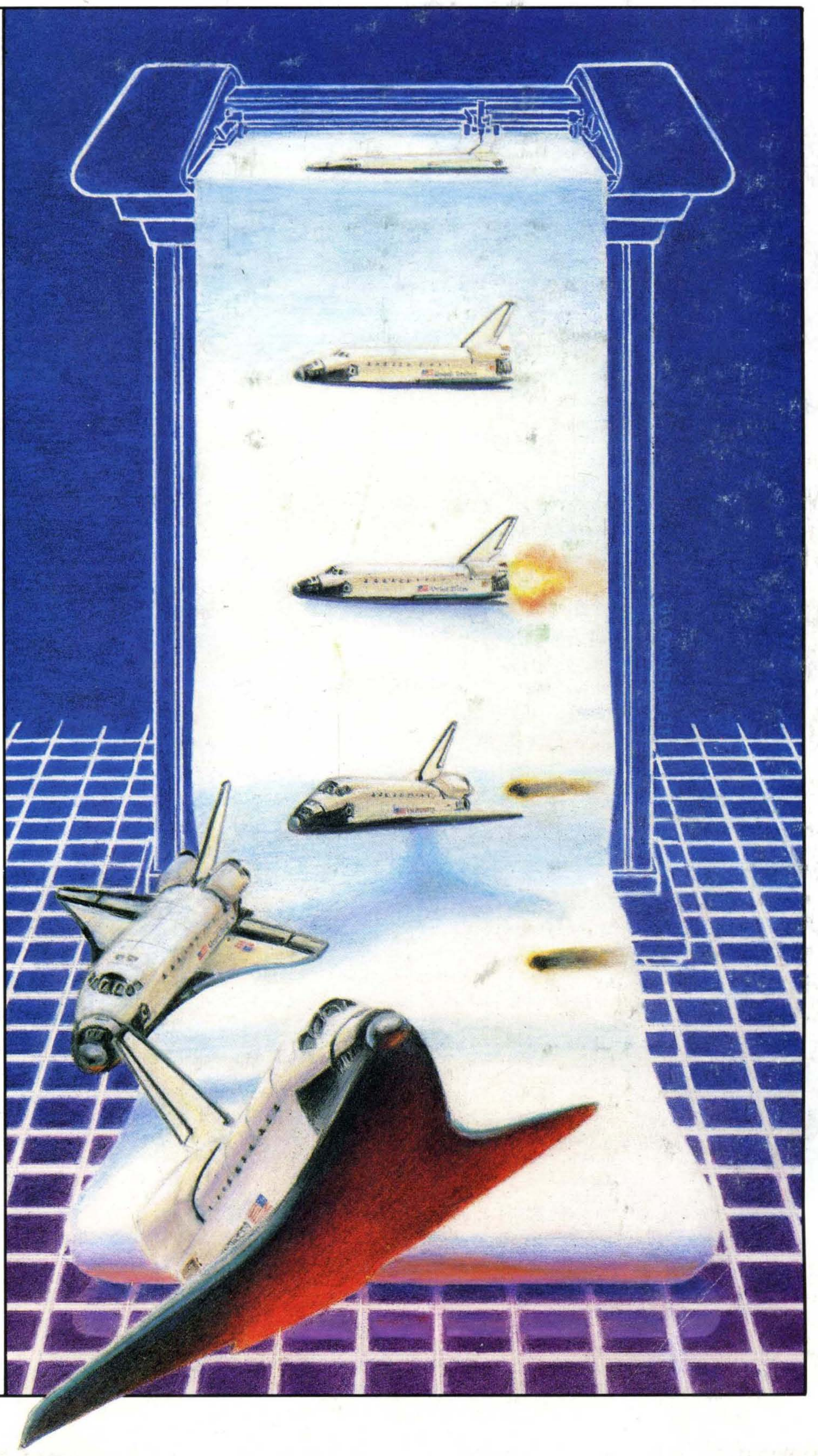
What to expect from a code optimizer (and what to watch out for).

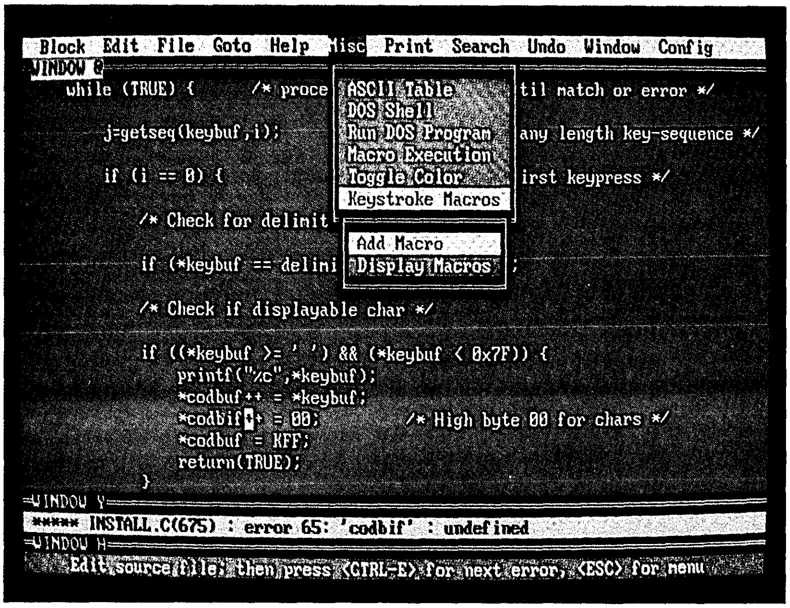
Karl Lunt tells us where he finds the real hardware bargains. What he watches for and what he watches out for.

Plus:

12

0 74470 19388 3





```
WINDOW 0
    while (TRUE) {        /* proce             til match or error */

        j=getseq(keybuf,i);                    any length key-sequence */

        if (i == 0) {                          irst keypress */

            /* Check for delimit

            if (*keybuf == delimi                ;

            /* Check if displayable char */

            if ((*keybuf >= ' ') && (*keybuf < 0x7F)) {
                printf("%c",*keybuf);
                *codbuf++ = *keybuf;
                *codbif++ = 00;         /* High byte 00 for chars */
                *codbuf = KFF;
                return(TRUE);
            }
=WINDOW Y
***** INSTALL.C(675) : error 65: 'codbif' : undefined
=WINDOW H
  Edit source file, then press <CTRL-E> for next error, <ESC> for menu
```

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

NEW VERSION 3.0

- **Best Multi-Level Undo**
- **Regular Expressions**
- **Pop-Up ASCII Table**
- **Pull-Down Menus**
- **Compiler Support**
- **Column Blocks**

Until now, if you wanted the best Undo, the best compiler support, regular expressions and column blocks you chose BRIEF™. 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, unequaled 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.

FREE EVALUATION COPY*
Call 1-800-45-VEDIT

- **Fully Network Compatible**
- **Call for XENIX and OS/2 versions**
- **30 Day Money-back guarantee**

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.

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PC-MOS/386 is a trademark of The Software Link, Inc. CP/M-86 and FlexOS are trademarks of Digital Research. MS-DOS, OS/2 and XENIX are trademarks of Microsoft. DESQview is a trademark of Quarterdeck Office Systems.

*Also available for TI Professional, Tandy 2000, DEC Rainbow, WYSE 700, Amdek 1280 and Others.
*Free evaluation disk is fully functional and can even edit small files.

Reader Service Number 7

1955 Pauline Blvd., Ann Arbor, MI 48103
(313) 996-1299 • Telex 701821 • Fax (313) 996-1308

CompuView

You can deliver it before 1991?

New Version 2.2
Compatible with DOS 4.0
Uses less memory, runs faster, and adds communications support

It's coming. An era of more powerful PCs. Easier to use PCs. With graphics and character-based programs working side by side. Talking to each other. Multitasking. Windowing. Menuing. Mousing. Letting your customers get their work done easier and faster.

Sell it all now.

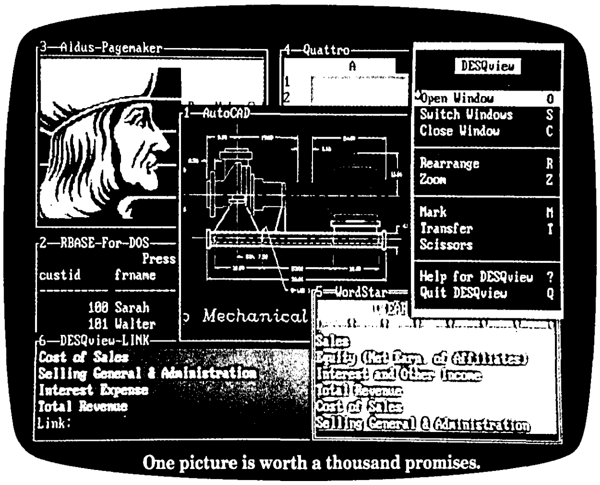
DESQview™ is the operating environment that gives DOS the capabilities of OS/2.™ And it lets your customers, with their trusty 8088, 8086, 80286, or 80386 PCs, leap to the productivity of the next generation. For not much money. And without throwing out their favorite software.

Add DESQview to your customers' PCs and it quickly finds their programs and lists them on menus. So they can just point to the program, using keyboard or mouse, to start it up. DESQview knows where that program lives. And what command loads it.

For those who have trouble remembering DOS commands, it adds menus to DOS. It even lets them sort files and mark specific files to be copied, backed-up, or deleted— all without having to leave the program they're in.

Best of all, DESQview accomplishes all this with a substantial speed advantage over any alternative environment.

For programmers, DESQview's API, with its strengths in inter-task communications and multitasking, brings a quick and easy way to adapt to the future. With the API's mailboxes and shared programs, programmers are able to design programs running on DOS with capabilities like those of OS/2.



```
custid   frname
--------------------
   100 Sarah
   101 Walter
```

```
Cost of Sales
Selling General & Administration
Interest Expense
Total Revenue
Link:
```

One picture is worth a thousand promises.

Multitasking beyond 640K.

When your customers want to use several programs together, they don't have to leave their current program. Just open the next program. View programs in windows or full screen. Open more programs than they have memory for. And multitask them. In 640K. Or if they own a special EMS 4.0 or EEMS memory board, or a 386 PC, DESQview lets them break through the DOS 640K barrier and multitask. For instance, they can start 1-2-3 calculating and tell Paradox to print mailing labels while they're writing a report in Word Perfect, or laying out a newsletter in Ventura Publisher, or designing a building in AutoCAD.

DESQview even allows them to transfer text, numbers, and fields of information between programs.

Fulfill the 386 promise.

For 80836 PC users, DESQview becomes a 386 control program when used in conjunction with Quarterdeck's Expanded Memory Manager (QEMM)-386—giving faster multitasking as well as virtual windowing support.

And when you use DESQview on an IBM PS/2™ Model 50 or 60 with QEMM-50/60 and the IBM Memory Expansion Option, DESQview gives multitasking beyond 640K.

Experts are voting for DESQview. And over a million users, too.

If all of this sounds like promises you've been hearing for future systems, then you can understand why so many VARs and system integrators have chosen DESQview.

PC MAGAZINE EDITOR'S CHOICE NOV. 24, 1987

INFO WORLD 1986 AND 1987 PRODUCT OF THE YEAR

And why *PC Magazine* gave DESQview its Editor's Choice Award for "The Best Alternative to OS/2," why readers of *InfoWorld* twice voted DESQview "Product of the Year" why, by popular vote at Comdex Fall for two years in a row, DESQview was voted "Best PC Environment" in *PC Tech Journal's* Systems Builder Contest.

DESQview lets you sell it all now.

DESQVIEW SYSTEM REQUIREMENTS:
IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286, or 80386 processors) with monochrome or color display; IBM Personal System/2• Memory: 640K recommended; for DESQview itself 0-145K• Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage; EMS 4.0 expanded memory boards•Disk: two diskette drives or one diskette drive and a hard disk•Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA). IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA)• Mouse (Optional): Mouse Systems, Microsoft and compatibles• Modem for Auto-Dialer (Optional): Hayes or compatible• Operating System: PC-DOS 2.0-4.0; MS-DOS 2.0-4.0• Software: Most PC-DOS and MS-DOS application programs; programs specific to Microsoft Windows 1.03-2.1, GEM 1.1-3.0, IBM TopView 1.1• Media: DESQview 2.0 is available on either 5-1/4" or 3-1/2" floppy diskette.

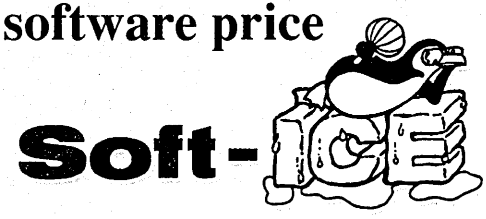
Quarterdeck
150 Pico Blvd., Santa Monica, CA 90405
(213) 392-9851 Fax: (213) 399-3802

Trademarks are property of their respective holders: IBM, OS/2, PS/2, 1-2-3, Paradox, Word Perfect, Ventura Publisher, AutoCAD, Intel, Above Board, AST, RAMpage, Advantage, Hercules, Mouse Systems, Hayes, Microsoft, Windows, TopView.

Reader Service Number 109

SERIOUS DEBUGGING *at a* REASONABLE PRICE

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



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*

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.

RUN CODEVIEW IN ONLY 8K!

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	$199
Soft-ICE	$386

Buy Both and Save $86!

CALL TODAY
(603) 888 - 2386
***or FAX* (603) 888 - 2465**

30 day money-back guarantee
Visa, Master Card and AmEx accepted

NU-MEGA TECHNOLOGIES
P.O. BOX 7607 • NASHUA, NH 03060-7607

Reader Service Number 110

MagicCV with Soft-ICE

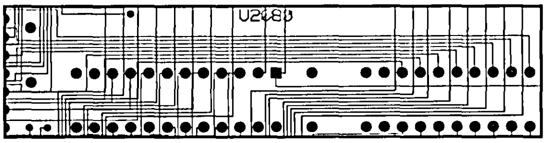
Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

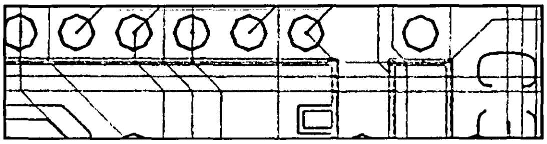
THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

JANUARY/FEBRUARY 1989 - ISSUE NO. 45

FEATURES





COLUMNS

FUTURE TENSE

Cover illustration by Paul Leatherwood

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Editor and Publisher
David J. Thompson

Associate Editors
Gary Entsminger
Cary Gatton

Technical Department
Larry Fogg

Director of Advertising & Distribution
Jackie Ringsage

Accounting
Sandy Thompson

Order Department
Tammy Westfall

Graphic Design
Carol Steffy

MICRO CORNUCOPIA (ISSN 0747-587X) is published bi-monthly for $18 per year by Micro Cornucopia, Inc. 155 NW Hawthorne, Bend, OR 97701. Second-class postage paid at Bend, OR and additional mailing offices. POSTMASTER: Send address changes to MICRO CORNUCOPIA, PO Box 223, Bend, OR 97709.

SUBSCRIPTION RATES:	
1 yr. (6 issues)	$18.00
2 yr. (12 issues)	$34.00
3 yr. (18 issues)	$48.00
1 yr. Canada & Mexico	$26.00
1 yr. Other foreign (surfaca)	$36.00
1 yr. Foreign (airmail)	$50.00

Make all orders payable in U.S. funds on a U.S. bank, please.

CHANGE OF ADDRESS: Please send your old label and new address.

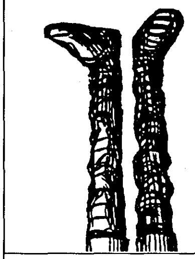
MICRO CORNUCOPIA
P.O. Box 223
Bend, Oregon 97709

CUSTOMER SERVICE: For orders and subscription problems call 503-382-8048, 9 am to 5 pm, Pacific time, M-F.

For technical help call 503-382-8048,
9 am to noon Pacific time, M-F.

BBS - 24 hrs. 300-1200-2400 baud
8Bits, No Parity, 1 Stop Bit
503-382-7643

ISSN 0747-587X



AROUND THE BEND

By David J. Thompson

Hello New Zealand

When Pat O'Leary of Dublin, Ireland (where else?), suggested I use Around The Bend for the title to this column, I assumed he was one of those poor defects who can't resist a pun. (*Micro C* is based in Bend.)

Silly me.

A New Zealander recently subscribed because *Micro C* was the only technical journal whose editor admitted being "around the bend." He added that I'm giving his friends quite a chuckle because in that part of the world, "around the bend" means "more than a bit off."

Fine. I enjoy a joke as well as the next guy. (Really I do.)

In fact, I enjoy a joke so much that I've just called the outfit that prints *Micro C* and told them to print the Australian/New Zealand copies right side up from here on.

It'll be great fun knowing you fellows "down under" will have to stand on your heads to read this issue.

(That's it, no more Mr. Nice Guy. Now if I got pictures, I might have a little mercy.)

This Issue Is A CAD

There's been a lot of interest in CAD (computer aided design) lately. Desktop publishing, cheaper workstations, and the faster AT and 386 systems have all contributed to that interest.

Package designers have a target. There are wonderful packages running on Suns and Apollos, but they're a bit pricey for us mortals. (At least we know what we'd like to have.)

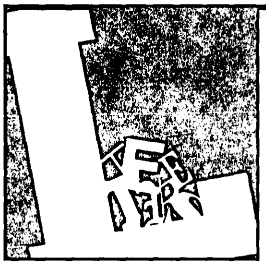
This issue we have three CAD pieces. Scott Baker, who comes from the workstation environment, talks about autorouting. Sam Azer, who's used a number of Micro CADs (read between the lines), talks about his experiences, and Bruce Eckel describes his frustrations trying to review them.

Basically, a CAD package is like a lifemate. You invest a great deal of time and energy getting to know one another. If the relationship doesn't work out, it hurts. The package's price in dollars may be small potatoes compared with its price in sweat.

Finally, I mentioned desktop publishing. I've been using AutoCad to enter a small schematic (and Sandy used it on our XT schematic). I can understand why she wouldn't go back to drawing schematics by hand, just

Continued on page 70

Aztec
The 680x0 Family
The 8086/80x86 Family
The 8080/Z80 Family
The 6502 Family



Letters

Ya Done Good

Just wanted to let you know that your latest issue of *Micro C* (Nov.-Dec. 88) is a winner! Would you believe I spent the whole evening reading it?

I liked the editorial of course (I never miss that!). But, about your "fly around": that would have given me quite a scare if I had known about it while you were in flight.

Then there was Laine Stump's travelogue from Turkey. Oh, I skipped over the few references to computers, but mostly it was good. The comments on SOG VII made me proud, especially the mention of Debee and the hit she made with everyone. I liked what Barbara Hall wrote, too. Actually, this issue tripled my reader interest.

"Bits From Your Past" was okay, but I could fill you in a little on that. Like when you were in Junior High and refused to write even one paragraph in English class and did a balsa wood project in place of a theme. And when you were in college and wrote those six-line weekly letters home.

Actually your new format and cartoons and color are all impressive — and yes, even the plastic bag. If I could understand computers at all, I'd read the whole magazine. But just the same, I'm proud of you and love to display your brainchild on the coffee table.

Now to get back to *Better Homes.*

Your ever lovin' Mom.

P.S. Even though we haven't paid for it yet (we're saving for a trip to Turkey), please keep our subscription coming. Many thanks.

Editor's Note: Thanks ma. I guess being grown up means never having to ask for the keys to the plane.

Slingin' Chips!

In your July-August issue there was a letter from one Bevin J. Pettitt. While I agree with most of what Mr. Pettitt wrote, I must take exception to his generalization that all computer store sales staff are nothing more than a bunch of pretentious buffalo chip slingers.

As an assistant manager of a large computer store chain, I would like to know where this dishwashing jarhead gets off calling me a pretender! And, as if that weren't bad enough, you took the liberty to address some of his letter in your editor's note without scolding him.

Even though I'm just a lowly computer store person, I am well versed in several languages such as 6502, 8086, and 68000 assembly, FORTH, C, Prolog, BASIC, and APL. I have a good

Continued on page 82

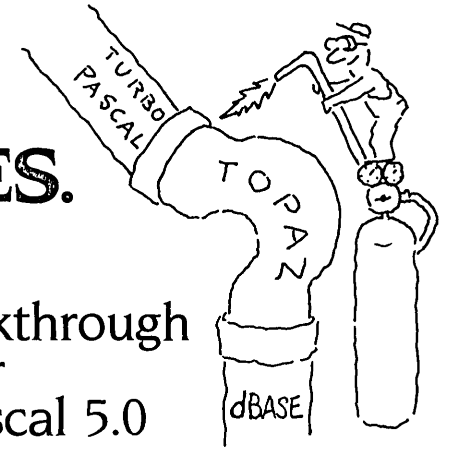
YOU'LL LOVE THESE UTILITIES.

SAYWHAT?!

The lightning-fast screen generator

TOPAZ.

The breakthrough toolkit for Turbo Pascal 5.0



WE GUARANTEE IT!

IRON CLAD MONEY-BACK GUARANTEE.
If you aren't completely delighted with Saywhat or Topaz, return them within 30 days for a prompt, friendly refund.

It doesn't matter which language you program in. With Saywhat, you can build beautiful, elaborate, colorful screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can all be displayed with as little as one line of code in *any* language. Batch files, too.

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**

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 units for Turbo Pascal 5.0) was specially created to let you enjoy the best of *both* worlds. The result? You create truly dazzling applications fast. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you easily write outstanding, polished programs. Now you can write in Pascal using **SAYs** and **GETs**, **PICTURE** and **RANGE** clauses, then **SELECT** and **USE** databases and **SKIP** through records, plus **BROWSE, INDEX ON, FIND, PACK, APPEND,** a complete set of time and date handling routines and lots more.

In fact, we've emulated nearly one hundred *actual dBASE commands and functions*, and even added *new* commands and functions! All *you* do is declare Topaz's units in your source code and you're up and running!

The bottom line? Topaz makes writing sophisticated Pascal applications a snap. Data entry and data base applications come together with a minimum of code and they'll always be easy to read and maintain.

Topaz comes with a free code generator that automatically writes all the Pascal code you need to maintain a dBASE file with full-screen editing. Plus truly outstanding technical support, 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.

Topaz, your programmer's toolkit for Turbo Pascal 5.0, 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 ____________________

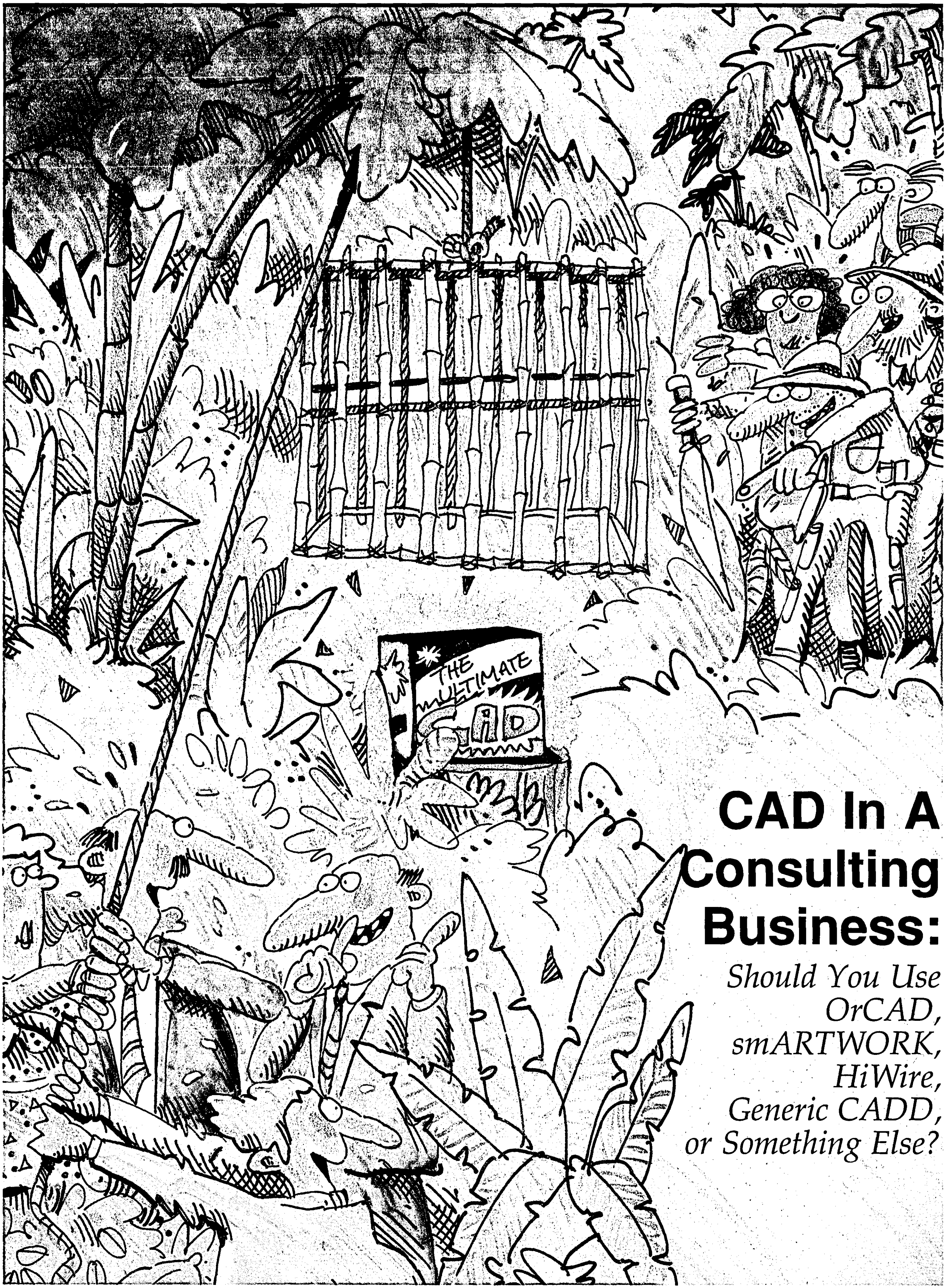
CITY ____________________ STATE __________ ZIP __________

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

# ____________________ Exp. date ______ Signature ____________________

T H E R E S E A R C H G R O U P

Reader Service Number 129



CAD In A Consulting Business:

Should You Use OrCAD, smARTWORK, HiWire, Generic CADD, or Something Else?

By Sam Azer
AzerTech
2095 Patricia Ave. #1
Montreal, PQ
Canada H4B 1Y7

Sam's a natural for this issue. He's been struggling with CAD software about as long as I've been struggling with desktop software. (And he's been desktop publishing, too.)

So here's one man's story of what works (and doesn't work) for him. Can you ask for anything more?

Like thousands of folks in North America, I make my living by "consulting." That is, I write software and design (and build) hardware for people and companies who don't do it themselves. I work across a rather broad range — I see people on the high levels, try to understand what they want. Then, I go into the depths and try to make it happen.

Over a year ago, I decided to bring PC-based CAD software into my business. I hoped it would improve my productivity. It did.

This article is about the systems I use (OrCAD SDT, smARTWORK, HiWire+, Ventura Publisher, and to a lesser extent, Generic CADD). I'll tell you what they do and what they're worth to me.

The CAD systems I'll talk about are the *only* CAD systems I've used. I've seen AutoCad, but that's about it. So it's possible that my notions about what's powerful, what's easy to use, what's useful, and what's not worth bringing home might not agree with yours.

Desktop Publishing

Yup, I know, Desktop Publishing is a strange place to start a CAD review. In fact, it isn't really even CAD. But I'm trying to paint a picture of a consultant using CAD. So I think a few digressions in the early going will help you understand my comments about specific CAD systems later.

Text is fundamentally important to the CAD consulting process. The consultants I know send out a seemingly endless stream of text. And I spend as much time writing (during a project) as on everything else combined.

Every week there's a letter, a preliminary analysis for a new project, a detailed analysis for one in progress, etc.

Over a year ago, I decided to bring PC-based CAD software into my business. I hoped it would improve my productivity. It did.

Documentation! There's tons of it — schematics, block diagrams, drawings, tables, mountains of listings, and just plain text.

It's important to present these documents properly. With some clients, it's critical. Obviously, typesetting won't change the relevance or correctness of information, but it will help in other ways. Having charts, diagrams, tables, and schematics with text can help a client plow through complicated material.

Ventura Publisher

Ventura is a sophisticated formatter. (In fact, it might be too sophisticated for some uses and users.) I chose it for simple reasons —

- it's very easy to use,
- accepts text and graphic files in a number of different formats,
- responds fast enough,
- and has a few features (such as style sheets) that I really like.

One big plus — Ventura documents don't have special file formats that exclude other software. This lets me use my favorite editor, Micro EMACS, to make changes. I'm not sure, but I think PageMaker uses its own file format, so you must edit with it. I like keeping my options open.

Recently, I got to tinker with Microsoft Word and was very impressed. Although Word can't do as much as Ventura, it does a lot. My guess is that I'd probably get good-enough results formatting with it instead of Ventura. But I'd hate to give up Ventura.

I couldn't imagine life without a Laser printer, either. It's true that they're several times more expensive to operate than a dot matrix, but they're versatile, very fast, and have unmatched character and graphics definition.

Don't spring for the Laser printer until you've got Ventura, though. Ventura does a great job on a dot matrix printer; it just gets better with the LaserJet. In my view, the LaserJet is crippled without Ventura.

Last week, I laid out a front panel for a prototype using the graphic drawing functions of Ventura. I then printed the layout onto a special kind of MacTac sticky paper. After peeling the wax paper off the back of the MacTac, I carefully pressed it onto the aluminum faceplate. Voilà! Almost-instant front panel.

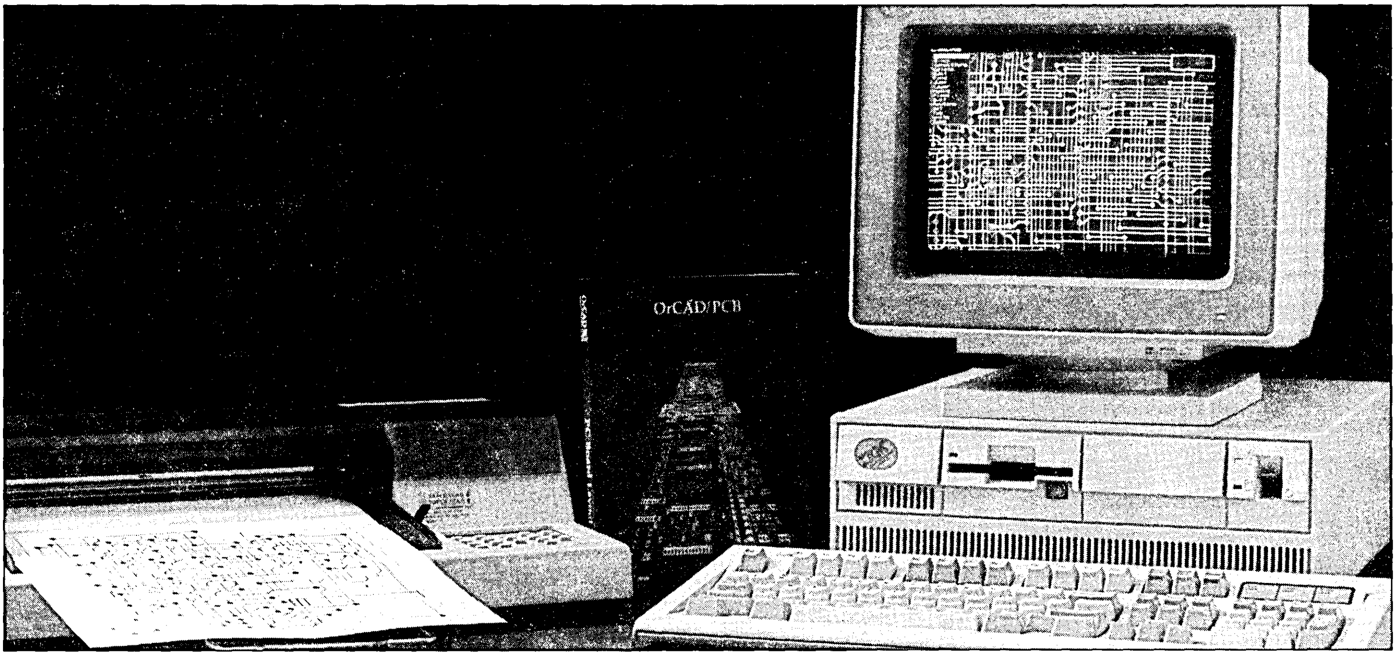
OrCAD SDT/III

As with most labor saving devices, computers don't actually save labor. It takes as long to draw a line with a mouse as with a pencil. The difference usually shows up only during editing. OrCAD SDT, like the washing machine, seems to be an exception to this rule.

I bought it about a year ago. It was fabulous! The learning curve was so short that I was producing schematics almost instantly. It's so easy to use that I actually scratch up rough ideas with it. I haven't scribbled a schematic onto looseleaf in so long that I'm not sure I still remember how. (Can you write home with a Crayon?)

The User Interface

Basically, OrCAD works around their DRAFT program. If you like rodents (I usually don't), you can use DRAFT almost as if it were running on a Macintosh.



Usually, I use the keyboard cursor keys for short moves and the mouse for long ones. I can select from menus with either the mouse, cursor keys, or a first letter.

If you don't like mice or keyboards, a simple keyboard macro facility allows you to map any sequence of keystrokes or mouse moves onto almost any key.

No matter which input method you use, OrCAD's user interface is easy to get used to, easy to use, and *fast*!

Libraries

If you type "G27256<enter>" — WHAP! — a 32K by 8 EPROM appears on the screen. Move it to where you want it, then type "P" (Place) to put it down. Move it some more, then type "P" again; until you've nailed down all your EPROMs. Ditto for any other part or group of parts. (You can specify groups on the fly using a window, etc.)

The standard library of parts is massive, and at first, I couldn't find a few oddball parts. But I quickly stopped worrying.

OrCAD has a 24-hour BBS full of user written libraries, which you can download free. And it's easy (easier than downloading, really) to make up the occasional part from scratch using their library edit program, LIBEDIT.

When I'm looking for a suitable part, and don't know what it might be called, I use the DRAFT program's Library Browse command. It lets me rummage around in the libraries until I find what I want.

When I first installed the OrCAD package on my system, I blindly put in all the libraries. Later, when I started getting the equivalent to out of memory errors, I decided to look through the libraries to see if I could omit some of the parts.

In the process I discovered that the SHAPES library contains the sorts of things that you'd use to make up flow charts and block diagrams (I hadn't known that). Now I use OrCAD for that, too.

Sheets & References

When you place a part on a sheet, the name of the part and its reference automatically appear next to it. You can easily change everything, including the relative position of the name and reference labels. But often, there's no need. If you think they'll clutter up the drawing, you make them invisible.

The default reference number for each part is "?" (i.e.: "R?" for a resistor, "C?" for a cap, "U?" for a dip, etc.). You don't have to edit those numbers if you don't want to. There's a program called ANNOTATE that'll number all the parts for you.

OrCAD supports single sheet schematics, multiple sheet schematics, and something called a hierarchy. That's a neat idea whereby you draw a circuit that you use often, then refer to it in other schematics.

If you need three of something, refer to it three times — but only draw it once.

You can draw on sheets from size A (8.5" x 11") through size E (32" x 44"), and work at zoom factors from 1:1 to 20:1. You can do it in color if you like, and it all runs at blindingly fast speeds. When printing, which you can do from within the DRAFT program, you can either compress your drawing to fit the paper you have, or print the drawing in sections.

Toolbox & Manual

OrCAD includes many utilities that make it easier to use —

- BACKANNO
- CLEANUP
- LIBARCH
- LIBLIST
- PARTLIST
- PLOTALL
- PRINTALL
- TREELIST

- FLDSTUFF
- FLDATTRB
- COMPOSER
- and DECOMP

and the following indispensable utilities —

NETLIST — currently generates 26 different kinds of netlists. It lets you communicate the structure of your schematic to other programs such as simulators and PCB CAD packages. It also generates a list of connections which can be used as a checklist for wire wrapped prototypes. *(Editor's note: A netlist is a list of all the parts and their interconnections. It's usually generated by the schematic capture program for use by the board layout package.)*

CROSSREF — generates a whole slew of handy cross reference tables. For example, it can actually figure out if you've got any spare gates available and list them for you!

ERC — an error checking program. This program does basic electrical rules checking. I've found that it helps highlight obvious flaws in a schematic.

Sometimes these are just drawing errors, but (during the wee hours?) they can also be conceptual errors. At any rate, you don't always have to wait until you're awake before you find out about the obvious ones with ERC on hand.

Command line options are consistent across all the programs in the package — a small detail, but it does make the programs easier to use.

The manual is a bit repetitive at times, but it's understandable and fairly well organized. Although there isn't a quick reference section, it's still very complete. For example, one of the many appendices contains sample device driver listings and an explanation of how they work.

OrCAD, The Company

The company's simply a joy to deal with. I needed to get some schematics out of OrCAD and into Ventura one night, so I wrote a simple program to convert the output from their generic plotter driver (something they invented to help out in such emergencies) to a DXF file. I called them up a few days later and mentioned what I'd done.

A couple of weeks later, I received an update with a new DXF driver. Then Bruce Eckel (see his piece this issue) asked me if they had a PostScript driver (he needed one). A few days later, another update arrived. This one included a file called PSCRIPT.DRV!

I noticed with one update that the Repeat command didn't work properly with labels. I discovered they'd added an Auto Place option in the Set Repeat menu, which was almost better than the Repeat command. Then a few months later I got an update which fixed the bug in the Repeat command. (That one must'a stumped 'em...)

I complained once that the compressed size B sheet came out with the title block away from the paper margin. This made it difficult to punch three holes on the side of the sheet and stick it into a binder. Again, it wasn't a problem for long.

I doubt that they actually jump when a customer calls, more likely they just make it seem that way! They obviously start working on something the moment they hear about it, but they don't say so. By the time I've noticed it, the fix must already be on its way.

The version of OrCAD that I originally got was 1.27. I liked it, but for some reason they didn't. So they rewrote it completely. They made up new manuals, put the whole thing into a nice box, and sent it out without so much as a post card to say it was coming. I was very happily shocked.

I've had the package for over a year now. They're up to version 3.11. They've sent me two complete manuals, one addendum manual, and I count four sets of update disks in my master box. I haven't sent them a penny since I bought the package.

Another little thing they do well is answer the telephone. Doing that, and doing it properly, adds up to quite a bit in my opinion. No joke. I've got at least a half dozen horror stories about companies with telephone problems, and I don't use the phone all that much.

Whenever I call, the folks at OrCAD answer the phone immediately; and I'm allowed to talk directly to the guy who's responsible for maintaining the SDT package if I want to. I never need to.

I rate this company Excellent on Azer's (very well known) Software Supplier Rating Scale. They get 100%, hands down, with no competition in sight. The product is top notch, not just in terms of the quality of the results it produces, but in: ease of use, speed of operation (*fast!*, even on my old 8 MHz XT), and flexibility.

The support behind the product constitutes a good definition for the term support. The price is quite reasonable, and at $495 is less expensive than any of the other packages that I bought last year (except Generic CADD, which doesn't really count.)

Okay, I realize that this sounds like a sales pitch. All I can say is that I'm not getting anything — money or otherwise — from OrCAD for writing it. I just happen to be very happy with their package. (Not that I wouldn't be just as happy to see a copy of their PCB package, if they'd send one.)

smARTWORK

I like this program quite a bit also. It has a few places where things can be improved, mainly in the user interface, but essentially it's a solid performer. The main thing to realize about smARTWORK is that it's got some limitations that are acceptable in many cases, and not acceptable in others. And (relatively speaking) it's inexpensive.

The main limitation is that the grid is fixed at 50 mils. Most parts have 50 or 100 mil pin spacing, but there are quite a few parts that don't. There are two solutions to this. The first one is to avoid using parts that have off-grid pins, and the other is to print the foil pattern, then add the off-grid pads by hand.

I've used both solutions, and note that parts with 50 mil pin spacing aren't hard to find, and don't cost any more or less than their off grid counterparts. I've also found that such parts as PCB mount DB25s are often difficult to fit into a board.

It's much easier to pay extra for the ribbon version, and use a 24 pin Winchester type connector on the board. (Remember, my production volumes are low, under 50 units per run, with a normal run being 2 to 10 units. So my concept of cost is different from a guy's who makes 100 or more units at a time.)

Another problem with smARTWORK: there can only be two trace widths on a board, Thick and Thin. And there are only three thin trace widths to choose from: 12, 16 and 20 mils. Thick traces can be combined to get trace widths in 50 mil increments. Again, most boards don't have to be any more complicated than that, so it often doesn't matter.

Last, but definitely not least, there are only two pad sizes available: 62 or 75 mils, round or square. I find that 62 mils is about as small as I'd bother with, especially after it's been shaved. (Pad shaving is done automatically to ensure the required minimum amount of spacing between conductors.) A square pad can be made any size by placing what they call "fat cells" around it.

smARTWORK is nice to have around because it's simple and relatively easy to use. And the problems I've mentioned don't bother me all that much, since I don't do that many boards in a year anyway.

Using smARTWORK

Drawing PC boards is a job nobody should have to do. The way I see it, the main problem is that the foil pattern doesn't even remotely resemble anything a human can read. It's just plain hard to figure out if there's a mistake.

> For a really complicated board, you'll still have to scribble wire routing ideas on paper before putting down the pads.

smARTWORK helps in two ways:

First, the EDIT program makes the knife, tape, and mylar that we're used to working with obsolete. Dry transfer symbols and the burnishing tool are partly replaced by commands that handle a number of common parts. A manual router automatically draws a trace between two pads, using the shortest path it finds.

The second form of help is even more valuable: it loads a netlist (in my case, from OrCAD) then shows me missing pads and traces.

A Simple Example

I once got into a friendly argument with a fellow consultant about the value of something like smARTWORK. Since we were both sitting in front of my system at the time, I tried the following demonstration:

First, I went into OrCAD's DRAFT program. I loaded the schematic for a speech synthesizer board, and exported the audio amplifier section to a new file.

I edited that file so that it became a simple, standalone, 1 Watt audio amplifier, with a trimpot volume control, an AC input, speaker output and connector for a battery. I'm a fairly quick typist, so this only took about two or three minutes.

OrCAD's ANNOTATE program then numbered all the parts, ERC checked for obvious errors, PARTLIST listed the parts, NETLIST did the netlist, etc. (I've put together a batch command that runs "the works.")

After looking over the printed schematic and the output file from ERC, we both agreed that the schematic was okay. So we changed to the smARTWORK directory.

I then ran smARTWORK's EDIT program. First, I loaded the netlist and checked the list of what they call loose terminals (i.e.: missing pads). Because I was just showing a friend how it worked, I didn't bother to put the pads in the right places, I just banged them out, one after the other.

Finally, I ran the autorouter. Because the circuit was so simple, and there was so much space between all the pads, the routing was completed in only a few seconds.

EDIT doesn't check for short circuits or extra traces that might have been entered by hand. So I ran two other utilities, NETGEN and NETCMP, to generate a netlist from the foil pattern. Then I compared the two netlists.

For added excitement, I purposely shorted two nets together, reran NETGEN/NETCMP, then erased a trace, and ran them again. Naturally, NETCMP produced the proper messages.

The schematic, foil pattern, and silk screen were all done in about 15 minutes. (And smARTWORK automatically generates solder masks.) My friend was impressed.

Less Simple

Unfortunately, for a real board, it's not that simple.

Laying down the pads is no picnic. Each pad has to be named according to the netlist, and the spacing and positioning all have to be checked by hand. For a really complicated board, you'll still have to scribble wire routing ideas on paper before putting down the pads.

The idea of an Autorouter is pretty exciting, but realistically it's just a glorified aid because it's extremely stupid. If you don't coach it, it'll run the

Reader Service Number 2

first few lines all over the place, then be unable to finish.

(I get around this problem by feeding the netlist to the autorouter in pieces. I found it helps to do the power nets first, making it easier to see which traces have to be fat.)

All in all, I estimate (very roughly) that electronically I'm about three to five times faster than I am with knife and tape. If the board is dense and you make a number of changes then you'll probably have to compare the whole thing against the original schematics. It'll take several seconds with NETGEN/NETCMP instead of several hours by hand.

Printing Or Plotting

The PLOT program works with a limited number of plotters, but I only use it to generate DXF files.

Since smARTWORK has fixed pad sizes, fixed spacing between conductors, and fixed conductor sizes, I've found that anything that I can do on smARTWORK, I can print on a dot matrix printer. For me, dot matrix output is fine, but not everyone agrees.

I've tried three PC board suppliers so far. Supplier #1 warned me that I'd have to pay even if the boards didn't come out. Supplier #2 asked me if I knew what I was doing. And supplier #3 outright refused to even try. (I originally tried making them myself, but I gave up on that. It's not worth the trouble.)

I agreed to take responsibility in the first case, and the boards came out perfectly. But the price was too high. In the second case, the price was okay, but the exposure was bad, and the supplier had to separate some traces by hand using a blade (!!!). In the third case, I had to send an unused blank board to get the supplier to agree to do the job. The results were perfect.

I've only tried my two 9-wire dot matrix printers with smARTWORK — an Epson RX-80 and a Roland PR-1215. I find that the condition of the ribbon is critical. The ink was too dry for the first batch of boards, so the edges of the traces had tiny but visible ridges on them.

If the ink is too wet, I can't move the print head up close enough to the paper without getting ink streaks. If I move the print head back, the reverse happens: some of the dots don't come out, leaving white streaks. Obviously, the ink and the position of the print head all affect contrast too. Good contrast is important.

All the problems listed above get worked out quickly, though. And it's infinitely easier than cleaning the tips on plotter pens.

Smart Conclusions

smARTWORK does what it does very well. The manual is complete and relatively easy to follow. It's sprinkled with little diagrams, and includes a tutorial and a command summary.

I was going to complain a bit about the silly user interface, but I found out that it's recently been modified. Since my gripes with smARTWORK aren't really significant, I'll keep my mouth shut until I get my hands on the update.

Support from the manufacturer is probably about average. Again, they answer the phone, and they allow me to speak to the guy who writes the software. (I like that.) They're helpful when a problem comes up, and they know what they're doing. They also send out, two to four times a year, technical bulletins and newsletters. The only thing that bugs me is having to send back the previous update in order to get the new one.

Overall, I would definitely recommend smARTWORK to a company that has several simple boards to do per year. It's so efficient that it'll pay for itself shortly. If you need something more sophisticated, or if you don't do many boards, check around before buying. Don't forget that Wintek has a 30 day money-back trial period.

HiWire+

I don't like HiWire. I bought it just as it was being released for beta testing, and I agreed at the time to live with whatever bugs I found until they could be corrected. As it happened, I didn't have time to look at it too closely, and my thirty day money-back guarantee ran out. In all fairness to Wintek, it must have been around three months before I complained, so I suppose I don't deserve to get my money back.

Essentially, the folks at Wintek are technically competent. They produce good, solid code. Unfortunately, my main problem with HiWire is that it's just too difficult to use. I've repeatedly made suggestions that have gone unheard over the couple of updates that I've so far received (and paid for, at $45 each.) It is getting better, but very slowly. They're working on an autorouter, which is great, except that I feel they should first improve the user interface and the editing functions.

I'll eventually send a list of gripes to Wintek, above and beyond what I've already mentioned over the phone. I hope that someday I can say that I've produced a board with it, but for now it just wastes space in one of my diskette boxes.

For PC boards, I'd say go ahead and try it out on the thirty-day trial if you're planning to work with SMDs or some such. But I only say that because I haven't tried anything else. (It's not fair to compare HiWire to smARTWORK, they're fundamentally different.) I just find it too difficult to bother with, although it's better than laying out a board by hand.

For schematics, forget it. It doesn't even begin to match OrCAD in performance, features, versatility, or price. Again, most of the code to do the work is there, it's just the way you enter information that makes the program so difficult.

Generic CADD

Generic CADD cost me $99. I hardly ever use it. I think I've worked on a design for an odorless litter box for my two cats, and several new and improved book shelves for my ever expanding library. Aside from that sort of thing, I edit DXF files with it, but that's about all.

I'd probably use it more often if it weren't so slow. It has a fairly good user interface, but it's also got a number of obvious problems. Not the least of these problems is that the dot matrix printer driver (costs extra) takes over 40 minutes to plot a small picture on my LaserJet compatible.

Basically, I figure I need to be able to work with DXF files, so it wasn't a bad investment.

I haven't seen the latest Generic CADD. It's probably got the more obvious things corrected and it's probably not a bad deal for someone like myself who doesn't do much mechanical stuff. So I can't recommend for or against it.

Wrap Up

I've learned quite a bit since I bought this software. For one, I don't do as much technical stuff as I thought I did. I got Micro EMACS for free (or almost for free, off BIX.) I use that and Ventura every day.

OrCAD is handy only when I'm interested in making a new board, and it's better to avoid that as much as possible. smARTWORK gets used only after a new design has been thoroughly checked out.

Basically, they say that a good businessman should always work up a long term cost analysis of anything he plans to do. I'm obviously a lousy businessman. I didn't do any cost analysis at all before buying the software.

I still haven't figured out what it costs me per contract to have this stuff on hand. And the more I think about it, I figure maybe I shouldn't have bought all of it. It's not just total losses like HiWire; look at smARTWORK — it's pretty fantastic! If I didn't have it, I'd still be telling my customers to get someone else to do the boards. However, I could've spent the money on a vacation, the one I haven't taken since '84.

Think about it: I charge by the hour. I now have all this expensive software that does in minutes what I used to spend hours doing. Hmmm...

Actually, I'm too childish to question certain things. The Desktop Publishing System is so fantastic that I couldn't imagine being without it. OrCAD lets me be creative while still being neat, and it does quite a bit of basic grunt work.

smARTWORK let's me promise to get stuff done faster than the competition can do it. It allows me to say with confidence that once I've checked my netlist against the schematic, there's no possibility that the PC board won't match the prototype.

So, I realize I should be charging flat rates for some types of work to make up the cost of all the software. Some of my clients are going to be upset when they see the quotes that I'll be sending out from now on ... "Documentation & PC boards no longer included: available at extra cost."

Notes

The Generic CADD version that I've got is 3.00. I'm sure it's out of date. The latest version of Ventura Publisher that's out is 1.20. Check the mail order ads or the local computer outlets for best price and delivery on these items.

I'm currently running smARTWORK version 1.4 r1, and I've got HiWire Plus version 1.1 r4. The latest updates for both should be arriving any day now. You can order these products directly from Wintek. Call them for pricing.

Wintek Corporation
1801 South Street
Lafayette, IN 47904
(317) 742-8428

To the best of my knowledge, 3.11 is the latest version of OrCAD SDT, their Schematic Design Tools package. They also have a great demo available for their logic simulator, and they recently released a PC board layout package.

OrCAD Systems Corp.
1049 S.W. Baseline Street, Suite 500
Hillsboro, OR 97123
(503) 640-9007

♦ ♦ ♦

Choosing A PCB Layout System

No More Tape and Mylar

Scott's like many of us, he's tried CAD on a work station and he's been hooked. So what does he use when designing boards on his home system? He's not sure yet. However, he's in the middle of a 30-day trial with Maxi/PC by Racal Redac.

As Scott will soon tell you, there's a lot to consider when picking a package (e.g., the money-back guarantee and the price with all the features). As I talk to users and manufacturers, I'm discovering there are about as many ways to create a PC board as there are people who want to. So, probably the best way to choose among the offerings is to check out the table of features in this article, then get on the phone and order one or two to try out.

I first became intrigued with using computers to lay out printed circuit boards (PCBs) two years ago when I had a chance to work on a Mentor Graphics work station. The system had a bit-slice 68000 CPU with 8 megabytes of main memory, several hundred megabytes of disk storage, and a 19 inch high resolution color screen. It was a pretty neat toy, and ever since then I have dreamed of having a home system with (at least some of) the same capabilities.

Well, recently I've noticed a number of advertisements for PCB layout software packages that run on PC clones. So I've been clipping the ads and comparing features. Although I have tried (either the real thing or demo versions of) four of these packages (Tango, OrCAD, Pads, and Maxi-PC), I haven't used any of them extensively enough to do a head-to-head comparison.

So I'm going to talk about the kinds of things I look for in a package. At the end of this article, I've included a table of features for eleven PCB layout packages (thanks to the *Micro C* staff for their research to complete this table). My apologies to those of you with Amigas and Atari STs, this table covers only PC-clone software.

From The Beginning

First a little background. A PCB consists of alternating layers of copper and fiberglass (or some other insulating laminate), the copper layers of which have electrical connections "printed" on them. The number of layers of a PCB refers to the number of layers of copper. Most PCBs in the PC clone world are either two or four layers, but it's not uncommon for PCBs to be eight or ten layers thick.

The copper layers in a PCB start out life as a solid plating of copper covering the fiberglass. A circuit is printed onto the copper layers by masking the parts of the copper that will form the circuit and etching away everything else. It is the process of creating the mask layers that we are concerned with in PCB layout.

In the good old days (ten years ago), the primary method of PCB layout was tape and mylar. The process (still in use today) consists of manually placing a thin opaque adhesive tape on a translucent (usually gridded) piece of mylar. The mylar used in this process is two or four times the size of the PCB.

Once the designer completes the taping, he takes the sheets of mylar to a service bureau where the masks ("films") are created by a photo-reduction process. The end result is that the tape on the mylar sheet corresponds to copper on the PCB.

Tape and mylar is a slow and tedious process, particularly doing revisions. So, with the advent of the minicomputer (cheap computing power!), people began searching for ways to automate the PCB layout process. A lot of progress has been made over the past ten years, and now PCB layout systems are available on personal computer systems. My experience is primarily with IBM PC clones, but there are also packages for the Amiga and the Atari ST.

Picking A Layout Package

It is important that your PCB layout package be compatible with your schematic capture package. Some of the PCB layout packages include schematic capture, some price the schematic package separately, and some must be used with schematic packages from other vendors.

The interface between the schematic package and the PCB layout package is a text file called a netlist. To be compatible, both packages must have a common netlist format. This is not usually a problem, since most schematic packages can write netlists in several formats.

The other compatibility consideration is back-annotation. This is the ability for changes made during PCB layout (such as swapping pins or gates) to be reflected in the schematic. This capability is much less common, particularly when using packages from two different vendors.

Two key features to look for in the PCB layout package are pin swapping and gate swapping (mentioned above). The ability to swap equivalent pins (such as the two inputs of a NAND gate) and the ability to swap gates (two NAND gates either in the same or different packages) can often simplify routing.

This is particularly true on two layer boards where routing is limited. Unfortunately, pin swapping and gate swapping are not available in many of the low-cost PCB layout packages.

Another feature to look for is the ability to display a "rats nest." A rats nest is a component placement aid, and is simply a straight-line connection of all the netlist connections. Just displaying these lines (rats) all at once is of limited use since it's usually too jumbled to make any sense.

However, if you can highlight the rats of an individual IC, you'll see right away which ICs it has the most connections to. Boy, is this useful. You can then move each IC to minimize the complexity. This is one area where PCB layout systems really shine when compared to tape and

By Scott Baker
Sierra Circuit Design
18185 West Union Road
Portland, OR 97229

mylar. And, every hour spent cleaning up placement is worth ten hours struggling with routing.

You may also need a package that handles multiple-layers (three or more). Some PCB layout packages were apparently designed to handle two layer boards and later modified to handle more layers. Usually these packages can show only two routing layers at a time, or it's difficult to get them to switch from one active layer to another. (The active layer is the one which you are currently routing and often must be switched to another layer when the route is blocked.)

However, if you're not after ultimate density, you'll probably be okay with two layer boards (just take a look at your PC's I/O cards).

User Friendly

Quick response to common operations (such as placing a route segment) is important. I can't maintain my concentration when I have to stop and wait after each mouse click (and, of course, I hate waiting).

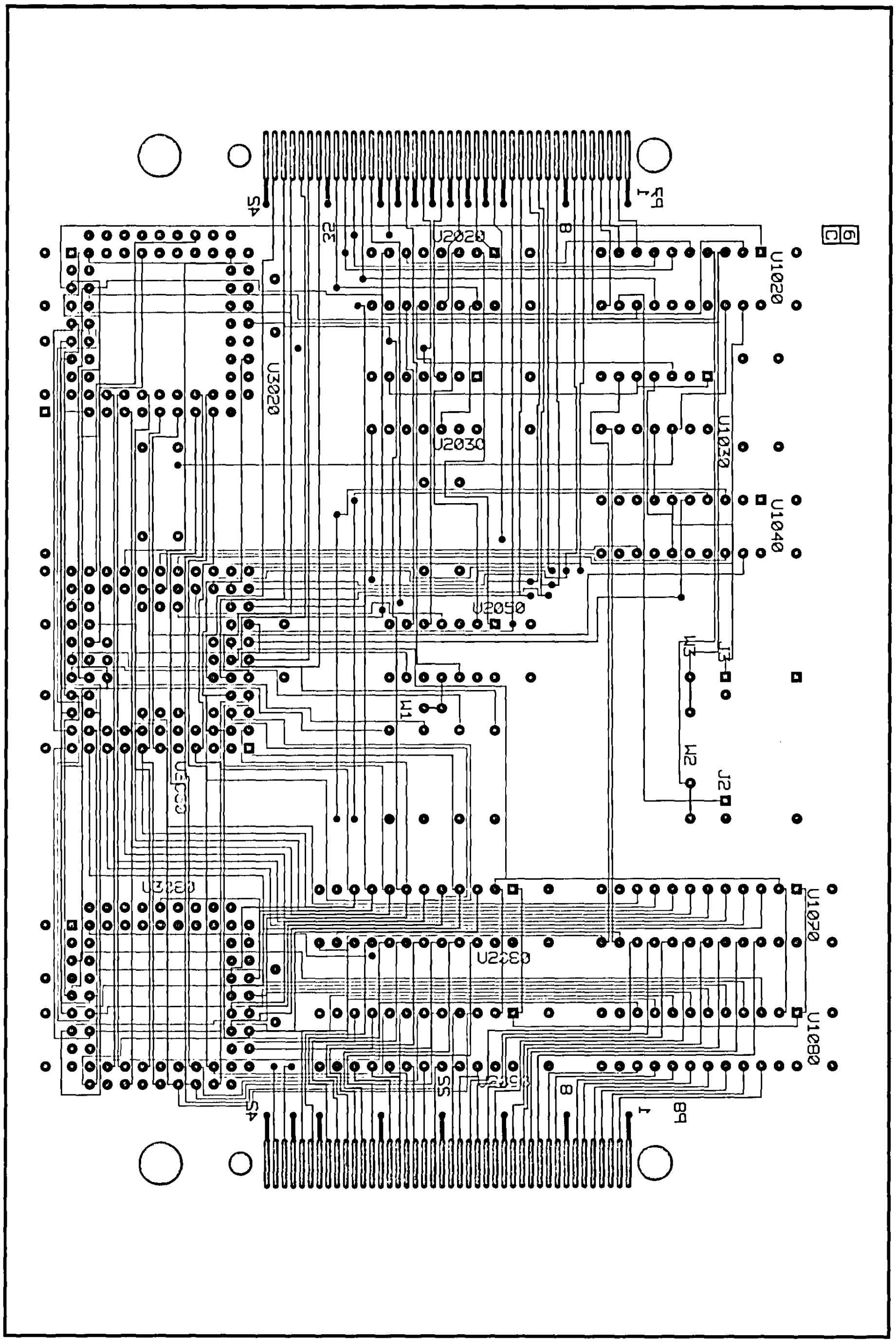
I like pull down menus, particularly while learning the system, but single-key commands and a menu bypass are handy once I'm comfortable with the program.

Zooming in and out and windowing sections of the PCB are definite requirements. Also I appreciate auto-panning, where the view shifts automatically whenever the cursor reaches the screen boundary. It's easy to get lost when looking at the routes on a PCB, so another useful feature is the ability to point to a route, and see its name on the screen.

Analog

You'll run into special problems when designing PCBs for high-speed analog and digital circuits where trace inductance and intertrace capacitance are important.

For these circuits, you'll often need to customize trace routes and widths. This



CAD Generated Masks

contrasts with common digital design where traces are usually the same width. Also, it really helps to be able to freehand draw copper areas. This capability is often called "area fill."

Ground & Power

The ability to create ground and power planes is also a requirement when building boards for high-speed circuits. On a ground plane, for instance, the entire layer remains copper clad, with copper removed around ungrounded pins and vias.

Thus, these copper planes provide a very low impedance path for ground and supply currents.

Editor's note: These copper planes, because of their proximity to the surface traces, also tend to reduce cross talk between signal lines.

Some designs may require multiple or split ground planes (a digital ground and an analog ground for example).

You'll face a very difficult problem if you're doing an analog design that's restricted to two layers (for cost reasons) and one of the two layers must be a ground plane (for electrical reasons). In this situation it may be necessary to also run traces on the ground plane layer. If you think you might run into this case, it's worthwhile asking the PCB layout vendor if his package will handle this.

Another useful feature for analog design is known as "area copy." This lets you duplicate sections of your PCB layout. If you have to freehand draw a section of circuitry, and if that circuitry repeats several times, then area copy will save you a lot of time.

Other Features

Post-processing capabilities that you should look for are a design rule checker (DRC), NC drill file output, and photoplotter file output.

A design rule checker should check the trace-to-trace and trace-to-pad spacings on the PCB as well as verify that the connectivity of the PCB matches the original netlist.

The industry standard drill file format is the "Excellon" format. You can save money if you create this file for the board manufacturer, otherwise they will charge you a tooling fee to hand-generate it.

The drill file specifies the location and size of all of the drill holes on a PCB. Given this information, a numerically controlled (NC) drill can automatically drill all of the holes in your PCB (aren't computers neat).

The industry-standard photoplotter format is the "Gerber" format. Photoplotters create the films for a PCB directly and tend to be more accurate than the photo-reduction process.

If your package doesn't generate photoplotter output, then you plot the layers on a precision plotter at two times actual size. Use a low pen speed with a black opaque ink. These layer plots can be photo-reduced onto film in the same way as tape-on-mylar.

Limits

Also look for the program's limits. What is the minimum grid size? (1 mil resolution is nice.) What is the largest possible board? What's the maximum number of routing layers? Can components be placed on both sides of the board (for maximum density PCBs may have surface-mounted components on both sides)? What's the maximum number of ICs per board? How large is the parts library? How easy is it to add new parts to the library?

Autorouters

Autorouters come in two basic categories: gridded and gridless.

Gridded routers are most common. These routers have a fixed grid on which PCB traces (wires) can be laid. They work best on moderately dense PCB designs and their main fault is that routing time increases dramatically as PCB density increases. This is because the number of nodes in the grid increases by the square of the dimensions of the grid.

Gridless routers should more properly be named variable-grid routers (I didn't make up the terminology), since they use denser grids to route areas of high congestion and sparser grids in the less congested areas. Gridless routers will vary the grid size on the fly (on the board, actually) in order to complete a route.

In addition to gridded and gridless routers, there and many sub-categories. One very common router is known as the Maze router, or Lee router (after its inventor C. Y. Lee). The maze router routes a single net at a time, starting at the source pin and proceeding toward the target pin. It does not remember previous routes, nor does it consider future routes which may be blocked by the current route. Thus, the maze router tends to be somewhat erratic and uses many unnecessary vias (feed throughs).

There are many (improved) versions of the maze router, such as the costed-maze router, with better performance.

The pattern router is a specialized router which recognizes repeated circuit patterns. It is intended for routing structured areas of a PCB such as memory arrays.

The bus router is similar to the pattern router in that it recognizes repeated circuit patterns. The bus router routes the most direct possible, it does not iterate, and does not use vias. Both the pattern router and the bus router should be used early in the PCB routing process.

The rip-up-and-retry router is a multipass router which attempts to complete difficult routes by removing routing obstacles, completing the difficult route and then rerouting the previously removed routes. In this way, a rip-up router emulates the way a human router completes difficult routes. (The rip-up strategy is actually used in combination with a routing strategy, such as costed-maze.)

People usually use rip-up routers to clean up problems left by other versions. Rip-up routers often do an ease-of-manufacturing pass, where tracks are rearranged and unnecessary vias removed.

Editor's Note

As we were finishing the final edit in this article I called Scott about his reactions to his new Racal-Redac package.

"I don't have it working yet. The copy protection requires one of those little devices connected to the serial port. My serial port apparently doesn't have all its wires hooked up so the software refuses to work.

"I've had this type of copy protection before — some schemes require devices hooked to the parallel port, others need to be hooked into the serial. All are supposed to be transparent but I've found that most of them have to be removed before running other programs that talk through the ports."

I asked him if all the CAD layout packages were protected this way.

"Most seem to be, though Orcad is an exception. They tried it but customer reaction was so negative they dropped copy protection altogether."

♦ ♦ ♦

Features of PCB-Layout Software for the IBM-PC.

The list is accurate as of 10/19/88, many of the vendors promise new/better (and in some cases more expensive) releases "real soon now."
DISCLAIMER: Much of the information in this table was gleaned from advertisements and from talking with sales representatives on the phone.
By all means do your own research, and if possible get a hands-on demo before buying.

Vendor	Price/ Version	Schematic Capture	Component Auto-place	Displays Rat's-nest	Pin & Gate Swapping	Autorouter	NC Drill Support	Photoplotter Support	Design-Rule Checker	Number Of Layers	General Comments
Design Computation Route 33 Sherman Sq Farmingdale, NJ 07727 (201) 938-6661	$695.00 Draftsman E.E.	included	$200	yes	no	no	included	included	no	64 total	"Vector Based graphics which makes an 80287 very important"
	$1195.00 DC Design	included	$200	yes	no	manual 32" X 32"	included	included	included	64 total	
	$1995.00 DC 810	included	$200	yes	no	automatic 8" by 10"	included	included	included	64 total	
	$3495.00 DC CAD	included	$200	yes	no	auto+ripup 32" X 32"	included	included	included	64 total	
Racal Redac 238 Littleton Rd PO Box 365 Westford, MA 01886 (508) 692-4900	$995.00 Maxi/PC	included	yes	yes	yes	3 algorithms Power & ground Orthogonal Memory Plus Partial	included	included	included	16 signal + power & grnd	"Best price/performance: a very short time ago we charged $10,000 for this kind of package."
AMS Inc. 1321 NW 65th Place Ft. Lauderdale, FL 33309 (305) 975-9515	$250.00 PC Pro II	$100.00	no (initial placement)	yes	no	$250.00 Custom version of Lee's algorithm	included (Nov '88)	included	$100 (Nov '88)	255 total	"Intuitive to use, no commands to memorize online help. "
(33OrCAD Systems Corp. 1049 SW Baseline St Suite 500 Hillsboro, OR 97123 (503) 640-9488	$1495.00 OrCAD PCB	$495.00	no	yes	yes	included	included	included	included	16 circuit 2 silkscreen 2 soldermask	"Productive work within an hour."
Cadisys 624 E. Evelyn Ave Sunnyvale, CA 94086 (408) 732-1832	$1295.00 AutoPCB 500	included	no	yes	no	no	$295	$495	included	16 signal 6 power & gnd + silkscreen & soldermask	"Our router uses C turns and S turns and AI to minimize use of vias."
	$1995.00 AutoPCB 1000	included	no	yes	no	Pattern generated	$295	$495	included		
	$2995.00 AutoPCB 2000	included	no	yes	no	Pattern generated incl surface mnt	$295	$495	included		
PCAD 1290 Parkmore Ave. San Jose, CA (800) 523-5207	$8495.00 Master Designer II	included	included with either autorouter	yes	yes	$5995 maze type $5995 ripup type	included	included	included	100 total	"Changes to board affect schematic & vice versa, 6000 component library, custom analog traces."
Visionics Corp. 343 Gibralter Dr Sunnyvale, CA 94089 (800) 553-1177	$995.00 EE Designer I	included	yes	yes	yes	included 50 mill orthogonal	included	included	included	3 total	
	$3995.00 EE Designer III	included	yes	yes	yes	included 25-50 mill ortho & 45 deg	included	included	included	26 total	"Version III has larger library & supports 2 meg of extended memory."

Features of PCB-Layout Software for the IBM-PC.

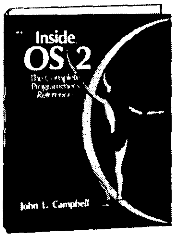
Vendor	Price/ Version	Schematic Capture	Component Auto-place	Displays Rat's-nest	Pin & Gate Swapping	Autorouter	NC Drill Support	Photoplotter Support	Design-Rule Checker	Number Of Layers	General Comments
CAD Software Inc. PO Box 1142 Littleton, MA 01460 (800) 255-7814	$975.00 Pads-PCB	$975.00	optional $300	yes	yes	optional std.$750.00 rip-up & reroute $4500.00	included	included	included	30 total	"Equal to $100,000 CAD workstations."
Wintek Corp. 1801 South St Lafayette, IN 47904 (317) 742-8428	$895.00 Hi-Wire Plus	included	no	no	no	$895.00 gridless ripup & reroute	included	$495.00	included	255 total	"Easy to use."
Interactive CAD Systems 2352 Rambo Court Santa Clara, CA 95054 (408) 970-0852	$695.00 ProCAD Plus	included	no	yes	single command	part of $1595 package	part of $1595 package	part of $1595 package	no	50 total	"Total solution from schematic to PCB."
	$795.00 ProCAD Extra	included	no	yes	single command				yes		
Accel Technologies 7358 Trade St. San Diego, CA 92121 (800) 433-7801	$595.00 PCB Series II	optional $495.00	With toolkit $295	yes	no	$495.00 multi-pass multi-algorithm	included	included gerber	included with auto router	19 total (6 signal)	
Omation 1210 E Campbell Rd Suite 100 Richardson TX 75081 (800) 553-9119	$975.00 Schema-PCB	$495.00	yes	yes	yes	$750.00 ripup & retry	$300.00	$300.00	included	30 total	"Ease of use."
Generic Software 11911 N. Creek Pkwy S. Bothell WA 98011 206 487 2233 (Schematic Capture Only)	$99.95 CAD Level 1	included	no	no	no	no	no	no	no	0	"Both do schematic capture for OrCAD"
	CAD Level 2	$199.95	included	no	no	no	no	no	no	0	"Faster version snaps,trims, etc"

THE COMPUTER PROFESSIONALS' BOOK SOCIETY

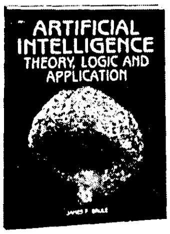
The easy, reliable way to satisfy your professional book needs . . . .

3 books for only $1.95 (Values to $162.40)

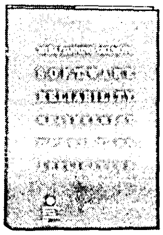
and get **Testing Computer Software** *FREE!*



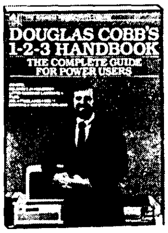
3019 $39.95
Counts as 2



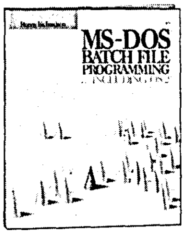
2671P $12.95



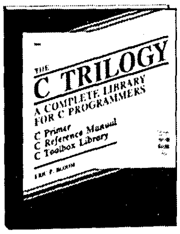
9824 $49.95


DOUGLAS COBB'S 1-2-3 HANDBOOK
THE COMPLETE GUIDE FOR POWER USERS

9779P $22.95

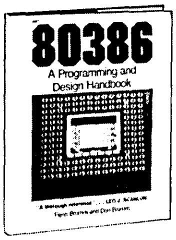


3028 $27.95



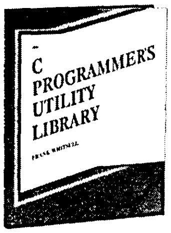
2890 $32.95

3016P $16.95



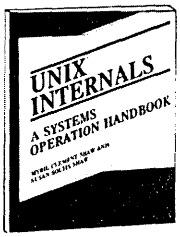
2937 $29.95

3026 $22.95



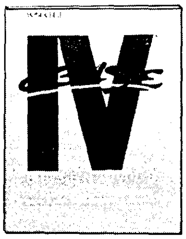
2855P $16.95

9808 $34.95

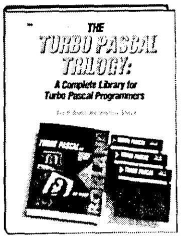


2951P $17.95

9829 $45.00

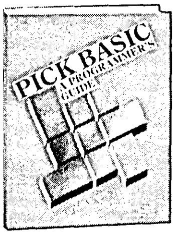


3116 $27.95


THE TURBO PASCAL TRILOGY:
A Complete Library for Turbo Pascal Programmers

3080 $34.95

2897 $27.95



2845 $29.95

How the Club Works

YOUR BENEFITS: You get 3 books for $1.95 plus shipping & handling when you join. You keep on saving with discounts from up to 50% as a member.

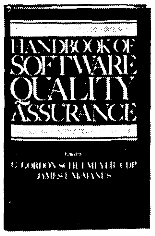
YOUR PROFESSIONAL BOOKSTORE BY MAIL: Every 3-4 weeks, you will receive the Computer Professionals' Book Society News describing the Main Selection and Alternates, as well as bonus offers and special sales, with hundreds of titles to choose from.

AUTOMATIC ORDER: If you want the Main selection, do nothing and it will be sent to you automatically. If you prefer another selection, or no selection at all, simply indicate your choice on the reply form provided. As a member, you agree to purchase at least 3 books within the next 2 years and may resign at any time thereafter.

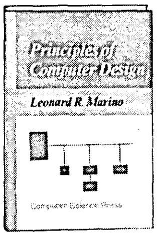
BONUS BOOKS: Starting immediately you will be eligible for our *Bonus Book Plan* with savings of up to 80% off publishers' prices.

IRONCLAD NO-RISK GUARANTEE: If not satisfied with your books, return them within 10 days without obligation!

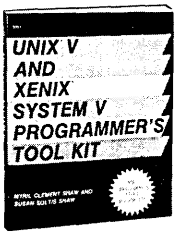
EXCEPTIONAL QUALITY: All books are quality publishers' editions especially selected by our Editorial Board. (Publishers' Prices Shown)



9786 $32.95



9813 $39.95

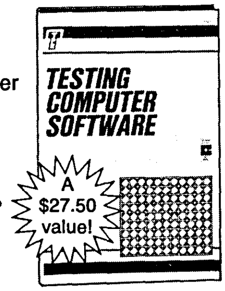


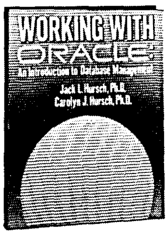
2751P $19.95

FREE when you join!

Testing Computer Software by Cem Kaner

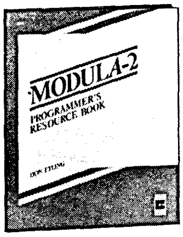
Describes and explores the role of software testing. Provides a realistic look at analyzing, isolating, and preventing bugs!





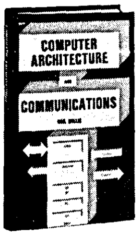
2916P $19.95

2838 $19.95

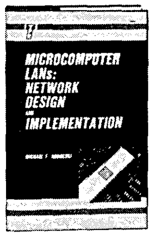


2741P $19.95

2620 $25.95



2870 $24.95



2888 $28.95

All books are hardcover unless numbers are followed by a "P" for paperback.

©1989 Computer Professionals' Book Society, Blue Ridge Summit, PA 17294-0870

Computer Professionals' Book Society
Blue Ridge Summit, PA 17294-0870

Please accept my membership in the Computer Professionals' Book Society and send the 3 volumes listed below, plus my FREE copy of *Testing Computer Software (2763),* billing me $1.95 plus shipping/handling charges. If not satisfied, I may return the books within ten days without obligation and have my membership cancelled. I agree to purchase at least 3 books at regular club prices (plus shipping/handling) during the next 2 years and may resign any time thereafter.

Name ______________________

Name of Firm ______________________
(If you want Society Bulletins sent to your office)

Address ______________________

City ______________________

State ____________ Zip ________ Phone ____________

Signature ______________________

Valid for new members only. Foreign applicants will receive special ordering instructions. Canada *must* remit in U.S. currency. This order subject to acceptance by The Computer Professionals' Book Society. MC189

Building Circuits With Your Computer:

An Introduction To Electronic CAD

The Micro C review team led by Bruce Eckel tries to review 3 1/2 CAD packages. What they discover, probably more than anything else, is the value of a 30-day money-back guarantee and the difficulty reviewing something this complex.

I'm going to walk you through the complicated world of Circuit Board Computer-Aided Design (CAD), and share our experiences with several commercial CAD packages. This isn't a comprehensive review, although at times I'll try to sound like a reviewer.

Instead, consider it a review/experience piece which tries to give you enough information to decide whether you can use a CAD system. Micro computer CAD systems, in my opinion, haven't reached the "state of the art" yet. They're neither cheap nor easy to use. But there are a few good ones out there if you can see your way through the jungle.

Once Upon A Time

I started this article 2 1/2 years ago, when I began writing for *Micro C*. Really. I desperately wanted a CAD system so I could design boards, but every one of the CAD systems was too expensive. So I wrote for *Micro C* instead, thinking that eventually I'd become famous (as opposed to rich), and could lure CAD companies into sending me their products to review for free.

Boy was I wrong. Reality struck when I mailed my first letters requesting review packages. Seventy percent of the companies completely ignored me (I even used *Micro C* stationery!). Fifteen percent sent me demos, which I ignored. Four companies sent complete systems.

So the review aspect of this piece includes four volunteers —

- OrCAD SDT III ($495) & OrCAD/PCB ($1,495) from OrCAD
- HiWire ($895) & smARTWORK ($895) from Wintek
- Cell, PCpro & PCroute ($600) from AMS
- The Unmentionable from ?

The Review Team

Circuit-Board CAD is complicated software, with lots of details to work out before you really can make use of a system. I didn't have the resources (or time) to learn four systems thoroughly, so I asked for help. Adam and Carl were both planning major circuit board projects for their research and for commercial applications, so they offered to lend a hand (actually they lent four, and we needed all of them).

Adam has a Ph.D. in oceanography and designs instruments for use on the ocean floor. Carl was well into a Ph.D. when he decided to become an adventurer instead. (I won't go into his adventures here, except to note that they'd make good stories in *Argosy* or *Omni.*) They're both designers and they're smart guys. And they can usually figure out how things (like CAD systems) work. When they can't figure out a system (but I'm getting ahead of my story), they tend to suspect the system.

The Design Process

Creating a printed circuit board on your PC begins with a "schematic capture." This consists of entering your diagram (usually with a mouse) into a graphics program especially tailored for electronic symbols. Most schematic capture packages have libraries of pre-defined symbols. If a symbol isn't in a library, you must create it yourself; complete libraries are very important.

Entering a diagram for the first time isn't always much faster than drawing it by hand. Once it's entered, however, you can make changes and print the diagram very quickly. This is incredibly handy when you're trying to find errors — the neatness of the diagram makes the bugs jump out.

Schematic capture programs usually generate output in two forms: as a diagram and as a "netlist." Most packages support Epson and other printers, at least one laser printer, and (often) more than one plotter.

A netlist is a file of text which describes the components in the circuit and the connections between the components (but not the physical locations you established on the diagram).

There are many different formats for netlists (everyone thinks they have a better idea of how to do it). It's important that your schematic capture program generate as many different types of netlists as possible if you want to interface the output to different types of PCB layout packages.

A Rat's Nest

Most PCB layout packages allow you to start from scratch and lay the board out as if you were taping it on a drafting board. You can also input the netlist, and the layout program will place the chips haphazardly with all the connections made as straight lines between chips (you must specify the package dimensions at some point in the process). This is the "rat's nest."

It's your job to place the chips where you see fit, and then put the traces down sensibly (yes, I know that sounds vague, but this is art!).

In the past few years, autorouting software has become more common — you place the chips, and the program puts the traces down (sometimes with a little help from you). Autorouters can save a lot of work, but they generally add to the price of a CAD system. A number of systems include non-optional autorouters (Wintek smARTWORK and OrCAD/PCB, for example).

When creating analog or high-speed digital boards, the PCB layout phase can

By Bruce Eckel (with Carl Haflinger & Adam Schultz)
Eisys Consulting
501 N. 36th St. Suite 163
Seattle, WA 98103

be very critical. Grounding, capacitive coupling between lines, and shielding can determine how much noise is in your system. And these, in turn, are strongly affected by board layout.

Autorouters (and many human routers) know nothing about good design rules for reducing noise. So someone who does know the rules must tweak the design.

Buy One Of These

I'd like to make something very clear before I go into the trouble and complications we had dealing with PCB CAD systems. If you design electronic circuits, you should have a CAD package.

It's arguable whether you can lay out a circuit by hand or with the computer more quickly (I think the computer wins). But once you print it out and look at it and decide to change something, the choice is obvious. There's nothing like making changes, viewing, and printing a pretty, unmussed drawing. If you're like me, you muss a lot when you work.

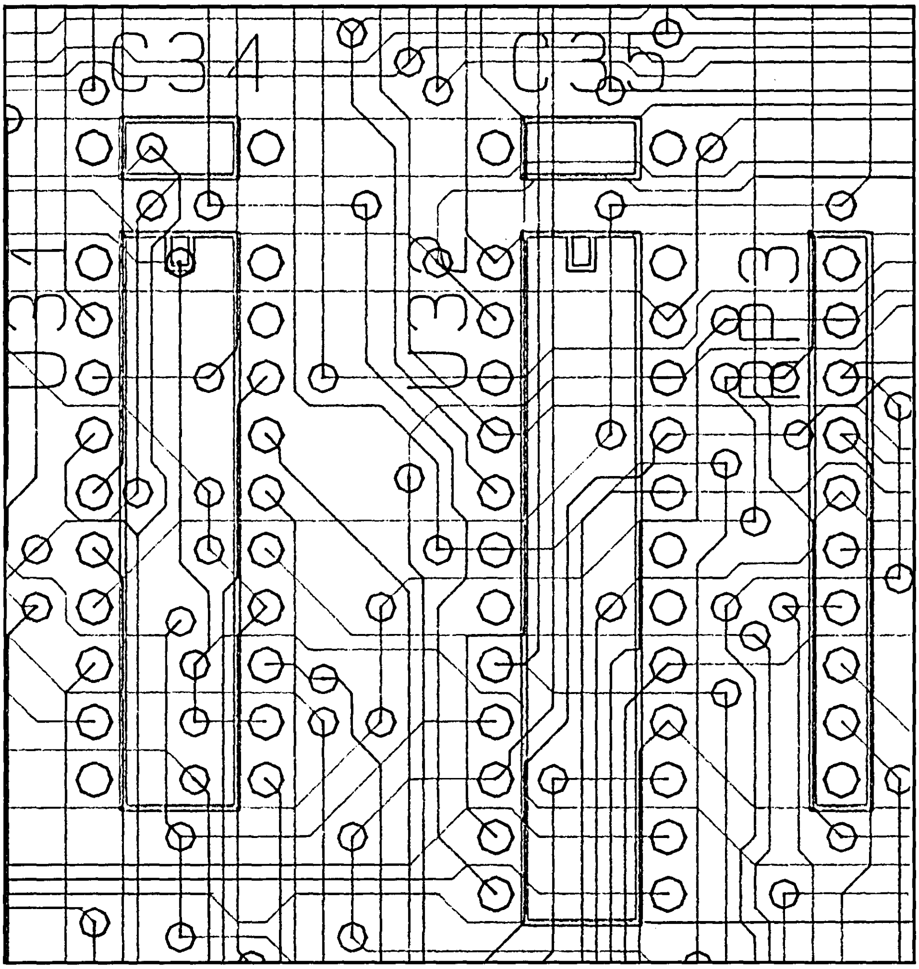
During board layout, you can see if something's wrong — there isn't a communication gap between you and a board layout person. The schematic goes from design to board layout under your control. This means —

- you can catch errors earlier,
- fix them yourself (correctly!),
- and get your board back without surprising problems.

And if you're a one-person operation, you save hiring a layout person.

Wintek

Wintek was the first company to respond, and I was thrilled to have their system. At first. Their schematic capture, HiWire, is very simple to use, and I created drawings quickly. But there were a lot of places where Wintek's designers opted for simplicity instead of functionality. And these began to bug us early in the review.



Connection Verification Plot

HiWire is a fine program, and if it were priced at $200 or $300 I would say, "you have to buy it." At $895, I'm not saying it. HiWire at $895 is just too expensive for what you get. If you decide to try it, make sure they support your output device. They don't support my LaserJet (only PostScript).

smARTWORK is their PCB layout program. When I chatted with Sam Azer (see his CAD review/experience article in this issue), he mentioned that he'd used smARTWORK and liked it. In his opinion, it was simple to use, worked great, but (again) had limited features, and was spendier than it should have been.

Unmentionable

I told all the companies I wouldn't mention their name if I didn't like them. But without mentioning the company's name, I can still tell you my experiences with a program which cost $4,000 and purported to do everything (schematic capture, simulation, PCB layout, and autorouting).

They sent me a package for review,

and not too many days later, billed me. I called and reminded them I was reviewing their program for a magazine, and would return the system when I finished writing. But it didn't stop the flow of bills.

Besides, Micro C'ers are usually looking for bargains (I know I am), and $4,000 for anything short of a system that writes my articles and washes my laundry probably isn't a bargain. But Adam argued that a more expensive program might be "more professional."

So he spent several days poring over the manual and trying to make it work on his Compaq 386 with VGA. Finally we determined there was no driver for the VGA. The company sent us a driver (and another bill), but it still wouldn't work.

As the threatening letters and the bills piled up, we decided we must not be professional enough for a $4,000 package and bailed out. We returned the "unmentionable" package unreviewed. So much for the theory that "more expensive (or 'more', in general) must be better."

AMS

The package from Advanced Microcomputer Systems (AMS) is usable, and it certainly has the best price around. It includes schematic capture (called CELL), board layout and autorouting for $600.

It would be best if you had at least an AT and preferably a 386 machine because scrolling in CELL is maddeningly slow. The program only supports Monochrome, CGA or Hercules — no high-resolution graphics.

You can design and build boards with AMS, but it's probably best suited for dabblers. If you're a professional (or a reasonable facsimile), be forewarned — you'll need lots of patience.

OrCAD

I'll try to suppress my enthusiasm for OrCAD. The general consensus was this: we struggled with the other packages, periodically cursing them for doing stupid things or making us work harder than we wanted. OrCAD SDT III was a relief — it was designed sensibly, and the designers worked very hard to save us effort and frustration.

There's a large bound manual, with a tutorial and lots of installation handholding. SDT III supports a wide range of printers and plotters, including Epson, PostScript and my HP LaserJet (as well as output for AutoCAD and Ventura Publisher).

SDT III supports a number of sizes on the LaserJet, and if the drawing is too big, it automatically prints it out on several sheets, which you can paste together and put on the wall. I really like that!

You can browse through the library and actually look at diagrams of the parts, instead of just numbers (in case you aren't a walking data book).

It generates netlist information in 26 other formats. So, if you prefer another PCB layout program to the one OrCAD supplies, you aren't locked into using theirs.

SDT III generates a nice parts list, contains a macro facility, and many of its logic symbols have a built-in DeMorgan equivalent conversion. You can automatically number labels on duplicate wires.

Carl summed it up: "Everywhere you look, you find something they've done which is really helpful."

SDT III is very professional and stands far ahead of the others in our admittedly small sample. It would be hard to recommend anything else.

Universal Deficiencies

I think all of these companies have made the assumption that the user already has some experience on another system. The manuals tell you what to do, but often omit the big picture. An introduction to CAD for novices would be most helpful. I know there are books on the subject (although they're usually not found in stores), so it could be accomplished.

Best Price/Performance

For schematic capture, the clear choice is OrCAD SDT III at $495. OrCAD's PCB package is terrific, but $1,500 might be out of your price range. Since SDT III outputs 26 different types of netlists (and is very popular), most other packages work with it.

Wintek's smARTWORK is easy to use, solves most problems, and only costs $895. Another company to check out is Tango — they never responded to my request for a review copy, but they have a layout program for $495 which seems to have a good reputation. Most of these places have some kind of money-back guarantee, so it might be worth trying a few.

The Continuing Search

This project turned out to be far more complex (and led to a much more incomplete picture) than I originally imagined — beginning with CAD companies' reluctance to let us review their packages. Neither Carl, Adam, nor I even got to the point of generating complete circuit boards.

This isn't a project I'm going to drop. (After 2 1/2 years, things are just beginning to pick up.) In my opinion, CAD is one of the most important things you can do with a computer (desktop publishing, if you think about it, is just another form of CAD).

I'd like to offer adapter cards for the PC to accompany my hardware projects in *Micro C*, but I'm not about to do it without a CAD system, so the search will continue. If you have a package you use and like, let us hear about it. If you manufacture a package, send it. If I make any discoveries, you'll read about it here.

Next Time

Reviewing is such hard work, I'm going back to the drawing board and magical hardware projects for the PC. "Turning an XT into a controller" — now that's something I can sink my teeth into.

Review/experience complete. What a relief!

Products mentioned:

OrCAD SDT III $495
and OrCAD/PCB $1495
OrCAD Systems Corporation
1049 S.W. Baseline Street, Suite 500
Hillsboro, OR 97123
(503) 640-5007

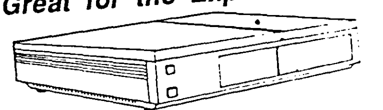
Wintek HiWire $895
Wintek smARTWORK $895
Wintek Corporation
1801 South Street
Lafayette, IN 47904-2993
(800) 742-6809

Advanced Microcomputer Systems, Inc.
Cell, PCpro & PCroute $600
2780 SW 14th Street
Pompano Beach, FL 33069

Editor's note: Bruce often refers to articles in previous issues. You can get a book of all his hardware articles, Computer Interfacing with Pascal & C, *plus a disk including the source code and numerous other examples, by sending a check for $30 (plus 8.1% sales tax in Washington State) to: Eisys, 501 N. 36th Street, Suite 163, Seattle WA 98103. Bruce is also putting together a library of C++ source code; disks are available at the above address.*

♦ ♦ ♦

Great for the Experimenter!



Pioneer Laser Disc Player

These units were removed from service for upgrades. They run on 100VAC from supplied 120VAC adapter. While they should be working, some may have minor problems so we must sell them on an "As-Is" basis.

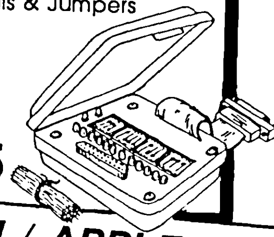
- 1-2 mW He-Ne Laser Tube
- Laser Power Supply
- 2 Front Surface Mirrors
- Two 1/2" Voice Coil Actuated Oscillating Mirrors
- One Beam Splitter
- Two Optical Lenses
- One Optical Detector
- Mini Gear Reduction Motor
- All Controlling Electronics
- Assorted Switches, Fan, Solenoid

THESE UNITS ARE OEM MODELS AND REQUIRE A COMPUTER CONTROLLER (NOT AVAILABLE FROM HSC). THEY CANNOT BE MANUALLY CONTROLLED! **$99.00**

NEW!

RS-232 BREAKOUT BOX

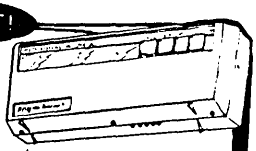
- Switchable Lines
- LED Indicators
- Patch Terminals & Jumpers
- Compact Size



$31.95

CLOSEOUT!

DISPLAY PAGING RECEIVER



- Dual Conversion Superhet 450 MHz Crystal Controlled Receiver Module (Plug-In!)
- Twenty-Char. Alpha-Numeric LED Display (ASCII Encoded, ANSI Char. Set)
- RCA CDP 1802 Based
- Peizoelectric "Beeper" Unit
- Vibrating "Silent" Alert
- Untested - "As-Is"

$9.95 each or two for **$17.95**

IBM / APPLE SOFTWARE BLOWOUT!

IBM TITLES		APPLE TITLES
Visi On Graph	VisiSchedule	FlashCalc
Visi On Calc	Applications Manager	VisiCalc
Visi On Word	Desktop/Plan	VisiLink
~~VisiCalc~~ SOLD OUT!	VisiFile	VisiDex
VisiWord	FlashCalc	VisiTutor for VisiCalc
VisiSpell	VisiAnswer	VisiCalc (Advanced)
VisiTutor for VisiWord	VisiTrend/Plot	VisiFile
StretchCalc		

* **VisiCorp Closeouts** * **Priced from $6.95 to $12.95**

* Mouse and Basic required for some programs

COMPUTER CHASSIS with POWER SUPPLY

These attractive system chassis were manufactured by Televideo for the TS806/20 Computer System. They are brand new and include the following features:

- Heavy Duty Plastic Case
- 17" x 17" x 8" O.D.
- Hinged Drive Mounting Assembly for 2 Floppy Drives and 1 Internal Hard Drive
- +5 +12 -12 Power Supply
- IEC Receptacle, Power Switch
- Fan

NEW! IN ORIGINAL BOX! **$69.00**

Mitsumi UVEX-AW51P

UHF / VHF VARACTOR TUNER

* 3 1/2" x 2" x 1/2"
* Pinout on Case

TWO FOR **$17.95**

HALF HEIGHT EXTERNAL DRIVE ENCLOSURE

- Attractive Low Profile Case
- 19" x 15" x 3" O.D.
- Fits nicely directly under PC
- Standard IBM Colors
- Bezel fits One 5 1/4" and One 3 1/2" Drive Only

$29.95

SPECIAL!

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

300W, 2500W, 5000W also available

TEAC FD-55B DISK DRIVE

* 5 1/4"
* IBM COMPATIBLE
* 360K
* 90-DAY WARRANTY

$89.95

*ENGINEERING! * TEST! * DEVELOPMENT!*

AT/XT 3-SLOT MOTHERBOARD EXTENDER

- Fused Extender Card
- One 16 Bit Slot
- Two 8 Bit Slots
- Test Points for All Bus Points
- Power Connector
- Cables Included
- Not an Expansion Chassis!

$89.95

WE SHIP C.O.D.!

Call the HALTED ELECTRONIC RESOURCE BBS! (408) 732-2814

HSC of Santa Clara
3500 Ryder Street
Santa Clara, CA 95051

VISA

AMERICAN EXPRESS

HSC of Sacramento
5549 Hemlock St.
Sacramento, CA 94928
(916) 338-2545

HSC of Santa Rosa
6819 Redwood Drive
Cotati, CA 95841
(707) 792-2277

HSC *Since '63!*

Call Now! *Outside California* **800-4-HALTED** *California Residents* **408-732-1573**

TERMS: Minimum order $10. California Residents add 7% sales tax. Prepaid orders sent 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 orders over $100 use money order or certified check. Please no not send cash. Some items limited to stock on hand. Prices subject to change.

Reader Service Number 11

Secrets Of Compiler Optimization

Inside The Zortech C And C++ Compilers

So you're finally writing real C. I mean C with those short, opaque, programs. Whole utilities on one line. (And you're doing it without cracking K&R.)

Well stay tuned as Walter Bright (author of Datalight C, Zortech C, and Zortech C++) explains how to optimize your shorts.

Fortunately for programmers (and users), intense competition among language vendors has produced a new crop of compilers. Very special compilers that use aggressive, sophisticated optimizers to improve performance.

In this article, I'll concentrate on the design and implementation of a sophisticated optimizer: the one I've written for the Zortech C and C++ compilers.

I'll discuss optimizing technology with an emphasis on the engineering design tradeoffs necessary to produce a practical compiler. In other words, I want to stress the importance of developing compilers (and optimizers) that function well in the everyday world, as opposed to those intended for research.

I'll assume you know C and know how to use a compiler. I'll aim to help you make the most effective use of optimizers.

There have been many recent magazine articles "benchmarking" the new optimizers. The articles have frequently done disservices to the compilers by using them incorrectly or drawing erroneous conclusions from the results. I'll explain how to tell a good benchmark from a bad one.

Optimizers, being very complex creatures, frequently cause the generated code to produce unexpected, and sometimes bizarre, results. I'll explain how to tell the difference between bugs in the optimizer and bugs in your program.

So let's get to it.

Who Needs One Anyway?

It's true, a programmer can do most of the optimizations performed by a global optimizer. It takes time, however, to go through a function and tune it for maximum performance. If the program is large, let's say 100,000 lines, tuning every piece is a major undertaking. Contrast that with simply turning on the optimizer and going to lunch.

Human optimization also has other drawbacks. The goals of code readability vs. portability vs. maximum performance are often at odds with each other. Even worse, a function carefully tuned for one compiler and one architecture may be quite unoptimized for another compiler or architecture. So it's best usually to concentrate on the algorithm and let the optimizer worry about the tuning.

C is being increasingly used as the output of other programs. It's often easier to generate C than to generate object code. Notable examples include YACC, some C++ translators and Lotus 123 to C translators. Having an optimizer simplifies the task of the C generator, as it can generate code in a simple straightforward manner and let the optimizer worry about fixing it.

Optimizers Vs. Assembly Code

I've rarely seen any instances of optimized code that couldn't be improved by a good assembly language programmer, frequently by 50% or more. One reason for this is that the programmer knows a lot more about the expected values for the variables, so he doesn't have to always make worst case assumptions.

Also, the more complex the instruction set, the more the assembly language programmer can do with clever instruction sequences. The 8088 family of CPUs has a large number of idiosyncratic opcodes, so there are many opportunities. By contrast, the 68000 has a more regular instruction set so it's easier for the compiler to generate the best instruction sequence.

But turning a critical function into optimum hand-coded code is very expensive. Flow analysis optimizers can frequently get about half the gain you'd get by hand coding.

That's why the C runtime library routines are usually written in assembler — there's a high payback for the time invested. Not only will the customer's programs run faster, but the compiler will too!

Assembly language code also loses big when the time comes to port a product. You'll have to rewrite the code for the next architecture. C code is usually just recompiled, and you don't have to learn all the tricks of the new architecture. It's the compiler writer's job to do that.

An optimizer can also exploit the relationships it uncovers, those most experienced assembly language programmers would shy away from due to the complexity and delicateness of the result. This happens most frequently in very large functions.

Design Goals & Constraints

Numerous goals and constraints define the design of the optimizer in order to ensure that the result is truly a useful product.

The most obvious constraints are those of the PC and of MS-DOS —

- the program must run comfortably in 640K;
- it can't depend on virtual memory;
- it must run comfortably from floppies (no thrashing);
- it can't take an interminable amount of time to run on a 4.77 MHz 8088.

Generally, optimizers have a reputation for producing incorrect code. So a major design goal is reliability and absolutely correct optimizations.

By Walter Bright
13928 129th Ave. NE
Kirkland, WA 98034

The optimizer must follow the language rules to the letter (in this case the ANSI draft standard for C). It must avoid any "unsafe" transformations and always assume the worst will happen.

The optimizer should catch 90% of the easy possibilities for optimization. Then presume that catching the remainder would greatly increase optimizer size, complexity, and problems.

Machine and language independence are important, as the language won't always be C and the target architecture won't always be the 8088. I don't want to be trapped into any one machine; look what happened to CP/M, the TRS-80, etc. In fact, there are signs that the heyday of the 8088 architecture is drawing to a close.

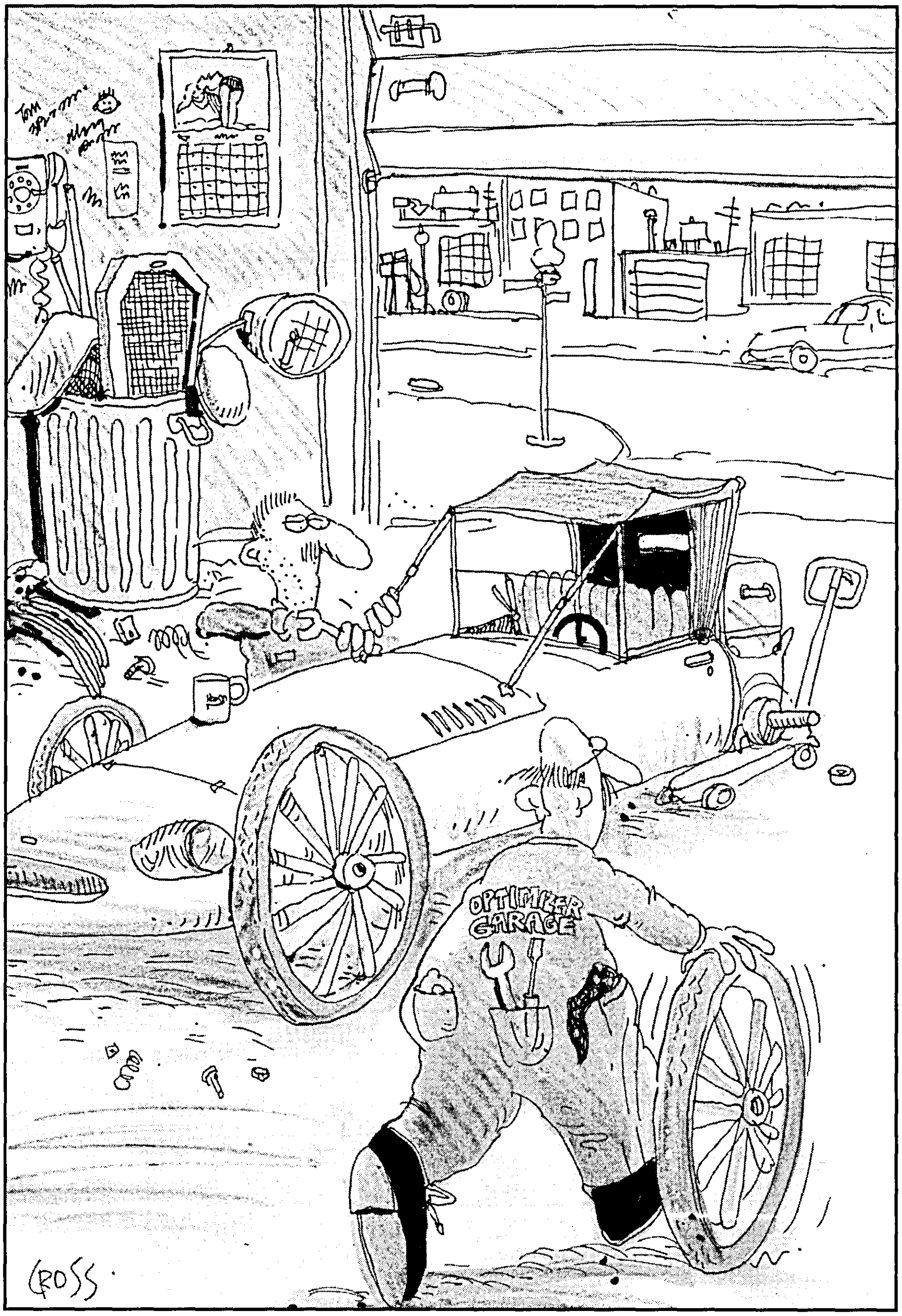
Aggressive optimization can be very time-consuming, obviously making the edit-compile-debug cycle unacceptably long. Therefore, the compiler should work just fine without the optimizer. The programmer needs the choice of quick compiles for development, only running the optimizer on the final version.

But running the compiler without the optimizer should still produce reasonable code, since it's unacceptable for the unoptimized code to be so large it won't fit in memory (and therefore can't be debugged).

In keeping with the philosophy that the optimizer should relieve the programmer from dealing with tuning details, the optimizer should do a reasonable job of register allocation: so the register keyword becomes irrelevant.

The debuggability of optimized code has always been seriously impaired by the addition, deletion, and rearrangement of code, and also by the addition of new variables and removal of old ones.

Since this is an unsolved problem, I decided not to let concerns about debuggability compromise my optimizer's optimizations. The optimizer should try to carry along things like source line



Model A Optimization

numbers. But if the line numbers get hopelessly scrambled due to code motion, it shouldn't worry about them.

Many existing optimizers will only optimize loops if the loop is written with the for keyword. If there are any gotos, optimizations are abandoned. A practical optimizer should be able to handle any scramble of gotos, fors, whiles, dos, etc. The loops should be detected no matter how they're written.

This is important because it may be easiest for C code generators to simply output a sequence of ifs and gotos. Standard algorithms exist to solve this problem.

Sometimes problems in the source code can be discovered as a result of flow analysis. Some examples are variables that are used before they're set, and variables that are never used.

When it sees this problem, the optimizer should holler: but not with "Possible use of variable before it's set." After all, it has irrefutable evidence.

Design Decisions

I used a three-pass design. The first pass is the parser, the second is the optimizer, and the third is the code generator. This design has numerous advantages:

- The parser is the only part of the system that is language-specific. It can be replaced with one for a different language, let's say Fortran, with little or no effect on the subsequent passes. In fact, the difference between my C and C++ compilers is simply a different front end!
- A similar advantage applies to the code generator. The current code generator for the 8088 could be replaced with one for the 68000 with negligible changes to the parser and the optimizer.
- The optimizer is optional and language independent. The parser or code generator can be replaced with no effect on the optimizer. Many other compilers implement the optimizer as a pass operating on the output of the code generator.

Placing it there, however, seriously cripples it as much information is thrown away as code is generated. Placing the optimizer right after the parser gives it full access to the symbol tables, aliasing, scope, and type information.

In order to maximize I/O performance, the optimizer should read its input file forwards only (i.e., it shouldn't be thrashing forwards and backwards through the data). Most disk systems perform best when reading file data sequentially.

Most CPUs are faster doing register operations than doing memory operations. Thus, the best results lie in making effective use of registers. Since the optimizer operates on the code without knowing about the target machine, it can't allocate registers directly. It can, however, provide live range information to the code generator so it can easily allocate registers.

> Placing the optimizer right after the parser gives it full access to the symbol tables, aliasing, scope, and type information.

The optimizer operates by reading in all the information about one function, optimizing that function as a whole, writing that function to the output file, and then proceeding to the next function.

Compiling

Here's a quickie overview of how the compiler works.

In the first pass, the compiler reads the characters and divides them into tokens. Macro expansion, parsing, and semantic analyses are all concurrent processes (i.e., occur in one pass). The resulting symbol table and expression trees are written to the intermediate file.

The second pass (the optimizer) reads the intermediate file, one function at a time. Each function is optimized as a whole, and then written to a revised intermediate file that is essentially in the same format as the first one (thus the optimizer pass is optional).

The third pass generates code. It reads and processes the intermediate file function-by-function. If you bypassed the optimizer, the code generator performs some simple optimizations. If you ran the optimizer, the code generation routines are called immediately.

Expressions are stored internally as binary trees, not as three-address code or quads. The code generator calls a hand-coded ad-hoc routine for each type of expression node. This results in a very fast, but rather kludgy and inflexible code generator.

After selecting the instructions, the code generator takes a look at register usage, compares this information with variable usage, and, where appropriate, assigns variables to registers. (Unlike other compiler writers I don't use a peephole optimizer.)

After generating the code and finalizing variable offsets, the compiler writes out the object file.

Optimizing

Tree optimizations are the simplest and most straightforward since they require a minimum amount of information. Nearly all compilers "do trees" to some degree. They involve looking at small pieces of the expression tree and changing them. Some examples are:

```
(a * 0) becomes: (0)
(a * 4) becomes: (a << 2)
(a + (3 * 6)) becomes: (a + 18)
```

I have a very long, long list of these kinds of things (including a bunch which are specific to the C++ parser).

Jump optimization is fertile area. For instance, I replace jumps to jumps with a single jump.

I replace variables with constants when flow analysis determines that the value of that variable at that point is always that constant. For example:

```
V = 5;
func(V);
/* V can be replaced with 5 */
```

Copy propagation is similar to constant propagation, except that:

```
V = W;
func(V);
/* V can be replaced with W */
```

Sometimes all uses of V will disappear, and V can be eliminated.

Dead assignment means that a variable is assigned but the result is never used. Flow analysis easily determines a variable's uses and a variable that's not used gets deleted. Dead assignments are one of the most frequent reasons why good optimizers "optimize away" many benchmark programs.

The live range of a variable is the section of code over which the value of a variable needs to be retained. The live range is extremely useful. Knowing it, we can determine if two or more variables have nonintersecting live ranges, and so can share the same register. This is called register allocation by coloring, and is done by the code generator. The optimizer provides the live range information.

A by-product of live range analysis is the detection of dead variables. Dead variables are variables that are never used, so storage needn't even be allocated to them.

I don't warn the programmer about dead variables because many of the variables have already been replaced by the optimizer. So my warning messages would appear very mysterious!

Dead code is simply code that can never be executed. So we remove it.

Common subexpressions are redundant computations that happen most frequently in array address calculation.

For example, we can replace —

```
struct ABC v1[5][3];
v1[i][j] = func(v1[i][j]);
```

with —

```
struct ABC v1[5][3],*p;
p = &v1[i][j];
*p = func(*p);
```

Loop invariant removal means I move expressions from inside a loop to outside the loop. For example —

```
for (i = 0; i < 10; i++)
  abc[i] = b * c;
```

can be rewritten as:

```
T = b * c;
for (i = 0; i < 10; i++)
  abc[i] = T;
```

Loop induction variable elimination is one of the most ambitious optimizations. The basic idea is to replace loop indices within loops with pointers. We could rewrite the above expression as —

```
T = b * c;
i = 0;
for (P = &abc[0]; P < &abc[10];
     i++, P += sizeof(abc[0]))
  *P = T;
```

We've removed an add and a multiply from inside the loop (the multiply comes from converting i to an offset from the start of abc[]).

Most of the above uncover possibilities for more optimization. For example, when we replaced i with P, i is obviously no longer needed and so the assignments to i are probably unnecessary (i.e., dead). Thus, the optimizer repeats all its optimizations until no more are possible.

The problem to watch out for, of course, is mutually inverse transformations. In this case the optimizer would forever flip back and forth between two states! (This bug occurred in a couple instances and was rather difficult to correct.)

Global variables suffer from so many obscure aliasing problems that under most cases, you simply can't optimize them. (It's preferable to avoid globals as much as possible.)

And, after all this, the real test. The compiler must successfully recreate itself.

Implementation

I started out armed with numerous papers on how to optimize, and of course the famous dragon book by Aho and Ullman. Most other compiler books are fairly useless. They typically devote less than one chapter to optimization or gloss over it in a cavalier manner.

I implemented all the algorithms per the paper (after extending them, of course) but the algorithms didn't cover things like multiple types, and tended to ignore aliasing problems. And, unfortunately, the algorithms for loop induction variables were seriously deficient.

Normally, a loop is written using a signed integer as the loop index. Pointers, on the other hand, are unsigned. They also have a different range. So I started seeing serious bugs when I began optimizing loops —

```
int i;
for (i = 10; --i >= 0;)
  abc[i] = v;
```

Replacing it with a pointer yielded —

```
int i,*P;
for (P=&abc[10]; --P>=&abc[0];)
  *P = v;
```

Hmmm. P can never go negative. Also, in segmented pointer models, P can never be less than &abc[0] (segment wrap will occur), so the loop will never terminate. More problems occur if i never reaches the termination condition (i.e., it skips past it) as in —

```
for (i = 10; i > 2; i -= 7)
```

Even more subtle is a problem at the opposite end of the range if i wouldn't overflow, but P would —

```
for (i = 0; i < 32767; i++)
  abc[i] = 2;
/*problems if sizeof(abc[0])>2*/
```

The solution is to determine all possible values of i within the loop. That's the only way to know if optimization is safe.

Is It Always Optimization?

It's possible to optimize like crazy and wind up with larger and slower code. How can this be? The trouble turns out to be addressing modes. A balance must be struck between the addressing modes available on the target CPU and the number of uses of each mode.

For example, the 8088 can add constants to index registers quite efficiently, so replacing —

```
v = abc[i] + 3;
abc[i] = 4;
```

with —

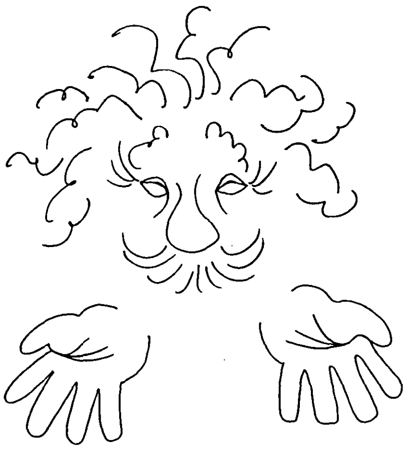
```
v = *(P = &abc[i]) + 3;
*P = 4;
```

is a net lose. But if abc[i] is used more than a certain number of times, it becomes a win.

A similar problem occurs with register allocation. If a register must be saved across function calls, and a variable is assigned to it, it must be pushed and popped.

If the variable is used only once or

Reader Service Number 107



twice, the expense of the push and pop negates any gain. Four uses of the variable makes it worthwhile to allocate it to a register (even when it must be saved across function calls). Of course, if it's okay to destroy the data when there's a function call, then there's no cost to assigning that data to a register.

Variables are assigned to registers based on how often they're used. Usages of variables inside loops are given a weighting of 10 (I assume all loops will execute an average of 10 times). Thus, the most heavily used variables, and the ones inside loops, get allocated to registers first.

C has a very relaxed attitude on types. A lot of C's power and flexibility results from this. In keeping with the design goal of correctness and reliability, the optimizer assumes the worst cases:

- if a store is made to a static or a global variable,
- if a function is called,
- if an assignment is performed through a pointer.

Then all globals, statics, and pointer references are presumed to have been modified. Occasionally, this results in worse code than that generated by other compilers (shudder!).

It's true that only rarely does aliasing cause trouble for these compilers. But when it does, the whole optimizer concept gets a bad reputation. After all, who wants an optimizer that breaks a perfectly correct program.

Optimizers that relax aliasing seldom explain their assumptions in the manual. So it's hard for the programmer to know exactly what to avoid. I prefer to sacrifice a little optimization for improved reliability.

Global variables suffer from so many obscure aliasing problems that under most cases, you simply can't optimize them. (It's preferable to avoid globals as much as possible.)

Recently, I saw a benchmark program where nearly all the loop index and temporary variables were global. Naturally, the optimizations couldn't be performed on them because of worst case aliasing assumptions. The author concluded that the optimizer (in this case mine) didn't do any optimization! But when the benchmark was rewritten with the variables made into automatics, all the expected optimizations showed up.

As further fuel to the jihad against global variables, programmers writing multiple-thread code for OS/2 will discover that they can't use many global variables, as it would make their routines non-reentrant.

I coded the entire compiler making heavy use of assertion checking. Assertion checking is using the standard assert() macro to verify that the values of variables are what I expected.

For instance, I check for null pointers and I verify that an enum variable really contains an expected enum value. More than half of the bugs have shown up during the assertion check!

Optimizing And C++

The optimizer proved to be an unexpected boon with C++. Much of what C++ does is add code. For instance, it adds code to constructors for storage allocation. It adds code to the destructors for storage deallocation. Then the optimizer can remove any redundant portions of the new code.

It really comes in handy when dealing with the expansion of inline functions. Inline functions are supposed to be a speed improvement over directly calling the function. The semantics for inline expansion, however, require —

```
inline int func(a,b){return a * b;}
...
c = func(d,e);
```

to be equivalent to:

```
c = ((a = d),(b = e),(a * b));
```

which is hardly an improvement. But when optimized (using copy propagation, dead assignment elimination, and dead code elimination), the above expression becomes —

```
c = d * e;
```

which is more what we had in mind.

I do the first expansion just in case d or e is an expression, or in case func modified a or b. Use of the dumb (but correct) expansion, followed by the optimizer, results in both correct and efficient code.

Intrinsic functions are similar to inline functions. The difference lies in the fact that the inline expansion is built into the compiler, rather than being defined by the programmer. There is special hand-coded support for them in the code generator. I/O port functions are one example.

There's no C syntax to read and write I/O ports, so they must be accessed through functions. There is, however, direct CPU support for I/O ports, and it seems a shame to have to call a function to perform one special instruction. Also, the function call overhead to access an

I/O port would impose an unacceptable speed penalty for many applications. So I have the compiler recognize I/O port functions and generate special code for them.

For the I/O port functions, this is a big win. For several projects I know of, this has made the difference between using C and having to use assembler. Of course, where do you stop? You could carry things to absurdity by having all the standard C library routines implemented as intrinsics.

The tradeoff comes down to: time to implement and possible bugs introduced in the compiler, versus the real improvement to the user. (And I don't care to make an intrinsic out of every function that appears in popular benchmarks!)

Currently, only the I/O port functions are intrinsics. Future candidates include memcpy() and the functions to set/clear the interrupt mask.

Experience

Experience with the optimizer leads to some conclusions. First of all, I/O bound functions remain I/O bound so they show no improvement when optimized. Second, functions that consist mostly of a list of calls to other functions show little improvement. Obviously I need to reinvestigate (and improve) function calling.

Functions where the optimizer manages to get most of the variables into registers show the most dramatic improvements. Clearly, proper register allocation is of major importance to optimization. Types of code that the optimizer works best on are functions containing loops that do lots of integer manipulation.

Unrealistic Expectations

From straightforward code generation to "why can't a clever optimizer figure it out" is a small step for mankind.

Nothing can improve the speed of a program more than selecting a fast algorithm. For example, the most heavily optimized bubble sort program will lose out to a suboptimal quicksort. Do everything possible to improve the algorithm before falling back on optimization.

The optimizer isn't a mind reader. All it has to go on is the source text of the program.

I've seen several ways to help an optimizer detect where aliasing was possible and where it wasn't. The trouble is, this is currently very impractical to implement. Both the memory and the CPU speed are still too limited and the necessary algorithms are still Ph.D. research topics.

Benchmarks

Many benchmarks suffer from being "optimized away"; that is, the optimizer discovered that the code produced no useful result and removed it. Typical causes of this are dead assignments. These benchmarks don't really measure true speed. That's because optimizers are written to optimize real code, not the endless empty loops and throwaway variables that make up most benchmarks.

A way to prevent this is to have the benchmark function return the result or assign it to a global variable.

Benchmarks should minimize the use of global variables, since globals make up only a small percentage of variables in a real program. Automatics are the most common type.

Benchmarks should be carefully scrutinized to make sure they conform precisely to accepted programming practice and the ANSI C standard.

Making The Most Of The Optimizer

It's fairly easy to arrange things so that the optimizer has the best chance of improving your program. The key factor is maximizing the amount of information that it has. This is accomplished by:

(1) Make variables local. This means declaring a variable in the innermost block possible. Minimize the use of global variables.

(2) Use function prototypes. This will be of paramount importance to compilers that are capable of passing arguments in registers.

(3) Organize the modules so that as many functions as possible can be made static. The compiler knows all the calls to static functions, and can do many more optimizations because it is freed from the constraints of the function call interface.

(4) Since the optimizer compiles by moving forward through the file, place lower level functions first in a file followed by the ones that call them. In other words, if the function calls form a tree, put the leaves first, followed by the branches, and then the trunk.

The Future

It's entertaining to optimize a function, and then write it in assembly code. Compare the two, and try to figure out why the hand-coded assembly is better than the machine-generated. Those are the areas where the optimizer needs improvement. The improvements I see are:

(1) Better handling of function calls. Most assembly language programs pass arguments in registers. It's very wasteful to push them on the stack. The compiler should be able to figure out how to pass arguments in registers.

(2) Flow analysis could be extended to registers, resulting in much better use of registers.

(3) More memory is becoming available with OS/2, and more processing power with the 80386 machines. This means it's becoming more practical to attempt optimizations that require real horsepower.

Optimizers are here to stay. Increasing competition from vendors, steadily improving reliability, and quality will ensure that. If you're a smart programmer, you'll get to know your optimizer and take full advantage of it.

♦ ♦ ♦

Finding Bargains in the Surplus Markets

You're tired of plugging together tinkertoy clones? Want to break out the soldering iron and really build something? Well, the surplus market is where you'll find cheap ICs, boards, connectors, cables, drives, power supplies, everything. (Well, almost everything.) Here's the story.

For the hardware hacker on a low budget, the surplus marketplace offers real bargains. With some planning and expertise, you can find quality tools and equipment for very few dollars.

And if you're just getting started in hardware, borrow the expertise of a friend when you go looking. Don't let inexperience deter you from enjoying a little bargain hunting.

You'll save yourself much time and trouble if you understand what's available and where to look for it. The surplus market divides up into two areas; mail order and local surplus stores. The kinds of merchandise found in each span the range from brand new to totally worthless.

What Is Available?

Don't make the mistake of equating "surplus" with "junk." Surplus equipment is simply equipment no longer useful to its current owner. A manufacturer may order 2000 top-quality power supplies, only to have the product to be built with those supplies suddenly discontinued. The supplies are now surplus.

Power supplies, floppy disk drives, computer terminals, line printers, modems ... I've seen them all available as surplus at very low prices. If you're looking for this type of overrun surplus, try to find items in the original packing boxes, if possible, with incoming QC inspection slips proving that the item truly works.

Sometimes a large company upgrades a range of equipment, sending the replaced material out to brokers who sell it to surplus shops. Often, the replaced equipment works just fine. Items that fit this category include computer terminals, hard drives, floppy drives, office furniture and fixtures, and line printers.

Don't make the mistake of equating "surplus" with "junk." Surplus equipment is simply equipment no longer useful to its current owner.

You're probably most familiar with the surplus equipment to be found at swap meets and hamfests. Here, someone sells a few unique items at whatever the market will bear. You (usually) get to talk to the owner about the item you want and may be able to negotiate the price.

Some surplus houses take a large piece of bizarre equipment (for which they would likely not find a ready buyer) and take it apart, selling the pieces for whatever they can get. If you're lucky and patient, you can find the odd pump, valve, hardware, or chassis that's just what you've been looking for.

Then there's the real junk. This stuff lays around "out back," gets rained on, coated with dust and rust. It may be printed circuit boards (PCBs) that are broken or fried. Even here, you can find real bargains if you're careful. Some PCBs go from a service center to the surplus shop as non-working, when in reality the service contract expired and the manufacturer simply didn't want them back. You can part these boards out yourself if you want, or perhaps even get them running again.

Buying Surplus Through The Mail

Through mail order, the all-time best place to shop is the pages of *Computer Shopper*. This inch-thick, large-page magazine mixes several computer articles with a mind-boggling array of advertising. You can find *CS* at many of the larger bookstores, newsstands, and electronic shops. *CS* also sells subscriptions. The staff at *CS* tries hard to screen their advertisers and works to make sure all their mail-order customers get fair treatment. I have ordered from the pages of *CS* many times, and always received prompt delivery and fair value.

Certain mail-order companies rate high marks for service and selection. DAK Industries, for example, offers a glossy color catalog, fast delivery, and an excellent return/refund policy if your item doesn't work. They offer overrun surplus, specializing in consumer electronics and some computer items such as printers and software.

Jameco Electronics, JDR Microdevices, and California Digital have been advertising in the pages of *BYTE* magazine (and others) for years. Each offers a wide range of overrun surplus and new items, backed by prompt delivery and good prices. I have dealt with each of these companies and always received satisfaction. Write to them for a catalog or flyer.

Other companies deserve mention as sources of good-quality surplus equipment. HalTed Electronics (lasers, computer components and electronic assemblies), Tri-Tek (computer components,

By Karl E. Lunt
7349 W. Canterbury
Peoria, AZ 85345

electronic assemblies and new test equipment), and Computer Parts Galore (PC/Apple boards and surplus computer components); all have given good service and prompt delivery.

Other Sources Of Surplus Equipment

It's always possible to find surplus bargains other than through the mail. Almost everyone knows a friend with a ham radio license; the two- and ten-meter bands usually carry informal swap meets on the air. If you have such a friend, drop by for a visit and listen in.

Many of the larger electronic and industrial businesses employ electronic technicians to keep the equipment running. Such people are usually gold mines of information as to company surplus, sources of cheap electronic parts, swap meets, etc.

The classified section of the local paper often contains bargains in the "For Sale" columns. Less obvious places to look for similar deals are the "For Sale" ads in the major computer networks such as GEnie. With a PC, modem, and a few dollars for registration, you can gain access to a national classified section.

The author sorts through the bins of surplus parts at Tri-Tek, looking for just the right component to finish off a project. Many of these bins are filled with overrun surplus items such as lamp assemblies, switches, and connectors.

The local university, college or technical school can provide information and contacts. The students, especially, of such places have proven to be valuable resources. In many cases, these same students can lend technical help or after-hours access to high-quality debugging and testing gear.

Sometimes contacts in one field lead to real savings in a totally unexpected area. A technician friend of mine told me of a local company that had developed a nifty new process for salvaging ICs from discarded PCBs. Upon calling the company, I learned they sold the pulled chips for 10% to 20% of retail. Although they now only deal in high volumes (pricing the hobbyist out of a bargain), I once repopulated an entire Apple II+ for only $25 using their chips.

Real Surplus Shopping

But you have to actually walk into a computer surplus shop and see the shelves full of equipment before you can truly appreciate the bargains available to the surplus shopper.

As an example of what you can find surplus, my current computer system consists of a 68010 CPU board ($40), power supply ($30), PC-style floppy disk drive ($10) and TeleVideo terminal (free from a friend). True, I put a little work into getting the 68010 board running, but I learned a lot and enjoyed the experience.

Before you go anywhere near a surplus shop, take some time to decide clearly what you want. You don't save money buying something you don't need or can't use, regardless of how good a bargain it is.

Suppose you're looking for a power supply for a small computer project. What is the largest size the supply can be and still fit its cabinet? What voltages and currents must it provide? Is there a problem with any heat given off by the supply? Do you need a linear supply or a switcher? Do you have the extra hardware (line cord, plugs, connectors, switches, etc.) that you will need? Believe me, time spent considering such questions beforehand is really money saved.

Be flexible in your approach to a problem. Once, I was hunting for a power supply for a small 6809 system I was building. I knew I needed +5, +12 and -12 volts for the system, but all the three-voltage supplies I found cost more than I wanted to pay. Then I realized that the current requirements for the -12 voltage were so low (just a few milliamps) that I could get by with a +5/-12 volt converter. While this was not a cheap part, I ended up saving money because then I could use a low-cost floppy disk drive supply for the other two voltages.

Often, you have several items you need or have been thinking about. Draw up a list of these items, with notes to remind you of special concerns. Try as you might, you will not be able to remember everything when you're standing in the store. Save yourself some trouble and write the details down.

When you finally get to the store, force yourself to first go down your list, looking for each item in turn. Usually, you can find more than one solution to a need; take the time right then to decide which item fits best. Of two possible solutions, give preference to the item with the most information (manufacturer, part number, revision code, connection information, ratings, etc.) stamped on it. You may end up having to write someone for help getting your "find" working.

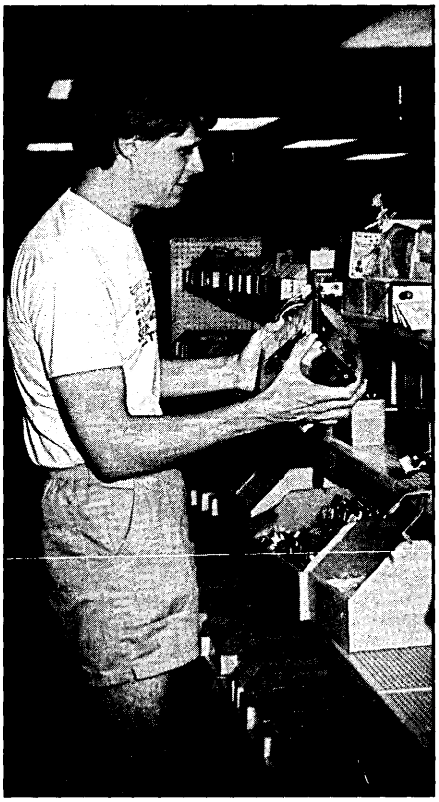
There are other points to consider as you judge an item. Give preference to overrun surplus; even though you may pay a little more for an item in its original packing, you have a better chance of getting a working unit. If your budget is limited, stick to those items with immediate use; money spent on a "rainy day" project is money diverted from your current project.

If you want to buy a PCB, look it over carefully for cracks in the board itself; while you might be able to repair a cracked double-layer board, a damaged multi-layer card is virtually unrepairable. Always look for the latest revision of a board; go through the stack carefully and check the manufacture date and revision codes stamped on each board. Give preference to boards with connectors, switches, LEDs and other small parts intact.

Generally avoid any PCB with charred spots or physically damaged components, unless you know you can easily repair the damage. Try to stick to PCBs that use readily available or standard connectors; this makes it easier to connect your new treasure to other systems or assemblies.

If you want to buy a large component such as a line printer, CPU board or disk drive, stick to a major name. This is important for two reasons. One, the major names usually make a better product (that's how they became major names). Two, you will have better luck finding documentation and service manuals for a major name assembly.

Do not expect to hit a home run with every purchase. Face it, some of the stuff out there simply doesn't work.



Not bad! A 63 watt PC power supply for $24.95. A typical overrun surplus item, this puppy could serve as a good bench supply or the base of a new clone system.

Do not expect to hit a home run with every purchase. Face it, some of the stuff out there simply doesn't work. But in the years spent scrounging through the surplus shops and buying mail-order, I found far more winners than losers. Be patient, look carefully, and have fun.

Survey Of Dealers

Jameco Electronics was originally James Electronics, one of the first advertisers in *BYTE* and *Kilobaud* (anyone out there remember *Kilobaud*?). They carry a large selection of new ICs and components, PC clone parts and some Apple equipment. Jameco often is the first large mail-order house to carry some of the newer chips. I've always gotten good service from Jameco and I love their catalogs. Recommended.

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

California Digital specializes in overrun surplus such as disk drives, power supplies and the like. They've been around for years and usually come up with first-rate bargains. The technical staff at CD can answer hacker questions about the hardware (just be sure to use the technical phone line for technical questions, instead of using the order phone line; that's the way the game is played). Check out their ads in *BYTE*, *CS* and other magazines.

California Digital
17700 Figueroa St.
Carson, CA 90248
(213) 217-0500 (technical & California)
(800) 421-5041 (toll-free order line)

JDR Microdevices started out years ago selling RAM chips, transistors and ICs when a RAM chip was a 1Kx1 static puppy. Since then, the component line has expanded dramatically and the service is still top notch. Their ads run in all the major magazines and you can call for a catalog. Good prices and prompt delivery.

JDR Microdevices
110 Knowles Dr.
Los Gatos, CA 95030
(800) 538-5000

GEnie is a worldwide bulletin board and database collection run by General Electric. Subscription fees are fairly standard for the industry but change periodically; call for the latest rates. The commands for maneuvering through the maze of boards, panel discussions and libraries are a little quirky, but there is gold here and it's worth the time needed to learn the system.

GE Information Services
401 N. Washington Street
Rockville, MD 20850
(800) 638-9636

ERAC CO.

8280 Clairemont Mesa Blvd., Suite 117
San Diego, California 92111
(619) 569-1864

AT

Motherboard 6 & 10 Meg
Zero Wait State
8 Expansion Slots
640K RAM On-Board
Math Co-processor Option
Phoenix Bios
200 Watt Power Supply
Hercules Compat. Video Bd.
Parallel Port
2 Serial Ports Active
Game Port
Clock/Calendar
Hard Disk & Floppy Controller
20M Hard Drive
1.2M 5¼" Floppy Drive
360K 5¼" Floppy Drive
5061 Keyboard
Case with Turbo & Reset,
Hard Drive Light and
Keyboard Disable Switch
Amber Graphics Monitor

$1581

EGA ADD $449
40M HD ADD $150
6 & 12 MHz ADD $73

BABY AT

Motherboard 6 & 10 Meg
Zero Wait State
8 Expansion Slots
80286 Processor
Math Co-Processor Option
640K RAM On-Board
Phoenix Bios
200 Watt Power Supply
Hercules Compat. Video Bd.
Parallel Port
2 Serial Ports Active
Game Port
Clock/Calendar
Hard Disk & Floppy Controller
20M Hard Drive
1.2M 5¼" Floppy Drive
360K 5¼" Floppy Drive
5061 Keyboard
Mini AT Case with Turbo &
Reset, Hard Drive Light and
Keyboard Disable Switch
Amber Graphics Monitor

$1531

EGA ADD $449
40M HD ADD $150

XT/TURBO

Motherboard
5 & 8 MHz Switchable
8088 — V20 Optional
Optional Co-processor
8 Expansion Slots
ERSO or Bison Bios
640K RAM
150 Watt Power Supply
Hercules Compat. Video Bd.
Parallel Port
2 Serial Ports Active
Game Port
Clock/Calendar
Hard Disk and
Floppy Controller
20M 5¼"Hard Drive
2 ea. 360K 5¼" Floppy Drive
AT Style Keyboard
Standard Slide Case
Amber Graphics Monitor

★ ★

$999

EGA ADD $429
40M HD ADD $150
5 & 10 MHz ADD $21

NiCds

AA Cells .6ah	$1.00
12V Pack AA Cells .6ah	6.50
Sub-C Cells 1.5ah	1.50
12V Pack Sub-C	10.00
Double D Cell 2.5V 4ah unused	8.00
C Cells	1.75
F Cells	2.50

GEL CELLS

12V 2.5ah	$8.50
D Cell 2.5ah	2.00

ROBOTICS

5V DC Gear Motor with Tach 1"x2"	7.50
Joystick, 4 switches, 1" knob	5.00
Z80 Controller with 8-Bit A/D	15.00
Brushless 12VDC 3" Fan	7.50
Capacitor, .47farad 10V 1"x1¾"	4.00
Solar Cells .5V .5A, .8"x1.6"	2.50
High Voltage Power Supply Input: 15-30V DC Output: 100V 400V 16KV	6.50

KAYPRO EQUIPMENT BARGAINS

9" Green Monitor	$50.00
Keyboard	75.00
PRO-8 Mod. to your board	149.00
Host Interface Board	15.00
Replacement Power Supply	50

IC'S

81-189 Video Pal	$15.00
81-194 RAM Pal	15.00
81-Series Char. Gen. ROMs	10.00
81-Series Monitor ROMs	10.00

★NEW★ CPM COMPUTERS

K4-83	$350
K2-84	400
K4-84	425
K10	500

ELGAR UNINTERRUPTIBLE POWER SUPPLY

400 Watt MODEL SPR401 $149

These supplies may have some 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

POWER SUPPLIES

0-8VDC 100A Metered Volt & Current Regulated	$249.00
5V/1A. -5V/.2A, 12V/1A, -12V/.2A, -24V/.05A	9.90

HOURS: Mon.-Fri. 9-6 — Sat. 10-4
MINIMUM ORDER — $15.00
TERMS: VISA, MasterCard, Certified Checks, Money Order, NO COD. Visa and MasterCard add3%. *Personal checks must clear BEFORE we ship.* Include shipping charges. California residents add 6½% Sales Tax. For more information please call.

CPU & SUPPORT CHIPS

MC68000-8 CPU	$8.00
Z80 CPU	.75
Z80A CPU	1.50
Z80 CTC	1.50
Z80A PIO	2.00
Z80A 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	.70
LM2877P 4W Stereo Amp Dual	2.50
MB81464-15	2.75
2716	3.00
2732	3.25
2764	3.50
27C128-1	9.00
74HC00	.38
74LS125	.30
74LS373	.50
74LS174	.30

SWITCHERS

5V/9.5A, 12V/3.8A, -12V/.8A	$39.00
5V/3A, 12V/2A, -12V/.4A	19.50
5V/6A, 12V/2A, -12V/1A	29.00
5V/6A, 24V/1¼A, 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

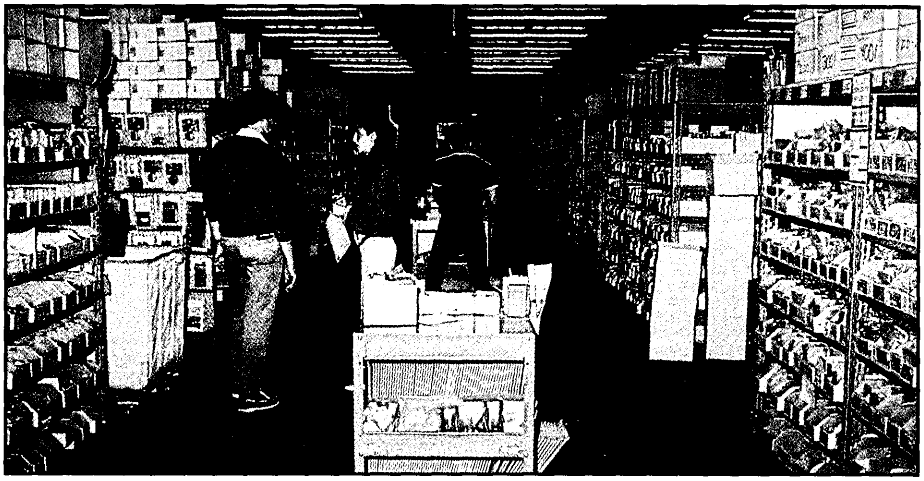
OSCILLOSCOPES	
TEK 7403N/7A18N/7B50A 60 MHz	$650
TEK 465 Dual Trace 100 MHz	1000
ANALYZERS	
TEK 491 10MHz - 40 GHz	$4500
Nicolet 500A 1 Hz 0-100 KHz	1800
Biomation 805 Waveform Rcrdr	259
Biomation 8100 2-Channel Waveform Recorder	795
HP1600A Logic Analyzer	600
HP1600A/1607A Logic Anlyzr	1000
FREQUENCY COUNTER	
Optronics 550 MHz	$100

DBASE BOOK OF BUSINESS APPLICATIONS *by Michael J. Clifford*
Reg. $19.95 *NOW ONLY* **$3.00**

Reader Service Number 93

Computer Parts Galore carries a large line of Apple and PC hardware, emphasizing bare expansion boards for these systems. If you want to build a RAM card or serial card yourself, you can save some bucks going through CPG. Their inventory varies quite a bit, so watch the ads in the major mags and give them a call from time to time. Prompt delivery and good support for technical questions.

Computer Parts Galore
56 Harvester Ave.
Batavia, NY 14020
(800) 431-9008



Filling Mail Orders at Jameco Electronics.

Computer Shopper, what more can I say? This mag is hacker heaven. Pages upon pages of hardware and software ads, interspersed with articles for hobbyists by hobbyists. Of all the things that make *CS* unique, perhaps the most important is their dedication to supporting the mail order customer. They screen their advertisers constantly for honesty and good service. If you have a problem with an advertiser, contact *CS* and they will work with both parties to get the problem resolved. I have ordered from *CS* often and never had a problem with misrepresentation or poor service. Pick up a copy at any large electronics store or bookstore, or write for a subscription.

Computer Shopper
5211 S. Washington Ave.
P.O. Box F
Titusville, FL 32781
(407) 269-3211

More years ago than I like to recall, Chuck Watson and Mel Cleverley started Tri-Tek in a small store next to a laundromat. The laundromat is gone, Tri-Tek has moved several times and Chuck has left for another business, but this remains one of the best retail electronics stores in Phoenix. Carrying a good line of new components and ICs, Tri-Tek also finds bargains in the overrun surplus market and passes the savings on. They used to do a mail-order business (Trivia Time: what was the name of the buxom cartoon character appearing in the Tri-Tek ads in *BYTE* magazine?), but have pretty much discontinued that sideline. If you're ever in Phoenix, stop by and say Hi. These are some of the best people in the surplus business.

Tri-Tek
9014 N. 23rd Ave. Suite 11
Phoenix, AZ 85021
(602) 371-0446

Most large cities with high-tech industry contain at least one (usually more) major surplus electronic outlet. In Phoenix, the place I do most of my surplus shopping is Apache Reclamation. PCBs, pumps, motors, wiring, connectors, ICs ... you name it, it's probably somewhere in Apache. Any dedicated bargain hunter passing through Phoenix will certainly want to include a stop at Apache Reclamation.

Apache Reclamation & Electronics
313 W. Apache St.
Phoenix, AZ 85004
(602) 254-0613

PS: If you do stop by Tri-Tek, you might notice the cartoon of Ample Annie behind the counter. I don't know how many power supplies Annie sold in her day, but she did have all the right parts.

♦ ♦ ♦

FAST
80386/80286 DTK MOTHERBOARDS

286-16

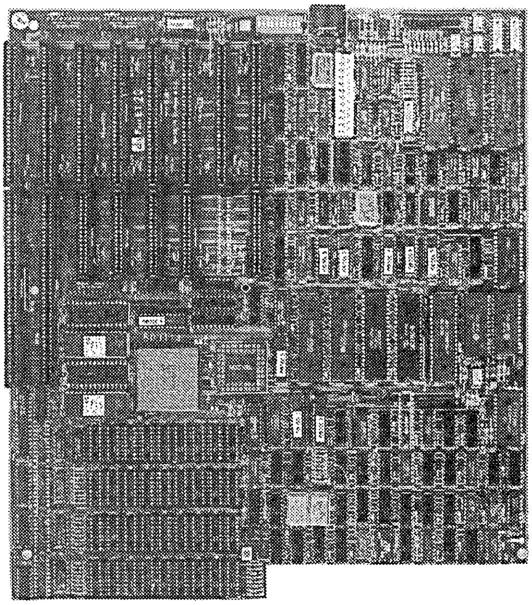
8/16 MHz
Ø or 1 wait state

8 Expansion Slots
6 - 16 Bit
2 - 8 Bit

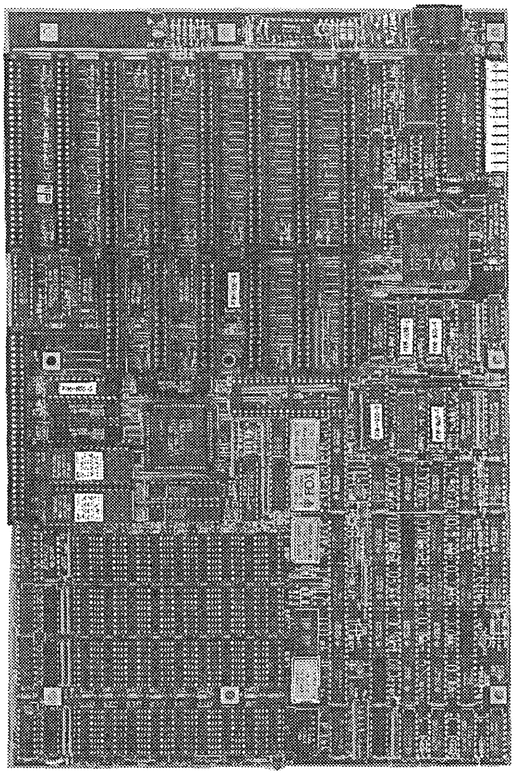
Local Memory Expansion Slot
Ø wait 60 NS DRAM
1 wait 100 NS DRAM

Norton SI = 18
Accepts 80287
8/16 MHz 80286 Motherboard

$475^00



20 MHz 80386



16 MHz 80286

386-20

8/20 MHz Switchable
1 wait state

Expands up to 17 MB
2 Serial - 1 Parallel
Hardware Reset/Turbo Switch
Accepts 80387-20
8 Expansion Slots
2 — 32 Bit Memory
5 — 16 Bit
1 — 8 Bit
Uses 80 NS DRAM
Norton SI = 24
8/20 MHz 80386 Motherboard

$950^00

**32 MB Memory Card for DTK 80386.
Holds up to 8 MB per card.
Accepts either 256K or 1 MB DRAM
$99^00**

Boards will fit in standard AT cabinet.

Prices subject to change without notice.

MicroSphere INC
COMPUTERS

MicroSphere, Inc.
855 N.W. Wall St.
Bend, Oregon 97701
(503) 388-1194

VISA MasterCard

Hours: Monday-Friday
9:00-5:30

Printer Graphics

Writing BGI-Like Printer Routines

When I called Borland about printer support for their BGI graphics interface, I got a "yes we're getting lots of requests for printer graphics but don't hold your breath."

The following day, Larry mentioned he was looking for something really esoteric to work on for this issue. Boy, did this project get esoteric all of a sudden.

Real programmers don't run applications. Therefore I try to avoid any use of a computer that might be construed as practical. But there comes a time when even I must face reality.

So, at the risk of writing something useful, let's continue on last issue's theme and add printer graphics functions to the Hercules graphics screen dump code.

Design Considerations

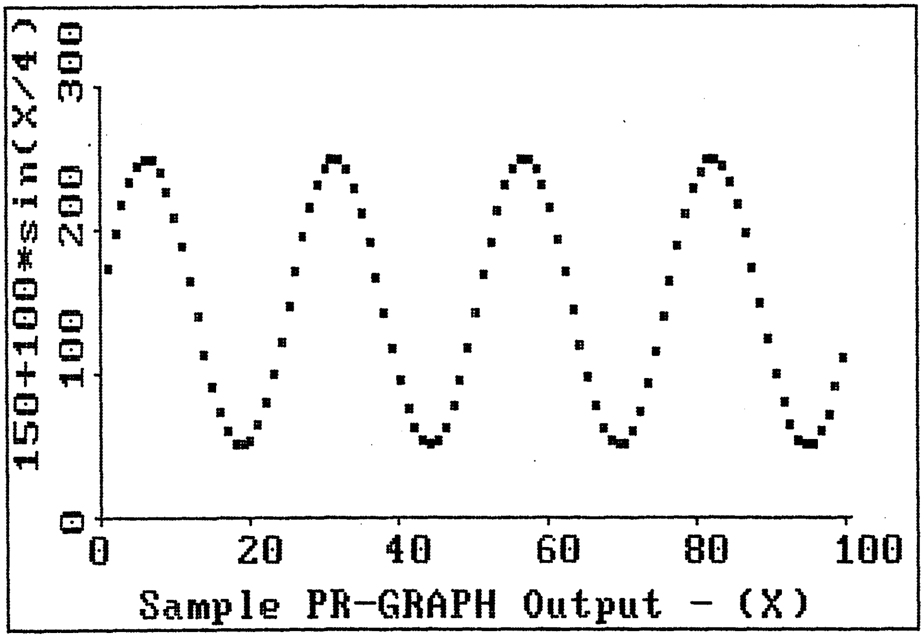
First off, we need an area of memory to represent the paper surface in the printer. Something to draw on, if you will. Call it printer memory.

I wanted to use the same interleaved memory organization as the Hercules card so that functions from HERCLIB.C would still work (see Hercules Primer in issue #39). And one simple block move of memory would suffice to load a video graphics screen into printer memory. Very nice when you consider the alternative: a byte by byte translation of video memory to printer memory.

Persuasive arguments, I felt. Dave thought otherwise. It got ugly for a while with flying floppies, ballistic boards, and uncalled-for comments. (Since Dave learned C, he's been casting aspersions.) You think I'm kidding? A few winters back we lost an 8" drive power supply to a direct hit from a snowball. Curious management style.

But Dave came up with the final word. With a linear space, organized as successive rows (just how the printer wants to see it), the only video hardware specific code lies in the translation routine, p_get_screen(). So it'll be easier to support new video cards in future versions.

Once implemented, my concerns about the speed of p_get_screen() proved to be groundless. It takes only a second or so to read a Hercules screen into printer memory. Take a look at Figure 1 for p_get_screen() and the rest of the code.



An array would be easier to deal with, but run-time sizing of printer memory has two advantages. First, it lets you use different printer modes easily. Higher density modes use more memory for a given size printout.

And, even if you only use a single printer mode, you won't always want to print the same size graphic. Dynamic sizing of printer memory with different values for p_max_line and p_max_col lets you print anything from a single graphics line to as many pages as you can stuff into 64K. Think of it. You could create something truly useful like a banner program.

Second, run-time allocation lets us easily recover printer memory while a globally declared array sticks around for the duration of the program's execution. While declaring printer memory as an array local to graph_test() would free the memory on return from graph_test(), the first method seems better to me. Either way, your program will appreciate the extra elbow room.

Clipping

I usually try to keep code for the magazine simple. That means no error checking on inputs, no checking for successful file I/O, etc. These absolutely necessary parts of a finished product tend to obscure the more interesting stuff.

But, in this case, one potential source for disaster must be considered. What happens if you get carried away and

By Larry Fogg
Micro C Staff

Figure 1 — Printer Graphics Functions

```
/* PR-GRAPH.C - library of printer graphics functions

   This code is up for grabs. In the immortal words of Laine Stump,
   "Permission to do whatever the hell you want with it."
   Compiled in compact model with Turbo C v1.5
   Requires the font file PR_CHARS.DAT generated by CHARGEN.C.

   Larry Fogg - Halloween, 1988
*/

#include <stdio.h>
#include <stdlib.h>
#include <dos.h>
#include <math.h>              /* only included for sin () example */
#include "dump.c"            /* printer functions from Micro C #44 */


#define boolean char
#define yes 1
#define no 0

#define N 'N'                     /* "directions" on printer paper */
#define E 'E'          /* N is the direction of normal text output */
#define W 'W'
#define S 'S'

#define herc_base 0xb000                        /* Herc parameters */
#define v_max_row 347
#define v_max_col 719

#define ch_size 11                   /* dot highth of a printer char */


typedef struct                 /* used for graph plotting routines */
{
  char orientation;                /* direction of X axis of graph */
  int x_min;                            /* actual axis coordinates */
  int x_max;
  int y_min;
  int y_max;
  char x_label [50];                                /* axis labels */
  char y_label [50];
  int xdata_min;                                    /* data bounds */
  int xdata_max;
  int ydata_min;
  int ydata_max;
  int x_step;                      /* increments for labeling axis */
  int y_step;
  int points [2][100];                     /* data in (x, y) pairs */
} plot;

char *p_buf;                          /* pointer to printer memory */
char p_chars [256][11];    /* holds 11 byte def for each character */

int p_max_line;                       /* printer memory boundaries */
int p_max_col;

unsigned p_buf_size;
char p_mode;                                       /* printer mode */
```

Listing Continued on page 42

write to an area outside of the current boundaries of printer memory? Perhaps you'll overwrite some other data from your program. Or you might even lobotomize the program. Whatever happens, it won't be pretty.

The initial logic of p_draw_point() takes care of this problem by "clipping," or simply not drawing any point that lies out of bounds. You might want to change the else clause during program development. Printing illegal points along the edge closest to where they disappear would at least show you the direction the errant points were going when last seen.

Complete User's Guide

To use these functions in your own code, just call p_init() and tell it how large an area you want to allocate. If it can't find that much memory, it'll complain and you can exit gracefully. Otherwise, you're on your way.

Most of the code is pretty self-explanatory. But... I wanted to be able to do text and graphics in any of four directions. Hence the compass headings. To orient yourself, a string printed "North" would look like normal text output of a printer.

Printer "lines" run North-South, and "columns" run East-West.

p_get_screen() grabs Herc graphics page 0 and stuffs it into printer memory in an Easterly direction. That means text printed along the bottom of the screen would run East.

You can position video graphics within the printer page using the two offset parameters. Note that a line_ofs of 1 is really 8 printer lines (one pass of the printer head). Also, be sure the screen doesn't go outside printer memory boundaries — the function doesn't check. Again, p_get_screen() is the only function you'll need to change to support other video cards.

Fonts

I've hard coded for the one font I created. ("Created" is a euphemism for "ripped off from the Hercules font.") Figure 2 shows a couple of examples from CHARGEN.C, the program which creates the font file. I finished all the alphanumerics before I ran screaming from the building. And I'll get as many symbols done as I can for the version of CHARGEN.C on the Micro C BBS and the Issue #45 disk.

The eleven bytes defining each character correspond to eleven "scan lines." Byte 0 defines the bottom line of the character and byte 10, the top. I guarantee you'll need a new prescription for your glasses after a few hours staring at the screen figuring out which bits should be "on" for each character.

On To The Practical

That's it — no more fun. Time to create something for the applications-oriented among us. I'll make it a bit more palatable by doing graphing functions that could be used for scientific work as well as business. (Sorry. Didn't mean to talk dirty. I'll avoid the "B" word from now on.)

graph_test() is a bit rough and rocky, but it'll give you an idea of the possibilities for using these printer routines.

A labeled and scaled plot requires a ton of information. Rather than pass enough parameters to choke a horse, I stuck 'em in a structure. graph_test() fills the structure as follows.

The orientation of your graph will be the direction of the x-axis. x_min and its ilk locate the axis in printer memory while xdata_min, etc., reflect the actual data bounds. x_step and y_step determine the increments marked by "ticks" on each axis.

Ideally, data sent to p_draw_plot() would be in the form of a pointer to a linked list. But the array of data simplifies matters and will work fine for many applications. In this example, we load the array with values from a sine function and send a pointer to the structure off to p_draw_plot(). Notice that points[0][0] holds the total number of data points.

To shorten the listing, I've shown only the East versions of the three functions called by p_draw_plot(). Lord knows why anyone would want to print upside down or inside out, but the code on our BBS and on the Issue #45 disk does support graphing in all four directions (for the more perverse among you).

The floats in p_draw_scale() and p_draw_data() force real division. Necessary, since we need to find what percent-

```
boolean p_init (char p_m, int max_line, int max_col)
{
  p_buf_size = (max_col + 1) * (((max_line+1)/8) + 1);
  p_buf = malloc (p_buf_size);           /* allocate printer memory */
  if (p_buf != NULL)           /* malloc () returns null if it fails */
  {
    p_max_line = max_line;                       /* set global vars */
    p_max_col = max_col;
    p_mode = p_m;
    return (yes);
  }
  else
    return (no);            /* not enough memory for printer memory */
}  /* p_init */


void p_load_chars ()           /* load the character font from file */
{                                 /* this file created by CHARGEN.C */
  FILE *f;

  f = fopen ("PR_CHARS.DAT", "rb");
  fread (p_chars, 0xb00, 1, f);
}  /* p_load_chars */


void p_clear_buf ()                        /* clears printer memory */
{
  unsigned i;

  for (i=0; i<p_buf_size; i++)
    *(p_buf+i) = 0;
}  /* p_clear_buf */


void p_draw_point (int col, int line, boolean fill)
{
  int i;      /* offset within memory for byte containing the point */
  char mask;                        /* locates point within the byte */

  if ((col>=0) && (line>=0) && (col<=p_max_col)
               && (line<=p_max_line))             /* coordinates OK */
  {
    mask = 1 << (7 - (line % 8));
    i = (line/8)*(p_max_col+1) + col;
      if (fill)                                   /* draw the point */
        *(p_buf+i)= *(p_buf+i) | mask;
      else                                       /* erase the point */
        *(p_buf+i) = *(p_buf+i) & ~mask;
  }
  else;                   /* do nothing - coordinates out of bounds */
}  /* p_draw_point */


void p_big_point (int col, int line, boolean fill)
{                             /* draws point and its nine neighbors */
  int i, j;

  for (i=col-1; i<=col+1; i++)
    for (j=line-1; j<=line+1; j++)
      p_draw_point (i, j, fill);
}  /* p_big_point */

void p_draw_line (int col1,int line1,int col2,int line2,boolean on)
{                        /* uses Bresenham algorithm to draw a line */
  int dX, dY;                                  /* vector components */
  int row, col,
      final,                           /* final row or column number */
      G,                        /* used to test for new row or column */
      inc1,        /* G increment when row or column doesn't change */
      inc2;               /* G increment when row or column changes */
  boolean pos_slope;

  dX = line2 - line1;                     /* find vector components */
  dY = col2 - col1;
  pos_slope = (dX > 0);                      /* is slope positive? */
  if (dY < 0)
    pos_slope = !pos_slope;
  if (abs (dX) > abs (dY))                     /* shallow line case */
  {
    if (dX > 0)            /* determine start point and last column */
    {
      col = line1;
      row = col1;
      final = line2;
    }
```

Listing Continued on page 43

```
    else
    {
      col = line2;
      row = col2;
      final = line1;
    }
    incl = 2 * abs (dY);     /* determine increments and initial G */
    G = incl - abs (dX);
    inc2 = 2 * (abs (dY) - abs (dX));
    if (pos_slope)
      while (col <= final) /* step through cols, check for new row */
      {
        p_draw_point (row, col, on);
        col++;
        if (G >= 0)                    /* it's time to change rows */
        {
          row++;   /* positive slope so increment through the rows */
          G += inc2;
        }
        else                               /* stay at the same row */
          G += incl;
      }

    else
      while (col <= final) /* step through cols, check for new row */
      {
        p_draw_point (row, col, on);
        col++;
        if (G > 0)                     /* it's time to change rows */
        {
          row--;  /* negative slope so decrement through the rows */
          G += inc2;
        }
        else                               /* stay at the same row */
          G += incl;
      }
  }  /* if |dX| > |dY| */

  else                                         /* steep line case */
  {
    if (dY > 0)              /* determine start point and last row */
    {
      col = line1;
      row = col1;
      final = col2;
    }
    else
    {
      col = line2;
      row = col2;
      final = col1;
    }
    incl = 2 * abs (dX);     /* determine increments and initial G */
    G = incl - abs (dY);
    inc2 = 2 * (abs (dX) - abs (dY));
    if (pos_slope)
      while (row <= final) /* step through rows, check for new col */
      {
        p_draw_point (row, col, on);
        row++;
        if (G >= 0)                 /* it's time to change columns */
        {
          col++;     /* pos slope so increment through the columns */
          G += inc2;
        }
        else                            /* stay at the same column */
          G += incl;
      }

    else
      while (row <= final) /* step through rows, check for new col */
      {
        p_draw_point (row, col, on);
        row++;
        if (G > 0)                  /* it's time to change columns */
        {
          col--;     /* neg slope so decrement through the columns */
          G += inc2;
        }
        else                            /* stay at the same column */
          G += incl;
      }
  }
}  /* p_draw_line */
```

age of the total span is used by a point in order to scale it along the axis.

In p_draw_scale() dx and dy represent the number of printer dots per increment on each axis. "Exact" values for these variables would be real numbers. But the printer only understands integral values for addressing lines and columns. So we need to round them off.

For non-negative numbers, adding 0.5 and truncating the result performs rounding. So when calculating dx, we add 0.5. Now the compiler casts this real result to an int during the assignment to dx. Voilà. The casting truncates the real and we have a well-rounded result. Use of the same technique in p_draw_data() lets us plot the data as close to its real location as possible.

This points out the main failure of these plotting routines — they can't deal with real number data and axis labels.

BUGS!!!

You know the feeling. At the end of a programming project you look back and say to yourself, "What a baboon. If only I'd ..." Chasing bugs can take up more of your time than design, coding, *and* documenting.

Well, I certainly had my share of silly errors this time. Since I'm a Pascal programmer at heart, a lot of my problems have to do with sloppiness in parameter passing. Of course, prototyping would help a bunch, but I haven't gotten into the habit.

So I tried to send malloc() a long int when all it really wanted was an unsigned int. Now C always tries to cooperate with the programmer. No matter how foolish your code is, C really wants to compile it. That's not to say that the compiled code will execute properly, but it's usually good for a laugh. I got a lot of laughs from malloc().

My perception of my programming abilities tends to vary from godlike to gawd *&%$#@. With my technical self-esteem at a low ebb this issue, I assumed that the most pernicious of the latest bug batch stemmed from my code. A very entertaining problem, it caused the printer head to do the jitterbug from time to time. Interesting behavior, but hardly useful.

I pared down the code until it did nothing but send graphics lines to the printer. And found that firing the top pin repeatedly would set the print head dancing. My tired old Gemini just couldn't handle the physical stress. Figures. When I'm sure my code is bad, it's good. And when it just *has* to be good, it's snot.

Universality

Last issue I marveled over the apparent compatibility between printers in the MS-DOS world. I've since had my bubble burst. And to make matters worse, from within the same product line.

I've always liked Star Micronics for their low-cost workhorse printers, but they're the ones who tipped me off to the dangers of printer programming. Their LV 1210 and 1215 require ESC '2' to enable 8/72" line feeds. Unfortunately, ESC '2' sets 1/6" line feeds on the Geminis. Oh well. So much for the "universal" printer interface.

If Figure 1 makes your printer do unnatural acts, check the escape sequences in DUMP.C against your printer's documentation. One of them will no doubt be the culprit.

We have a pretty fair printer museum here at the world headquarters of *Micro Cornucopia*. But even so, it covers only a tiny percentage of the printer kingdom. If you have a particularly peculiar printer, why not send us the codes necessary to make it jump through the hoops we've talked about here. I'll compile the responses for an entry in Tech Tips.

And Furthermore ...

Some things come to mind as this article slips past three weeks overdue:

- More fonts and more sizes would be nice.
- As would circle drawing capability for pie charts.
- We need more than 64K of printer memory for use with the higher density printer modes or with very large printouts (banners). I think you can do this with a huge pointer to printer memory and use of calloc() to allocate storage of lines rather than malloc() to store bytes.

But I leave these as exercises for the reader. (The ultimate academic cop-out.)

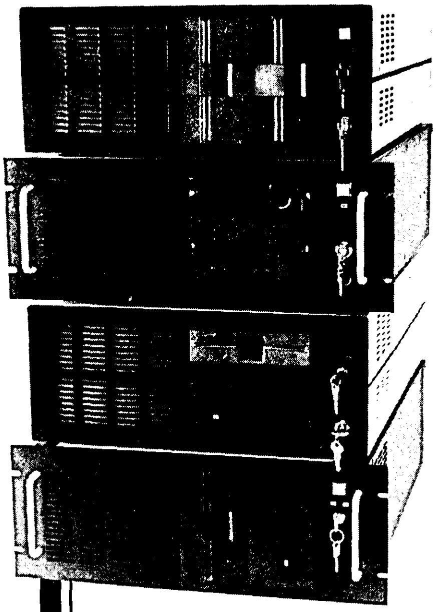
♦ ♦ ♦

```
void p_put_c (unsigned char ch, int col, int line, char direction)
{                                              /* print a character */
  int r, c;

  switch (direction)
  {
    case E: for (r=0; r<11; r++)
              for (c=0; c<8; c++)
                p_draw_point (col+r, line+c,
                              p_chars [ch][r] & (1 << (7-c)));
            break;
    case W: for (r=0; r<11; r++)
              for (c=0; c<8; c++)
                p_draw_point (col-r, line-c,
                              p_chars [ch][r] & (1 << (7-c)));
            break;
    case N: for (r=0; r<11; r++)
              for (c=0; c<8; c++)
                p_draw_point (col+c, line-r,
                              p_chars [ch][r] & (1 << (7-c)));
            break;
    case S: for (r=0; r<11; r++)
              for (c=0; c<8; c++)
                p_draw_point (col-c, line+r,
                              p_chars [ch][r] & (1 << (7-c)));
            break;
  }  /* switch */
}  /* p_put_c */


void p_put_s (unsigned char str_out [80], int col,
              int line, char direction)
{                                                  /* print a string */
  int i;

  i = 0;
  switch (direction)
  {
    case E: while (str_out [i] != '\0')
            {
              p_put_c (str_out [i], col, line + 9*i, E);
              i++;
            }
            break;
    case W: while (str_out [i] != '\0')
            {
              p_put_c (str_out [i], col, line - 9*i, W);
              i++;
            }
            break;
    case N: while (str_out [i] != '\0')
            {
              p_put_c (str_out [i], col + 9*i, line, N);
              i++;
            }
            break;
    case S: while (str_out [i] != '\0')
            {
              p_put_c (str_out [i], col - 9*i, line, S);
              i++;
            }
            break;
  }  /* switch */
}  /* p_put_s */


void p_print_buf ()         /* print out contents of printer memory */
{
  int col, line, total_lines;

  total_lines = p_max_line/8 + 1; /* # passes for the printer head */
  prn_setup ();

  for (line=0; line<total_lines; line++)
  {
    setup_gr_line (p_mode, p_max_col+1);
    for (col=0; col<=p_max_col; col++)         /* send graphics line */
      prn_out (*(p_buf + (unsigned)(line*(p_max_col+1) + col)));
    prn_out (10);                                        /* print it */
  }
}  /* p_print_buf */
```

Continued on page 45

```
void p_draw_border (int col1, int line1, int col2, int line2)
{
  p_draw_line (col1, line1, col2, line1, yes);
  p_draw_line (col2, line1, col2, line2, yes);
  p_draw_line (col2, line2, col1, line2, yes);
  p_draw_line (col1, line2, col1, line1, yes);
}  /* p_draw_border */

void p_get_screen (int col_ofs, int line_ofs)
{                        /* load Herc graphics page 0 into printer memory */
  unsigned buf_index;          /* offset from base of printer memory */
  int byte_ofs,                  /* offset from base of video memory */
      vid_col, vid_row, blanks;

  blanks = p_max_col - v_max_row; /* skip these between video cols */
  buf_index = line_ofs*(p_max_col + 1) + col_ofs;    /* start here */
  for (vid_col=0; vid_col<v_max_col; vid_col+=8, buf_index+=blanks)
    for (vid_row=v_max_row; vid_row>=0; vid_row--, buf_index++)
    {
      byte_ofs = 0x2000*(vid_row % 4) + 90*(vid_row/4) + (vid_col/8);
      *(p_buf + buf_index) = peekb (herc_base, byte_ofs);
    }
}  /* p_get_screen */


void p_draw_axis (plot *plt)    /* draw axis and print text labels */
{
  switch (plt->orientation)    /* one case only for brevity's sake */
  {
    case E: p_draw_line (plt->y_min, plt->x_min, plt->y_min,
                         plt->x_max, yes);
            p_draw_line (plt->y_min, plt->x_min, plt->y_max,
                         plt->x_min, yes);
            p_put_s (plt->x_label, plt->y_min-3*ch_size,
                         plt->x_min+15, E);
            p_put_s (plt-> y_label, plt->y_min+15,
                         plt->x_min-2*ch_size, N);
            break;
  }  /* switch */
}  /* p_draw_axis */


void p_draw_scale (plot *plt)          /* do labeled scale on axis */
{
  int x, y,                             /* current drawing location */
      dx, dy,                                    /* axis increments */
      x_label, y_label,                 /* incremental axis labels */
      x_span, y_span;                              /* plotting range */
  float xdata_span, ydata_span;/*data range (float forces real div)*/
  char temp [6];               /* x_label and y_label in string form */

  x_span = plt->x_max - plt->x_min;          /* find plotting range */
  y_span = plt->y_max - plt->y_min;
  xdata_span = plt->xdata_max - plt->xdata_min; /* find data range */
  ydata_span = plt->ydata_max - plt->ydata_min;
  dx = (plt->x_step/xdata_span) * x_span + 0.5;  /* find axis incs */
  dy = (plt->y_step/ydata_span) * y_span + 0.5;
  switch (plt->orientation)                       /* one case only */
  {
    case E: x = plt->x_min;                       /* initial values */
            x_label = plt->xdata_min;
            while (x <= plt->x_max)
            {
              p_draw_point (plt->y_min - 1, x, yes);  /* draw tick */
              p_draw_point (plt->y_min - 2, x, yes);
              itoa (x_label, temp, 10); /*convert tick label to str*/
              p_put_s (temp, plt->y_min-16, x-4, E); /* label tick */
              x_label += plt->x_step;      /* set up for next tick */
              x += dx;
            }
            y = plt->y_min;                        /* same for y-axis */
            y_label = plt->ydata_min;
            while (y <= plt->y_max)
            {
              p_draw_point (y, plt->x_min - 1, yes);
              p_draw_point (y, plt->x_min - 2, yes);
              itoa (y_label, temp, 10);
              p_put_s (temp, y - 4, plt->x_min - 5, N);
              y_label += plt->y_step;
              y += dy;
            }
            break;
  }  /* switch */
}  /* p_draw_scale */
```

Continued on page 46



```
void p_draw_data (plot *plt)                        /* plot the data */
{
  int i,
      x_span, y_span;                              /* plotting range */
  float xdata_span, ydata_span;/*data range (float forces real div)*/

  x_span = plt->x_max - plt->x_min;         /* find plotting range */
  y_span = plt->y_max - plt->y_min;
  xdata_span = plt->xdata_max - plt->xdata_min; /* find data range */
  ydata_span = plt->ydata_max - plt->ydata_min;
  switch (plt->orientation)                         /* one case only */
  {
    case E: for (i=1; i<=plt->points [0][0]; i++)/*0,0 has # of pts*/
              p_big_point(plt->y_min +
                          (plt->points[1][i]/ydata_span)*y_span + 0.5,
                          plt->x_min +
                          (plt->points[0][i]/xdata_span)*x_span + 0.5,
                          yes);
            break;
  }  /* switch */
}  /* p_draw_data */


void p_draw_plot (plot *plt)
{
  p_draw_axis (plt);
  p_draw_scale (plt);
  p_draw_data (plt);
}  /* p_draw_plt */


void graph_test ()                             /* example of graph use */
{
  int i;
  plot *p;

  p = (plot *) malloc (sizeof (plot)); /* get memory for structure */
  p->orientation = E;                      /* load structure values */
  p->x_min = 50;  p->x_max = p_max_line - 50;
  p->y_min = 50;  p->y_max = p_max_col - 50;
  strcpy (p->x_label, "                         X");
  strcpy (p->y_label, "      150 + 100 * sin(6.28*X/25)");
  p->xdata_min = 0;  p->xdata_max = 100;
  p->ydata_min = 0;  p->ydata_max = 300;
  p->x_step = 10;
  p->y_step = 50;
  p->points [0][0] = 99;      /* element 0,0 holds # of data points */
  for (i=1; i<=99; i++)                           /* load data array */
  {
    p->points [0][i] = i;                                /* x value */
    p->points [1][i] = 150 + 100*sin(6.28*i/25);         /* y value */
  }
  p_draw_plot (p);                                   /* do the graph */
  p_draw_border (0, 0, p_max_col, p_max_line);
  p_print_buf ();                                        /* print it */
}  /* graph_test */


main ()
{
  if (p_init (75, 750, 470))   /* mode 75, 750 lines x 470 columns */
  {
    p_clear_buf ();
    p_load_chars ();
    graph_test ();
    free (p_buf);                          /* release printer memory */
  }
  else
    printf ("\n\n  Not enough memory ...\n\n");

}  /* main */

***
```

Figure 2 — CHARGEN.C

```
#include <stdio.h>

char p_chars [256][11];

void load_array ()
{
  /* A */
  p_chars[65][0] = 0;
  p_chars[65][1] = 0;
  p_chars[65][2] = 0xc6;
  p_chars[65][3] = 0xc6;
  p_chars[65][4] = 0xc6;
  p_chars[65][5] = 0xfe;
  p_chars[65][6] = 0xc6;
  p_chars[65][7] = 0xc6;
  p_chars[65][8] = 0x6c;
  p_chars[65][9] = 0x38;
  p_chars[65][10] = 0x10;
  /* y */
  p_chars[121][0] = 0x78;
  p_chars[121][1] = 0x0c;
  p_chars[121][2] = 0x06;
  p_chars[121][3] = 0x7e;
  p_chars[121][4] = 0xc6;
  p_chars[121][5] = 0xc6;
  p_chars[121][6] = 0xc6;
  p_chars[121][7] = 0xc6;
  p_chars[121][8] = 0;
  p_chars[121][9] = 0;
  p_chars[121][10] = 0;
  /* you get the idea */
}  /* load_array */



void make_file ()
{
  FILE *f;

  f=fopen("PR_CHARS.DAT","wb");
  fwrite(p_chars, 0xb00, 1, f);
}  /* p_load_chars */



main ()
{
  puts("\nPR_CHARS.DAT ...\n");
  load_array ();
  make_file ();
  puts ("All done!\n");
}  /* main */

***
```

By Thomas Gettys
700 Snowberry Ct.
Layfayette, Colorado 80026
(303) 665-5981

A New Way Of Life

Animating Conway

It's always fun to play Life, especially fun when you realize that early Life forms were created prior to computers. Imagine the excitement when the first glider emerged from a Life sciences study.

It's been twenty years since John Conway invented the popular cellular-automaton game of Life, but computer enthusiasts continue to be fascinated by its wild and unpredictable patterns.

The playing field is made up of a grid of cells. Each cell may be either alive or dead. Conway established a simple set of rules to determine which cells will be alive in the following generation.

But folks haven't left things alone investigating what happens when you modify the neighborhood definition, number of states, or the rules. Plus there are other dimensions to the work, including one dimensional Life[1,2] and, more recently, three dimensional Life[3].

The secret of Life's popularity rests on two special features. First, it's computation-intensive. Investigation of all but the most trivial patterns would be impossible without the aid of a computer.

Second, it's animated. Color certainly adds interest, but there is nothing like motion to really grab and hold your attention!

One Problem

Unfortunately, Life's usually pretty slow. *(Editor's note: we're still talking about the game here.)* So it's not unusual to count progress in generations per minute while many of the most interesting patterns require hundreds, sometimes even thousands, of generations.

Which brings us to the point of this article: making Life a little more interesting (and a lot faster) by using a more efficient algorithm.

The Basic Method

Let's look at the original, dumb algorithm. For each cell C do:

```
if C is alive
  then if C does not have
  2 or 3 living neighbors
    then C dies;

if C is not alive
  then if C has 3 living neighbors
    then C becomes alive;
```

It's easy to see that if the field contains mostly dead cells, you're going to waste a lot of effort on inconsequential cells. After all, the only possible changes to the current population come from live cells and their nonliving neighbors.

If we had a list of living cells, we could limit our search to those when deciding which will die. Likewise, if we had a list of neighbors to living cells, we could determine which, if any, would become alive in the next generation.

Let's Digress For A Moment

If you consult your favorite text on data structures, you will probably find a short discussion on the grave disadvantages of simple lists; encouraging you to use such finery as linked lists, queues, or binary trees.

However, there are problems which don't require a particular order. For instance, all you need here are insertions and deletions, so a simple list is quite sufficient. To insert a new item, simply append it to the list. Delete it by making it the last item in the list and decrementing the list size.

Besides the lists of living cells and nonliving neighbors of living cells, it'll be useful to maintain an array containing the number of living neighbors for each cell (add one if the cell is alive — more on this later).

Also, for each generation we will build two additional lists containing the changes to the current population: one for the births and one for the deaths. Each of these data structures is declared global, as each will be used by more than one procedure. (See Figure 1 for all of the code.)

Data Structures

I've set up five lists (arrays). Four of the lists — births, deaths, living cells, and nonliving neighbors — contain X,Y coordinate pairs (cell coordinates). These are variable length lists. (If there were two births, for instance, the birth list would contain two coordinate pairs.)

The lists of living cells and nonliving neighbors continue from generation to generation. The lists of births and deaths are created from scratch for each generation.

The fifth list is a fixed-length array of neighbor counts where there is one element in the list for each of the cells. This is a two-dimensional array with its dimensions corresponding with the cell's X and Y coordinates. The only thing stored in this array is the count of neighbors for each cell.

Four Procedures

I've divided the main portion of the program into four procedures.

The first scans the list of living cells to determine which will die. These are appended to the list of deaths and deleted from the list of the living.

The second scans the list of nonliving neighbors of living cells to determine where there'll be births. These cells are appended to the list of births and deleted from the list of nonliving neighbors. This procedure also removes from the nonliving neighbor list any cells which no longer have living neighbors.

The third procedure processes the list of births. Each cell is appended to the list of living cells and the array of neighbor counts is updated by incrementing each of the eight neighboring cells. If a neighbor count was zero, the cell must be appended to the list of nonliving neighbors, as it now has a living neighbor.

The fourth procedure uses the list of deaths to update the array of neighbor counts. If the cell that died has living neighbors (its own neighbor count is nonzero), it must be appended to the list of nonliving neighbors.

Figure 1 — Life.C

```
#include <graphics.h>
#define BYTE unsigned char
#define WORD unsigned int
#define FALSE 0
#define TRUE 1

/* rule parameters governing births and deaths */
#define EL 2  /*envrnmnt,lower:death if n_cnt<EL*/
#define EU 3  /*envrnmnt,upper:      or n_cnt>EU*/
#define FL 3  /*fertlty,lower:birth if n_cnt>=FL*/
#define FU 3  /*fertlty,upper:     and n_cnt<=FU*/
/* configuration constants (array sizes) */
#define MAX_ALIVE 600     /* max # living cells */
#define MAX_GHOST 1700/*max # non-live neighbors*/
#define MAX_BIRTH 300/* max # births/generation */
#define MAX_DEATH 300/* max # deaths/generation */
#define MAX_X 200              /* width of grid  */
#define MAX_Y 200              /* height of grid */
/* global variable declarations */
BYTE n_cnt[MAX_X][MAX_Y]; /* neighbor count ... */
                           /* ... for all cells */
BYTE alive[2][MAX_ALIVE];  /* list/living cells */
BYTE ghost[2][MAX_GHOST];  /*list/neighbor cells*/
BYTE birth[2][MAX_BIRTH];  /* list/new births   */
BYTE death[2][MAX_DEATH];  /* list/new deaths   */
WORD alive_cnt; /* size of list of living cells */
WORD ghost_cnt;    /* size of list of neighbors */
WORD birth_cnt;       /* size of list of births */
WORD death_cnt;       /* size of list of deaths */

void locate_deaths(void)
{
   BYTE c, x, y;
   WORD index;

   death_cnt = index = 0;
   while (index < alive_cnt)
   {
      x = alive[0][index];  y = alive[1][index];
      c = n_cnt[x][y] - 1;
      if (c >= EL && c <= EU) index++;
      else
      {
         death[0][death_cnt] = x;
         death[1][death_cnt] = y;
         death_cnt++;
         alive_cnt--;
         alive[0][index] = alive[0][alive_cnt];
         alive[1][index] = alive[1][alive_cnt];
      }
   }
}

void locate_births(void)
{
   BYTE c, x, y;
   WORD index;
   birth_cnt = index = 0;
   while (index < ghost_cnt)
   {
      x = ghost[0][index];  y = ghost[1][index];
      c = n_cnt[x][y];
      if (c >= FL && c <= FU)
      {
         if (x > 0 && y > 0 && x < MAX_X-1 &&
             y < MAX_Y-1)
         {
            birth[0][birth_cnt] = x;
            birth[1][birth_cnt] = y;
            birth_cnt++;
         }
         ghost_cnt--;
         ghost[0][index] = ghost[0][ghost_cnt];
         ghost[1][index] = ghost[1][ghost_cnt];
      }
      else if (c == 0)
      {
         ghost_cnt--;
         ghost[0][index] = ghost[0][ghost_cnt];
         ghost[1][index] = ghost[1][ghost_cnt];
      }
      else index++;
   }
}
```

```
void process_births(void)
{
   BYTE x, y;
   WORD index;

   for (index = 0; index < birth_cnt; index++)
   {
      x = alive[0][alive_cnt] = birth[0][index];
      y = alive[1][alive_cnt] = birth[1][index];
      alive_cnt++;
      n_cnt[x][y]++;
      x++;
      if (n_cnt[x][y]++ == 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
      y++;
      if (n_cnt[x][y]++ == 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
      x--;
      if (n_cnt[x][y]++ == 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
      x--;
      if (n_cnt[x][y]++ == 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
      y--;
      if (n_cnt[x][y]++ == 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
      y--;
      if (n_cnt[x][y]++ == 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
      x++;
      if (n_cnt[x][y]++ == 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
      x++;
      if (n_cnt[x][y]++ == 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
   }
}
void process_deaths(void)
{
   BYTE i, j, x, y;
   WORD index;

   for (index = 0; index < death_cnt; index++)
   {
      x = death[0][index];
      y = death[1][index];
      for (i = x - 1; i < x + 2; i++)
      for (j = y - 1; j < y + 2; j++)
         n_cnt[i][j]--;
   }
```

Continued on page 49

```
   for (index = 0; index < death_cnt; index++)
   {
      x = death[0][index];
      y = death[1][index];
      if (n_cnt[x][y] != 0)
      {
         ghost[0][ghost_cnt] = x;
         ghost[1][ghost_cnt] = y;
         ghost_cnt++;
      }
   }
}

void screen_update(void)
{
   BYTE x, y;
   WORD index;

   for (index = 0; index < birth_cnt; index++)
   {
      x = birth[0][index];  y = birth[1][index];
      putpixel(x, y, 1);
   }
   for (index = 0; index < death_cnt; index++)
   {
      x = death[0][index];  y = death[1][index];
      putpixel(x, y, 2);
   }
}

void init_grid(void)
{
   BYTE x, y;
   WORD index;

   for (x = 0; x < MAX_X; x++)
   for (y = 0; y < MAX_Y; y++)
      n_cnt[x][y] = 0;
   birth[0][0] = 100;   birth[1][0] = 100;
   birth[0][1] = 101;   birth[1][1] = 100;
   birth[0][2] = 102;   birth[1][2] = 100;
   birth[0][3] = 102;   birth[1][3] = 99;
   birth[0][4] = 101;   birth[1][4] = 101;
   birth_cnt = 5;   alive_cnt = 0;
   ghost_cnt = 0;
   screen_update();
   for (index = 0; index < birth_cnt; index++)
   {
      x = birth[0][index];  y = birth[1][index];
      n_cnt[x][y]++;
   }
   process_births();
   for (index = 0; index < birth_cnt; index++)
   {
      x = birth[0][index];  y = birth[1][index];
      n_cnt[x][y]--;
   }
}

void main(void)
{
   int step = FALSE;  int done = FALSE;
   int generation = 0;
   int gd = CGA;  int gm = CGAC0;
   initgraph(&gd, &gm, "");
   printf("Generation: 0    Run  Pause  Step  Quit");
   init_grid();
   gotoxy(1, 1);  getch();
   while (!done)
   {
      gotoxy(13, 1);  printf("%d", ++generation);
      locate_deaths();  locate_births();
      process_births();  process_deaths();
      screen_update();
      if (kbhit() || step)
      {
         switch(toupper(getch()))
         {
            case 'Q': done = TRUE;
                      break;
            case 'P': getch();
                      break;
            case 'S': step = 1;
                      break;
            case 'R': step = 0;
                      break;
         }
      }
   }
   closegraph();
}

***
```

A Subtle Problem

The array of neighbor counts includes an extra count if the corresponding cell is alive. I did this to deal with an especially subtle situation. To see the problem, suppose I don't do an extra count for living cells. Then consider the following situation:

```
  12345
1 .....
2 .*.*.      living cells = '*'
3 .*...      nonliving cells = '.'
4 .....
```

None of the living cells has 2 or 3 living neighbors, so all three will die. Since the cell at (3,2) has 3 living, it will be born. Thus the list of deaths has three entries and the list of births has one.

Now, in the third procedure when the neighbor counts are updated, the cell at (4,2), having a neighbor count of 0, would be appended to the list of nonliving neighbors to living cells.

When the death at (4,2) is processed in procedure four, it would again be placed in the list of nonliving neighbors because it now has a neighbor count of 1, and so must be a nonliving neighbor of a living cell!

By adding an extra count to each living cell, the first case is avoided and so the cell at (4,2) will only be added to the list of nonliving neighbors in the fourth procedure.

Mop-Up

Initialization zeros the array of neighbor counts and puts 1s into the neighbor counts of cells where there'll be births. Then it installs the pattern of living cells into the list of births. Finally it calls the procedure that processes the birth list.

After calling the procedure, the 1s are subtracted back out of the neighbor counts. This prevents the procedure from appending any of the cells in the initial pattern to the list of nonliving neighbors.

The initialization routine creates a single pattern known as the "R pentomino" into the grid. The pattern stabilizes after 1,103 generations, yielding 6 gliders, 4 blinkers, and 14 static forms[5].

Finally

You can update this algorithm for three (or more!) dimensions by simply adjusting the number of columns in the lists and changing the routines that update the array of neighbor counts. After that, the hard part is the graphics!

The parameters EL, EU, FL and FU (environment and fertility low and high) were borrowed from the article on three-dimensional Life[3]. Use them if you'd like to experiment with the change-of-state rules.

Bibliography

[1] *Scientific American* 9/84 (1 dimensional Life)

[2] *BYTE* 12/86 (1 dimensional life)

[3] *Scientific American* 2/87 (3 dimensional Life)

[4] *Scientific American* 4/84 (general cellular automata)

[5] *Wheels, Life, and Other Mathematical Amusements* Martin Gardner 1984

♦ ♦ ♦

C'ING CLEARLY

The Turbo Debugger

(You saw it here — finally)

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

Scott finally has the Turbo Debugger. That's right, Borland has birthed the real, long awaited, often debated, currently integrated, our appetite is sated — Turbo Debugger. Interestingly enough, it's twins.

It is now late September (as I write this). The aspen have turned glorious gold, the air has a nip, and I'm having a harder time getting out of bed in the morning. At least my pants and shoes (soaked at SOG) have dried out.

You'll notice the contents and organization of this column have changed. C programmers are a diverse lot so I'm trying to reach all of you. You'll see I've broken the column into four sections: *Toolbox, C Explorations, News & Reviews,* and *Resources.*

Toolbox will cover tools and how to use them. There are many tools: compilers, linkers, debuggers, profilers, etc. In this section I'll include in-depth reviews of noteworthy products, or a tutorial on how to make the most of a given tool. This time I'll talk about Borland's new Turbo Debugger, and discuss how (and when) to use a debugger.

Unlike some other C programming columns, I don't want to rehash last year's algorithms. Programmers have hundreds of resources which teach them how to write linked-lists, binary trees, and sort functions. Instead, in the C *Explorations* section, I'll be covering the poorly documented areas of C — those facilities, functions, and theories no one talks about.

Due to the length of the Turbo Debugger review, *Explorations* will debut in the May/June issue (#47). Future discussions will be about the ANSI C time and date functions, interrupt handlers and optimization techniques.

News and Reviews will contain new product mentions, industry news, and other nontechnical bits. I've had some very positive feedback from folks who want to know what's going on with vendors, products, and standards. Here's where you'll find it. This issue contains quick reviews of Turbo C 2.0, Turbo Assembler 1.0, Quick C 2.0, and the latest news on the ANSI standard.

Finally, there's the *Resources* part of the column. Here's where I'll talk about books and other information sources.

Hopefully, these changes will make this an even better column. Now, let's get down to business...

Toolbox

You've seen the hype. For years, Microsoft's CodeView has reigned supreme in the world of symbolic debuggers. Every review of a Microsoft language has extolled the virtues of CodeView — especially how no other products compare. At the same time, Borland was being flayed for its lack of same.

Enter Turbo Debugger, the so-called "CodeView killer" from Borland. Fast, sleek, and powerful, Turbo Debugger (hereafter referred to as TD) gives CodeView (CV) a run for its money. TD refines the leap in debugging technology first made by CodeView.

Debuggers may rank as the most misunderstood and misused programming tools. Many programmers have never used a debugger, and have no idea how one works or what it does. Before I provide an in-depth look at the Turbo Debugger, I'll provide an overview of the state-of-the-art in debugging.

About Debuggers

A symbolic (source-level) debugger shows you the source code of your program as it executes. You'll see which lines are executed in what order. You can even watch the machine-code instructions being executed.

You can set "breakpoints," which tell the debugger to stop your program when it reaches a certain place. You can examine and change variables while the program is running. Any variable marked as a "tracepoint" will have its current value displayed as the program executes, letting you see how the value changes and when.

"Watchpoints" are conditional breakpoints based on the values of variables; when the set condition becomes true, the debugger stops and shows you where the event occurred. All these features (plus many others) provide a detailed view of what's going on inside of your program.

Debuggers, however, aren't smart — they can only show you what's happening. It's up to you to interpret the information.

Turbo Debugger

Turbo C is a year-and-a-half old, but until the release of version 2.0 it hasn't had its own debugger. Instead, Borland has suggested using third-party products.

Now, however, Borland is to be commended for producing an exceptional product.

Borland's debugger is actually two: the limited debugger built into Turbo C, and the more-powerful standalone Turbo Debugger sold with Turbo Assembler. This is part of the reason you will see two Turbo C packages listed in advertisements.

The standard Turbo C 2.0 package is an integrated compiler (now including a debugger), along with command-line utilities (a linker, librarian, etc.) The Turbo C 2.0 Professional package has everything the standard package does, plus the standalone Turbo Debugger and Turbo Assembler. Turbo C Professional costs $250. The debugger and assembler can be purchased as a set for $150.

Why two debuggers? Borland built the most useful features of Turbo Debugger into the integrated package. There just isn't memory space to include all of the debugging features in the package.

I'll talk about both debuggers, so you know what you can (and can't) do in each of them. In order to keep my fingers from wearing out, I'll refer to the integrated debugger as ID, and the standalone Turbo Debugger as TD.

Using The Debuggers

The integrated environment is automatically set up to compile your program with debugging information built-in; when using the command-line compiler, you need to use the -v switch.

A program compiled for debugging contains references to the source. While it may seem like the debugger is executing your source code, it's actually cross-referencing current execution with your original file.

Both debuggers show you your source while the program executes. A pointer on the left side of the screen flags the line as it's executed. You can choose screen-swapping so your program's output (on one) doesn't interfere with the debugger's display (on the other). A single keystroke selects either screen. This allows the debuggers to work with programs that do graphics.

There are several ways in which you can execute your program in the debuggers. Both debuggers allow you to "run" a program until it encounters a breakpoint or the program terminates. You also have the choice of "stepping" through your program, line-by-line.

Another option, available only in TD, lets you "animate" your program. Animation is a slow-motion form of the run command, where you can watch a succession of statements execute in "slow motion." (With ID and TD you can only go forward through your program; you can't back-up or retrace your steps.)

> I know this sounds like old advice, but it's truer of C than for almost any other language: the best debugger you have is your eyes.

Breakpoints

You set breakpoints via hot-keys or menu. Simply move the cursor (in the source-display window) to the line where you want to stop execution and press a key (or make a menu selection). Each time the debugger reaches that line, execution will stop. You can also point at a variable, press a key, and have it placed in your watch window, so that you can see exactly how the value changes as your program executes.

ID was designed for debugging common problems. While it does everything I've noted so far, it lacks the more sophisticated features found in its big brother. For the average hobbyist, or for someone who is just learning to program in C, the integrated debugger is quite adequate.

Additional Turbo Debugger Features

The rest of you will probably want to purchase the Professional package. Turbo Debugger is ideal for fixing large multi-module and multi-language programs.

TD supports tracepoints and complex watchpoints. A watchpoint in TD can be a conditional statement, such as "i > 10". This will cause a program break when i exceeds 10. Or you can put a "pass count" on a breakpoint — the pass count tells the debugger how many times to ignore the breakpoint before stopping.

Many advanced programs are written in more than one language. It's not unusual to add assembly language routines to C programs and Turbo Debugger automatically switches into assembler mode when it encounters such code.

You can also debug your programs in assembly mode, so that you can see the machine instructions the compiler created. This may not be necessary for debugging, but it can help you learn assembly language mnemonics and programming techniques.

Hot Keys And Menus

TD knows what you're doing and what you have already done. While Microsoft's CodeView relies upon text commands, Turbo Debugger gives you hot keys and menus.

You set a breakpoint on the current source line by pressing CTRL-F8, or by selecting breakpoint from the menu.

Context-sensitive pop-up menus appear when you press CTRL-F10. They contain commands which are pertinent to the location and the situation.

All of this makes TD very easy to learn and work with. The only thing missing is mouse support.

Macros

TD keeps history lists which allow you to recall previously entered commands. If you are repeating a series of commands, you can record those commands in a keyboard macro. Pressing the keystroke associated with a macro will execute the commands contained in that macro.

Logging is a unique feature of TD. Whenever you run a program, TD keeps track of what has happened and stores that information in the log window. You can then examine the contents of the log window to see what events (breakpoints, variable changes, etc.) have occurred. If you've missed an important event, it'll probably show up in the Log window.

TD has an extra feature when used with an 80386-based computer. By using a special device driver, TD can run in a virtual partition in extended memory. This allows your program to be loaded in the lower 640K, where it will run almost exactly like it would without the debugger. In all fairness, for $199 you can purchase MagicCV (from Nu-Mega Technologies) which does the same thing for CodeView.

Some Conclusions

Borland provides some excellent tutorials (including example programs) which guide you step-by-step through debugging. Turbo Debugger has some of the clearest and friendliest documentation I've seen.

I can tell that Borland spent a lot of time studying CodeView and its users. Don't be surprised if Microsoft adds most of Turbo Debugger's features in the next version of CodeView.

Mundane Debugging

Now that I've talked about the latest in debuggers, I'd like to cover some other (sometimes forgotten) techniques.

I know that this sounds like old advice, but it's truer of C than for almost any other language: the best debugger you have is your eyes.

I recently studied one of my programs and found that it included a number of header files that weren't required! This certainly affected the compile time for this program, and it wasn't something a debugger would have caught.

Occasionally print out a copy of your source, and *read* it. Sometimes "desk checking" is faster than using a debugger. By the time you'd recompiled with debugger options and loaded the program into the debugger, you could have spotted the problem.

Build your programs in stages, if at all possible. The longer the program, the better it is to create and test portions before you paste it all together.

Run your compiler with *all* of the warnings turned on. Eliminate any warnings, even if they seem trivial. Compilers from Microsoft, Borland, and Zortech provide excellent warnings about mismatched pointers, unintentional assignments, and other nonfatal problems. Use function prototypes, and be sure to explicitly cast values in assignments between types.

C Explorations

Okay, okay — I said I wasn't going to include this section for another couple of issues. So I lied (grin)! I just wanted to point out a simple technique using the exclusive-or (^) operator in C.

It's possible in C to swap the values of two integral values without using a temporary variable. The standard way of swapping values is done like this:

```
void swap1(int *a, int *b)
   {
   int c;

   c = *a;
   *a = *b;
   *b = c;
   }
```

The XOR method does this:

```
void swap2(int *a, int *b)
   {
   *a ^= *b;
   *b ^= *a;
   *a ^= *b;
   }
```

If your compiler is smart enough, the second function is faster because it uses the processor's XOR instruction rather than a MOV (move). When two values are XORed together, one of the values can be retrieved by XORing the result with the other (presumably known) value.

There are other uses for this technique. It can, for instance, be used to create simple ciphers. Most languages won't let you do this, since they don't support bitwise operators. I wish I could take credit for inventing this trick, but it goes a long way back.

News And Reviews

Turbo C v2.0

Since next issue will have the big C compiler roundup in it, I won't spend too much time on Borland's latest release of Turbo C.

There have only been a few changes since version 1.5. The compiler is even faster; this may be the fastest compiler on the MS-DOS market. A new type, long double, is a 10-byte floating point number. The floating-point emulator itself has been made quicker. And, of course, there is a subset of Turbo Debugger built into the integrated environment. The only thing missing is OS/2 support.

Considering the current slow pace of the OS/2 market, I don't think this is very important (yet).

Turbo Assembler v1.0

Known as TASM, this is a clone of Microsoft's MASM Macro Assembler. TASM accepts the same source code and command-line options as MASM; in fact, they both assemble at about the same speed. TASM does have some extensions to the MASM 5.1 syntax, but these are mainly bells and whistles.

Borland's biggest improvement is in documentation. The tutorial is better organized than MASM's, and the documentation of the advanced microprocessors (80186/88, 80286, 80386) is longer and clearer in TASM's manual.

There is no integrated environment for TASM. I assume that Borland figured that assembly language programmers wouldn't use such an item. Rumor has it that Microsoft is working on a QuickMASM to be released sometime next year; I assume Borland will follow suit.

Microsoft QuickC 2.00

It's just entered beta-test, and is scheduled to be released in mid-November. Microsoft has told me that it will eliminate most of the problems people like me have been complaining about. For instance, QuickC 2.00 will support four or five memory models in the integrated environment, as opposed to the one (medium) supported now. There is talk of *finally* implementing a "tiny" or .COM model.

The biggest addition will be the new help system. While I haven't seen it yet, I'm told that QuickC will incorporate some Computer Assisted Instruction (CAI) features. According to my contacts at Microsoft, QuickC is squarely aimed at the new C programmer, and will have the tools necessary to make the learning curve easier. I'm eagerly awaiting the product, and hope to get it into the compiler roundup. By the time you read this, it should be out.

Finalizing The ANSI Standard

Yes, they've done it — the members of the X3J11 committee have sent the current draft C standard up a level for final approval as an official ANSI standard. Very few changes were made during the meeting in late September, and most of these involve minor clarifications of the document's wording. Hopefully, there should be an official announcement of a standard in December. Finally, I can stop using the terms "proposed" and "emerging."

Resources

Book Of The Issue

If you do any kind of scientific or engineering programming, *Numerical Recipes in C* by Press, Flannery, Teukolsky, and Vettering (Cambridge Press, 1988, ISBN 0-521-35465-X) is a necessity. This book is based on the classic volume *Numerical Recipes*, which was written for FORTRAN.

While I have some quibbles with the authors' opinions of C, this book does contain the most comprehensive collection of scientific algorithms anywhere. You get everything from sorting to interpolation to integration to Fourier transformations. They even have a source diskette with hundreds of programs and fragments.

This is great for scientifically-oriented college students. And, for those of us who enjoy recreational mathematics, this book can lead to hours of experimentation.

Micro Assembly: A Convention

Well, we've gone and done it — a group of us are putting on a convention! Micro Assembly is the name, Denver is the place, and Memorial Day weekend 1989 (May 27-29) is the time. The convention is being held at the Clarion Hotel, Denver Airport, and when you register, we'll send you a convention packet containing directions and a hotel reservation form.

Registration costs $35 before the convention, and $50 at the door. "Professional" conferences charge hundreds of dollars to keep working programmers out.

Since we have a limited membership, please register as early as you can. Send your name, full address, and telephone number along with a check or money order for $35 to: Micro Assembly, 2560 S. Hazel Ct., Denver, CO 80219.

What's going to be happening at Micro Assembly? The theme of the conference is "programming for its own sake." There will be presentations on all aspects of programming, including a wide variety of languages and applications. In fact, when you send in your registration form, please send along your suggestions for presentations and workshops. If you'd like to present something at the convention, tell us that too! We expect to have Microsoft, Borland, JPI, and others there.

That's All, Folks

If these columns get any longer, they'll be bigger than Dave's editorials! As I said above, next time around we'll have the Big C Compiler Comparison of 1989. And there will be more information on Micro Assembly.

♦ ♦ ♦

MASM 5.1, Assembler Extraordinaire

High Level Assembly Language

By Laine Stump
PC Tech
P.O. Box 128
Lake City, MN 55041

There are compilers and there are assemblers. Compilers generate one or more machine instructions per human instruction. Assemblers are pretty much one for one, which makes MASM 5.1 ah, well, a compiler-assembler. (You heard it here first.)

How many times have you thought of writing a program in assembly language, only to be intimidated at the last minute into whacking it out in Pascal once again?

"I don't know what hit me, Doc. I started thinking of all those [BP+4]s, recursion problems, and PUSH BXs. Then, sudden-like, my vision got real blurry and my temples started to pound. Next thing I knew, I'd managed to crawl out to the garage and grab the ax..."

Well, not to worry. Your days in the mental ward are over. With Microsoft's release last winter of MASM 5.1, assembly language programming makes much more sense. In these days of 250K address list programs and "high performance" software that crawls without a 10 MHz 286, I think that's good news.

Rather than bore you with a list of all the new features in 5.1, I decided to write a short example program in new MASM (5.1) and contrast it with what would have been required with old MASM (4.0 for me). It won't show all the new features, but hopefully it will be enough to convince you that you, too, can be an assembly language programmer (and if you already are, that you, too, should upgrade to MASM 5.1).

The example I've chosen is a classic: The factorial function (Fact). I wanted to write something recursive (self-calling) as an example of the usefulness of putting your data on the stack. I chose Fact because it's one of the simplest algorithms I could think of that could be done recursively. (I realize that Fact can be written much more efficiently with a non-recursive algorithm; but that isn't the point, is it?)

The Fact function, along with a main program to call it, gives me a chance to show several things: the new simplified segment directives, parameter passing (variable and value) and referencing, local variables, temporary labels, and a few new assembly time directives, as well as a macro I wrote that handles parameter passing on the stack in a much more readable fashion than normal.

Actually, I ended up with two examples: In Figure 1, Fact returns the result in register AX. In Figure 2, Fact returns the result in a VAR parameter (sometimes referred to as "passed by reference" or "passed by address"). First, I'll just take a hop through Figure 1, pointing out things of interest.

The .MODEL Directive

The first line of the listing has the .MODEL directive. .MODEL tells the assembler which memory model and parameter passing conventions it should use as default. In my example, I have chosen SMALL model (one code segment and one data segment) and PASCAL parameter passing conventions (parameters pushed in left-right order and cleared from the stack by the called function).

I could just have easily chosen LARGE mode (multiple code segments and multiple data segments) and C parameter passing conventions (parameters pushed right-left and cleared by the caller).

As well as initializing the assembler with default values for "NEAR" and "FAR" when none is explicitly given, .MODEL makes writing an assembly language routine callable by a high level language almost completely painless.

For example, it even puts an underscore at the beginning of all public symbols (if you choose C parameter passing). And it will set up the BP register to point to the parameters in the stack frame *and* define text macros so that you can refer to a variable as "COUNT" instead of as "word ptr [BP-2]."

I chose the PASCAL parameter convention for my example because, among other things, I can forget about clearing the stack after returning from a procedure call. Instead, the assembler inserts a RET xx instruction (return and subtract xx from SP) in place of each RET instruction. Besides, PASCAL convention is the standard for OS/2, so I may as well get used to it...

.TYPE and @Cpu

The second point of interest is inside the CALLS macro (which I'll discuss in a minute) where I use the two assembly time directives, .TYPE and @Cpu. These allow you to examine the properties of an expression, and of the target CPU, respectively.

In this example, I use the two directives to determine if I should generate a normal PUSH Param instruction or if I should first move Param to a register and then push it. This is sometimes necessary because processors earlier than the 80186 did not have an instruction to push a constant value. So I check to see if Param is a constant and if the CPU is lower than a 186 and, if so, I do the push through a register. As an example, if Param was "1," the instructions:

```
MOV  AX,1
PUSH AX
```

would be generated if assembly for 8086 CPUs was enabled, and

```
PUSH 1
```

if it's an assembly for 80186, 80286, or 80386 CPUs. On the other hand, if Param was "word ptr ES:[BX]," the instruction:

```
PUSH word ptr ES:[BX]
```

would be generated, for any CPU.

In my examples, I've enabled 80286 real mode instructions with the .286c directive. Remember that this decision is made at assembly time, *not* at run time! If you can't be certain that your program will be running on a certain minimum level of machine, you should always use 8086 instructions (the default).

MASM 5.1's new simplified segment directives Bufferin nearly all the segment headaches.

Figure 1 — First Version of Fact, Passing Result in AX

```
         .MODEL SMALL,PASCAL
         .286c

CALLS    MACRO   Routine, Parameters
  IRP    Param,<Parameters>
    IF   (((.TYPE Param) AND 4) AND ((@Cpu AND 2) EQ 0))
         MOV     AX,Param          ;constants cannot be directly
         PUSH    AX                ;pushed on cpu's below 186
    ELSE
         PUSH    Param
    ENDIF
  ENDM
         CALL    Routine
ENDM

         .STACK  1024
         .CODE
Fact     PROC    NEAR USES BX, Number:WORD
         CMP     Number,1
         JNE     @F
         MOV     AX,1
         JMP     Fact99
@@:      MOV     BX,Number
         DEC     BX
         CALLS   Fact,<BX>
         ADD     AX,Number
Fact99:  RET
Fact     ENDP

MAIN     PROC    NEAR
         CALLS   Fact,<5>          ;returns fact(5) in AX
         MOV     AX,4C00h
         INT     21h
MAIN     ENDP

END      MAIN

***
```

I was reminded of this when, while testing operation of the CALLS macro, I enabled 80386 instructions. The program assembled with no errors. But when I tried to run it, CodeView came up with an "Illegal Instruction" box. Closer inspection revealed that MASM had used a conditional near jump, which is only available on the 386 (earlier processors only have short conditional jumps). After changing to .286c and reassembling and linking, everything worked fine.

Simplified Segments

I remember that my biggest confusion when I wrote my first 8086 assembly language program was what the hell to do with the SEGMENT and ASSUME directives. They seemed to me, with my Z80 background, to be some kind of black magic mumbo jumbo thought up just to confuse and discourage people like me from getting mixed up in the whole affair.

Since then, I've learned the minute intricacies of SEGMENT and ASSUME, and how to use them. But, most times what I'm doing doesn't require all those intricacies, and it's really just extra typing and extra symbol names to think of. I usually put all my code in a single segment called CODE and my data in a single segment called DATA.

MASM 5.1's new simplified segment directives Bufferin nearly all the segment headaches. Of the directives, the three most commonly used are .CODE, .DATA, and .STACK. In reality, these directives are text macros that do the segment and assume declarations for you.

For example, .CODE is equivalent to:

```
_TEXT  SEGMENT WORD PUBLIC 'CODE'
ASSUME CS:_TEXT
```

The .STACK directive accepts a size argument (in bytes). If you put a .STACK directive in your program, SS:SP will automatically be initialized at execution time to point to the top of the stack segment.

The old SEGMENT and ASSUME directives are still there, however, for all us diehards.

Parameter Passing And Register Saving

Next in our list of Fact features is the PROC declaration for the function itself. This shows two new features of PROC: automatic register saving, and parameter passing.

The new USES keyword lets you specify a list of registers to automatically save on entry to the procedure and restore before returning. The assembler will make sure all listed registers are pushed at the entry of the procedure, and popped before any RETs.

After the optional USES section of the PROC declaration, comes the optional parameter section. In this section you put a list of all parameters to the procedure along with the type of each parameter. The type can be WORD, DWORD, FWORD, QWORD, TBYTE, any structure defined by a STRUC, or a NEAR or FAR PTR (pointer) to any of these types. Arrays are not supported by PROC (unless you use the C trick of declaring a STRUC which contains an array).

Declaring the parameters of the procedure in this fashion has the same effect as declaring text macros like

```
Number equ <word ptr [BP+4]>
```

so that you can just say

```
CMP Number,1
```

instead of

```
CMP word ptr [BP+4],1
```

Oh, yeah. I almost forgot to tell you that MASM also automatically sets up the BP register to point into the stack frame. All PROCs automatically have the code

```
PUSH BP
MOV  BP,SP
```

inserted at the beginning, and

```
MOV  SP,BP
POP  BP
```

just before any RETs.

Declaring the parameter in the PROC declaration, and including type information, such as PTR TO DWORD, has the added advantage of allowing MASM to send the necessary information to the CodeView debugger so that you can use the name of the variable for expressions and watchpoints while debugging.

Temporary Labels

The next unusual thing you'll notice is that funny little label "@@:". This is called a "temporary" label. Its main advantage is that it doesn't take up space in the symbol table in memory, or clutter up the listing of symbols at the end of the LST output or the MAP file. It can be referenced with @F, which jumps Forward to the next @@ label, or @B, which jumps Backward to the most recent @@ label.

The CALLS Macro

While I was looking through books trying to come up with a topic for this issue, I kept coming across examples of calling OS/2 library routines and other high level language routines. What was actually happening was confused by a plethora of PUSH instructions stringing down the page. I decided that was ugly, and I needed a way to make procedure calls in assembly at least somewhat reminiscent of a high level language (where the PUSHes are implicit in the syntax of the language).

What I came up with after a bit of playing around, was CALLS (CALL using Stack). It allows placing the parameters all on a single line with the call to the procedure. While it isn't perfect (it doesn't know the difference between VAL and VAR parameters, for example), it can make a program much more readable. For instance, the lines:

```
MOV   AX,Seg Array
PUSH  AX
MOV   AX,Offset Array
PUSH  AX
PUSH  FirstNum
PUSH  LastNum
CALL  Xlate
```

Figure 2 — Second Version of Fact, Passing Result on Stack

```
.MODEL SMALL,PASCAL
        .286c

CALLS   MACRO   Routine, Parameters
  IRP   Param,<Parameters>
    IF  (((.TYPE Param) AND 4) AND ((@Cpu AND 2) EQ 0))
        MOV     AX,Param  ;constants cannot be directly
        PUSH    AX        ;pushed on CPUs below 186
    ELSE
        PUSH    Param
    ENDIF
  ENDM
        CALL    Routine
ENDM

        .DATA
StaticResult    DW      ?
        .STACK  1024
        .CODE
Fact    PROC    NEAR USES AX ES BX, Number:WORD, Result: FAR PTR WORD
        LES     BX,Result        ;load address @Result to ES:BX
        CMP     Number,1         ;IF (Number = 1)
        JNE     @F
        MOV     word ptr ES:[BX],1      ;THEN Result := 1
        JMP     FACTEND
@@:     MOV     AX,Number
        DEC     AX
        CALLS   Fact,<AX,ES,BX> ;ELSE Result := Fact(Number-1)
        MOV     AX,Number        ;     + Number
        ADD     ES:[BX],AX
FACTEND:RET
Fact    ENDP
MAIN    PROC    NEAR
LOCAL   Result: WORD             ;allocate var on stack
        MOV     AX,@data         ;point DS to .DATA segment
        MOV     DS,AX
        LEA     BX,Result        ;calculate (offset Result)
        CALLS   Fact,<5,SS,BX>   ;returns fact(5) in SS:BX (Result)
        CALLS   Fact,<6,seg StaticResult, offset StaticResult>
                                 ;returns fact(6) in StaticResult
        LEA     BX,StaticResult
        CALLS   Fact,<7,DS,BX>   ;returns fact(7) in StaticResult
        MOV     AX,4C00h
        INT     21h
MAIN    ENDP

END     MAIN
***
```

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: R/O"**? 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/user selection using either a colon or semi-colon, 31 user areas, drive search path, multiple program command line, archive bit maintanence, and transparent time/date stamping; all in the same space as CP/M 2.2. Installs from a convenient customization menu, no software assembly required. Bootable disks available with CBIOS for Kaypro, Xerox (8" or 5¼", -1 or -2), & BBI.

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 syncronous mode
- Speeds of up to 1,920 bytes/second in asyncronous mode
- Local/remote disk drive and printer support
- Remote peripheral support for modems and real-time clocks
- All stations need not be on the network even though connected
- Local drive access protection and control
- Simple menu oriented configuration utility
- Extended DOS calls are provided for addition of custom network utilities.

Works with interrupt driven Z80 systems such as Xerox 820, Kaypro (KayPLUS & Advent ROMs), Eagle, and other computers running QP/M, or CP/M 2.2

QP/M Network File System	$ 39.95

Hard Disks

Need more speed and storage on your system?

Improve the productivity of your Z80 computer with a hard disk.

HDS Host Board

This daughter board provides a convenient interface for connecting a Western Digital WD1002-05 hard disk controller to your computer.

- Plugs into the Z80 socket, no other wiring required
- 40 pin interface for a WD1002-05 (or HDO) controller board
- Switch selectable I/O port addressing
- Comes as bare board or assembled & tested
- Kaypro '84 host board also available

Winchester Connection by MICROCode Consulting

The most simple and comprehensive hard disk software package available for CP/M.

- Designed for use with the WD1002-05 controller board
- Works with one or two hard disks - 5 to 64 meg
- Menu installed, no software to assemble
- Complete hardware tests and error handling
- Automatic swap, for warm boots from hard drive
- Software drivers install above or below CP/M
- Allows custom partition sizes and mixed drive types
- Independent block and directory sizes on each partition
- Includes manual, format, test, park, and swap utilities

Winchester Connection Software only	$ 39.95
HDS Board with Winchester Connection Software	$ 79.95
HDS Bare Board with software	$ 59.95
HDS Board, WD1002-05, and software	$245.00

Call or write for other pricing options

WD1002-05 HARD DISK CONTROLLER BOARD by Western Digital

- Standard ST506 drive interface
- Same size as standard 5¼" drive
- 40 pin interface to host computer
- WD2797 floppy disk controller interface on board
- Can control up to three hard drives
- Direct replacement for **Kaypro 10** controller

WD1002-05 Controller Board	$185.00

Other Western Digital boards available

Prices subject to change without notice. VISA and Mastercard accepted. Include $5.00 shipping and handling, $7.50 for COD, UPS-Blue or RED Label additional according to weight. Please include your phone number with all correspondence.

Kaypro

KayPLUS ROM Set by MICROCode Consulting

Want more performance and flexibility from your Kaypro? With the KayPLUS ROM set you can have the advantages of a Kaypro 4 or 10, even on your Kaypro 2.

- Install up to four floppies and two hard drives
- Boots from floppy or hard disk
- Supports 96 TPI and 3½" disk drives
- Can use any ST506 type hard drive - 5 to 64 Meg
- 32 character type-ahead keyboard buffer
- Automatic screen blanking (not avail. on 83 series)
- 12 disk formats built-in, unlimited configurable
- Full automatic disk relogging with QP/M
- Internal real-time clock support
- No software assembly required

Includes manual, format, configuration, diagnostics, sysgen, diskette customization utility, AND hard disk utilities. Available for '83 and '84 series Kaypros.

KayPLUS ROM Set, specify model	$ 69.95
KayPLUS ROM Set with QP/M	$125.00

Parts and accessories for the Kaypro

Kaypro 2X Real-time Clock parts kit	$ 29.00
Kaypro 2X Hard disk interface parts kit	$ 16.00
Kaypro 10 or '84 series Hard Disk host board	$ 49.00
Kaypro four drive floppy decoder board	$ 35.00

Complete parts and repair services available

Xerox 820

PLUS2 ROM and X120 Double Density Board by MICROCode Consulting and Emerald Microware

About had it with single density diskettes on your Xerox 820-1? Get unsurpassed versatility with our X120 Board and PLUS2 ROM package.

- Run up to four floppy disk drives at once
- Mix 8" and 5¼" at the same time
- Software compatible with Kaypro and Xerox 820
- Built in drivers for most serial and parallel printers
- Get mini-monitor functions and auto-boot capability
- 19 built in disk formats, including Xerox and Kaypro
- Includes custom disk format definition program
- Banked ROM BIOS for more space in your TPA
- Composite video adaptor on X120 board
- Runs 48 TPI diskettes on 96 TPI drives
- Supports real time clock from Z80-CTC
- Works on the Xerox 820-1 and Big Board I
- Both ROM and X120 board are required for operation

PLUS2 ROM Set and X120 Board A&T	$114.95
PLUS2 ROM Set and X120 Bare Board	$ 49.95
PLUS2 ROM Set only	$ 39.95
120 Bare Board only	$ 15.00

★★★ **Special** ★★★ 2 boards for $25, 5 for $50

Other kits, parts, and packages available

Parts and accessories for the Xerox 820

Xerox 820-2 CPU Board - new	$ 75.00
Xerox 820-2 Floppy Controller board - new	$ 65.00
Xerox 820-2 CPU board w/ Floppy Controller	$125.00
Xerox 820-1 CPU board - new	$ 75.00
Xerox 820 complete high profile keyboard	$ 65.00
Xerox 820 bare high profile keyboard - new	$ 25.00
Xerox 820 5¼" drive cable	$ 9.00
Xerox internal video cable w/brightness control	$ 9.00
Xerox 820 power supply	$ 35.00
Power connector, specify board or cable	$ 2.50
Xerox parallel printer interface cable	$ 35.00
Dual Half Height 5¼" Disk Drives - DSDD, in cabinet with standard Xerox cable	$265.00

Complete parts and repair services available

EMERALD MICROWARE

P.O. Box 1726, Beaverton, OR 97075

VISA **(503) 641-0347** MasterCard

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 disks and programs on your MS-DOS computer just as you would a standard MS-DOS diskette. Once the UniForm driver 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 275 CP/M and MS-DOS computers on your PC, XT, or AT. With UniForm-PC and the Compaticard, you can use 5¼" high density, 96TPI, dual format 3½" (720k/1.44 meg.- PS/2), and even 8" drives.

UniForm-PC by MicroSolutions	$ 64.95
Uniform for Kaypro and other machines	$ 64.95

CompatiCard by MicroSolutions

Meet the CompatiCard, THE universal disk drive controller card. This half card will let you run up to 16 disk drives (4 per CompatiCard) on your PC or XT, including standard 360K, 96 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 driver and format program for high density and 3½" diskettes. Use it with UniForm-PC for maximum versitility. 8" adaptor and additional cabling available.

CompatiCard Board	$169.95
CompatiCard with UniFORM-PC. . ★★ **Special** ★★. .	$225.00
CompatiCard with UniFORM-PC & high density or 3½" drive . . . . . . ★★★ **Special** ★★★	$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	$139.95
MatchMaker w/External Mac Drive	$325.00

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 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 Z80 code programs or for systems without a V20. UniDOS directly converts video and keyboard 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 8088 CPU chip that includes a full 8080 instruction set as well as the standard 8088 set. Systems using an 8086 may substitute a V30 chip.

UniDOS by MicroSolutions	$ 64.95
UniDOS w/UniForm and V20-8 chip	$135.00

UniDOS Z80 Coprocessor Board by MicroSolutions

This **8 Mhz. Z80H** half-card will run your Z80 and 8080 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 Z80 Coprocessor Card	$169.95

Reader Service Number 10

could be replaced with:

```
CALLS Xlate,<Seg Array,OffsetArray,
      FirstNum, LasNum>
```

Things can get more complicated if you want to pass the address of a parameter (see Figure 2 for an example).

Figure 2

I made the second version of Fact to show two more concepts that I hadn't covered in the first version: 1) local variables, and 2) variable parameters.

LOCAL

Variables can be declared local to a procedure (allocated on the stack) with LOCAL. I used LOCAL in the second version of Fact to allocate a variable local to MAIN. Local variables can be any of the types accepted by PROC, as well as single-dimensional arrays of those types.

LOCAL does three things: 1) It generates an instruction to subtract the size of the declared variable(s) from SP, allocating space on the stack; 2) For all language models except C, it remembers the size of the variables and later automatically generates the proper RET xx instructions to free the space and readjust the stack; and 3) it produces a text macro for you to use in addressing the new variable. For example:

```
LOCAL Result:WORD
...
RET
```

is equivalent to:

```
SUB    SP,2
Result EQU <word ptr [BP-2]>
...
RET    2
```

Notice that I use the name Result in two different places in the program.

MASM 5.1, unlike earlier versions, knows about variable scope. I can use the same name as many times as I like, as long as each occurrence is restricted to a different procedure.

PROC parameters are also local to the procedure in which they are declared (as are labels within the procedure if you specify a language with .MODEL). This is really nice, because it means that I don't have to think of silly names anymore just to keep from having duplicate labels.

Easy Recursion

A nice thing about the way local variables are handled by MASM, and the reason I was able to write Fact using recursion, is that these local variables are allocated on the stack. By using only variables of this type and declaring all register usage with USES, you can easily write reentrant routines.

You may *think* that reentrant code isn't important, but I can tell you that I've spent weeks tracking down problems caused by non-reentrant code. Until I figured out the problem, I didn't even realize that there was recursion in the program, because it was indirect (you know, PROCA calls PROCB calls PROCC calls PROCA) and only occurred under certain conditions.

> Although MASM takes care of allocating stack space for the pointer, you must still write the code for dereferencing the pointer. And, just as important, but sometimes not quite so obvious, you must write the code for determining the address to put into the pointer.

After all, there must be *some* reason why Nicklaus Wirth, Kernighan, Ritchie, and all those other guys decided that code should be reentrant.

Variable Parameters

The second difference in Figure 2 is the passing of the result of Fact as a variable parameter instead of through the AX register. Any parameter declared as "PTR xxxx" is a variable parameter. The effect is that a pointer to the variable is passed instead of the contents of the variable. This allows the called routine to modify the original variable instead of just a copy of it.

Now comes the slight difficulty of working with MASM in contrast to Pascal or C. Although MASM takes care of allocating stack space for the pointer, you must still write the code for dereferencing the pointer. And, just as important, but sometimes not quite so obvious, you must write the code for determining the address to put into the pointer.

Dereferencing (a fancy word for "getting the contents of") a pointer on the stack is easy, especially because MASM has declared the text macro to let us address Result as "Result" instead of as "Dword ptr [BP+4]." We just do an LES BX,Result, which loads ES:BX with the address of the variable we want to look at and modify. Then when we want to do something with it, we call it ES:[BX] instead of Result.

If I had used default Small model pointer sizes when declaring Result, I could have just used MOV BX,Result, then referred to the variable as [BX].

Determining the effective address can be a bit more confusing.

My first reaction was to use "offset Result" when making the call to Fact. This would work fine if Result were a "direct reference" (i.e., not relative to an index register), but all variables on the stack are relative to BP, and "offset [BP-2]" makes no sense to the assembler (since the value of BP can't be known until runtime). That's where the LEA (Load Effective Address) instruction comes in.

LEA takes any memory reference as its second operand, calculates the offset, and places that offset in the variable specified by the first operand. So, if BP is currently 55FCh, the instruction

```
LEA BX, [BP-2]
```

will put the value 55FAh in register BX.

Note that using LEA is unnecessary (although it doesn't do any harm) when the variable is a direct address. See the call to Fact which places the result in StaticResult for an example of using direct addresses with, and without, LEA.

What????

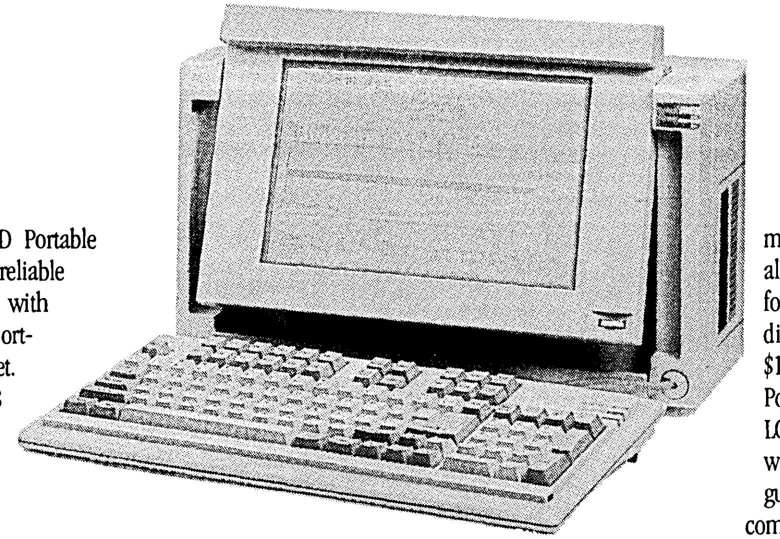
Probably some of this gobbledy-gook doesn't make much sense to you right now. I suggest that you assemble both

McTek

LCD Portable

(Also available in PLASMA PORTABLE)
(Including Hard Disk Only 19 lbs.)

RABBIT 286



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 ∅ wait state, it includes a 20MB hard disk, 720KB 3½″ floppy drive, parallel & serial ports, Award 3.03 bios, 640k & turbo indicator LCD. The screen is a fantastically readable, electroluminescently 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 5¼″ floppy port for reading and converting to 3½″ disks (5¼″ 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 ***$1799!***

386-20 MHz W/IMB ***$2899!***

3 MB On-Board AT!

Our McTek 286A is the most integrated AT-compatible to date. It utilizes the highly regarded Chips & Technology chip set, and includes memory upgradable *on board* to 3 megabytes. No more worries about speed compatibility with expanded memory cards! The 8/10 MHz, ∅-wait state McTek 286A runs at 11.5 Norton SI, and an effective 13.2MHz on the Landmark test. Serial, parallel & game ports are all standard on board. With Award 3.01 bios, 640k, 200W power supply, Samsung amber monitor with Hercules-compatible controller, locking case, AT-style keyboard, 1.2MB drive, 20MB Seagate. Assembled & fully tested, with a full one-year warranty. Get in on the most advanced AT-compatible on the market, at the lowest price ever offered! ***$1399!!***

286-16 MHz W/IMB ***$1699!!***

XT Turbos & Supers

640k 4.77/8MHz and 4.77/10 switchable XT turboboards; two 360k floppy-disk drives with controller; one parallel, one serial and one game port: AT-style keyboard; clock, FCC-approved slide-case; eight slots; Hercules-compatible graphics card; amber monitor w/base; fully assembled and tested; one-year parts *and* labor warranty.

$729 **XT Turbo 4.77/8MHz Complete**

Superturbo 4.77/10 MHz Complete **$779**

McTek Systems, Inc. • 1411 San Pablo Avenue • Berkeley, CA 94702 • 415-525-5129

DISK DRIVES	
Fujitsu 360k	$69
Fujitsu 1.2MB	$89
Teac	$75
Teac 1.2MB	$95
Toshiba 3½″ 720K	$89
Teac 3½″ 1.4MB	$109
20MB Hard Disk Kit	$279
30MB Hard Disk Kit	$309
ST-225	$215
ST-238	$239
ST-138 30MB	$399
ST-251 40MB	$369
ST-125 20MB 3½″	$279
ST-157 49MB 3½″	$479

PRINTERS	
Citizen CD 120	$159
Citizen CD 180	$189
HPLASAR Serial2	$1699
Epson LX-800	$219
Epson LQ-500	$379
Toshiba 321 XL	$559
NEC P2000	$369
Call for prices of other brands	

MODEMS	
Everex int. 300/1200	$79
Everex 2400 external	$195
Everex 2400 internal	$179

MONITORS	
Samsung amber	$79
Samsung EGA color	$359
Samsung RGB color	$259
NEC Multisync	$559
Sony Multiscan	$619
HGC-compat.mono card	$49
Color graphic card	$49
EGA Paradise 480	$149
VGA Paradise	$279
Genoa Super VGA	$299

MOUSE	
Logimouse C7	$69
Logimouse H1 Res.	$99

PC/XT	
640k TurboMothrbrd	$85
10MHz TurboMothrbrd	$89
Multi I/O w/disk contrlr	$59
640k RAM card	$39
2MB Expansion card	$89
RS232 2-port card	$35
4-serial port card	$79
Game I/O card	$15
384k Multifunction card	$69
FCC-app. slide XTcase	$29
150W power supply	$49
XT keyboard	$42
Clock Card	$19
Floppy Controller	$19

PC/AT	
McTek286-20MHz	$559
Baby McTek 286B-AT 8/10 O-wait	$269
McTek 386-16MHz	$899
McTek 386-20MHz	$999
McTek 386-24MHz	$1099
Locking slide case	$59
200W power supply	$65
Enhanced keyboard	$59
WD FD/HDC	$129
DTC FDC/HDC 1:1	$189
3MB EMS (0K)	$99

DESKTOP

MISC.	
Kingtech CRT Portable Kits: XT/AT (power supply, case keyboard, monitor)	$380/$410
Eprom burner 4-socket	$139
LCD Portable	$799
Plasma Portable Kits	$1499
AC power strips	$15
Diskette file box	$9
Printer or serial cable	$8
Archive Tape Backup 40MB	$339
386-20 MHz Desk Top	Call
286-20 MHz Desk Top	Call

Reader Service Number 42

of these programs and step through them with CodeView, watching the variables to see what's really happening. While you're at it, you can take a quick look at the "Mixed" display (showing both source code and corresponding assembly language) to see the instructions generated by the PROC and local directives, as well as the CALLS macro.

The command lines for assembling either program are:

```
masm /zi fact;
link /co fact;
```

The /zi and /co options instruct the assembler and debugger to include debugging information in the object files.

An Interesting Show

I was totally amused for at least an hour by calling up version two of the program, typing "g Fact" to get within the scope of the parameters of Fact, then setting watches (dynamic displays of chosen points in memory) on "Result," "*Result," and "Number." Then I used the Trace command (F8) to step through the recursive calls to Fact one at a time, watching the changes to the variables.

Notice that the actual variable displayed in the watch window for, say, "Number," changes with progressive calls. It always shows the current activation of Number.

When a variable is no longer valid (for example, after you return to MAIN), the display will show an error message for that variable. It's something like "out of scope." Assembly language has truly never been at a higher level.

Oh, yeah. While you're stepping through the program, pull down the "Calls" window now and then to see the nesting of calls and which parameters were passed during each call.

After a bit of playing around, you'll probably be as excited as I am.

More Goodies

By the way, I haven't mentioned MASM 5.1's wonderful reference books — including the *Mixed Language Programming Guide*, which gives explicit instructions for writing assembly routines which are callable from the Microsoft high level languages (Pascal, C, QuickBasic, FORTRAN), as well as how to call any one of those languages from any one of the others.

> When a variable is no longer valid (for example, after you return to MAIN), the display will show an error message for that variable. It's something like "out of scope."

I bet more than a few of you have spent much more than the cost of MASM 5.1 trying to figure out these details yourselves.

Also included is everything necessary to write assembly language programs for OS/2. And they include OS/2 versions of all the MASM utilities (LINK, CREF, LIB, EXEPACK, etc.)

And finally, of course, there's CodeView. Ahh. CodeView. Never has such a useful program been so fun. It has its problems (operation of the mouse scroll bar is very erratic, and it doesn't recognize any large screens other than EGA, for starters), but overall I have never seen a programming tool with as much potential. I mean, it will even debug programs written in multiple languages, switching automatically from C to Pascal to Assembly and back. All in source mode!

And, the OS/2 protected mode version of CodeView can debug programs with multiple threads (processes).

Ehh??

So what does all this mean? It means, forget about that copy of MASM 1.0 that your grandmother gave you. (She did give you one, right?) Forget about CHASM and SPASM and RASM and RAVINE. Don't even bother with MASM 4.0. Even if you haven't already bought a previous version of MASM and don't qualify for a special update price, call up the Programmer's Shop or Programmer's Paradise or Egghead Software (or any other shop with a good price) and order MASM 5.1.

And, as Al Franken always says: "You'll be glad you did."

♦ ♦ ♦

U.S. Postal Service Statement of Ownership

U.S. Postal Service Statement of Ownership, Management and Circulation (Required by 39 U.S.C. 3685) 1A.)Title of Publication: MICRO CORNUCOPIA 1B.)Publication Number: 0747-587X 2.)Date of Filing: 9-29-88 3.)Frequency of Issue: Bi-monthly 3A.)Number of Issues Published Annually: 6 3B.)Annual Subscription Price: $18.00 4.)Location of Known Office: 155 NW Hawthorne, Bend, Oregon 97701-2917 5.)Location of the Headquarters or General Business Offices of the Publishers: 155 NW Hawthorne, Bend, Oregon 97701-2917 6.)Name and Complete Address of the Publisher, Editor, and Managing Editor: Publisher: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917; Editor: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917; Managing Editor: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917 7.)Owner: Micro Cornucopia, Inc. PO Box 223, Bend, Oregon 97709-0223; David J. Thompson 1259 NW Iowa, Bend, Oregon 97701-1001; Sandra S. Thompson 1259 NW Iowa, Bend, Oregon 97701-1001 8.)Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1% or More of Total Amount of Bonds, Mortgages or Other Securities: None 10.)Extent and Nature of Circulation: Average Number of Copies Each Issue During Preceding 12 Months A.)Total Number of Copies Printed: 31,505 B.)Paid Circulation: 1)Sales Through Dealers and Carriers, Street Vendors, and Counter Sales: 13,118 2)Mail Subscription: 9,820 C.)Total Paid Circulation: 22,938 D.)Free Distribution by Mail, Carrier, or Other Means, Samples, Complimentary, and Other Free Copies: 4,587 E.)Total Distribution: 27,525 F.)Copies not Distributed: 1)Office Use, Left Over, Unaccounted, Spoiled after Printing: 838 2)Returns from News Agents: 3,142 G.)Total (Sum of E, F1 and 2)-Should Equal Net Press Shown in A: 31,505; Actual Number of Copies of Single Issue Published Nearest to Filing Date: A.)Total Number of Copies Printed: 30,123 B.)Paid Circulation: 1)Sales Through Dealers and Carriers, Street Vendors, and Counter Sales: 17,652 2)Mail Subscription: 10,233 C.)Total Paid Circulation: 27,885 D.)Free Distribution by Mail, Carrier, or Other Means, Samples, Complimentary, and Other Free Copies: 514 E.)Total Distribution: 28,399 F.)Copies not Distributed: 1)Office Use, Left Over, Unaccounted, Spoiled after Printing: 1,724 2)Returns from News Agents: 0 G.)Total (Sum of E, F1 and 2)-Should Equal Net Press Run Shown in A: 30,123

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers)	$500
	Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
	CQL Query System (SQL retrievals plus windows)	$325
	GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$250
NEW!	Greenleaf Business Mathlib (exact decimal math, formatting, depreciation, interest, cash flow, statistics)	$250
	Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$220
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$175
	TurboTeX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTeX)	$170
NEW!	Sherlock (C debugging aid)	$170
	Essential resident C (TSRify C programs, DOS shared libraries)	$165
	Greenleaf Functions (296 useful C functions, all DOS services)	$160
	Essential C Utility Library (400 useful C functions)	$160
	Essential Communications Library (C functions for RS-232-based communication systems)	$160
	WKS Library Version 2.0 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	ME Version 2.0 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$130
	TurboGeometry (library of routines for computational geometry)	$125
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$115
	Minix Operating System (U**x-like operating system, includes manual)	$105
	PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
NEW!	Tele Operating System (TeleKernel, TeleWindows, TeleFile, & TeleBTree by Ken Berry)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
	Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	$80
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
NEW!	Polyglot Lisp-to-C Translator (includes Lisp interpreter, Prolog, and simple calculus prover)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
NEW!	Kinetic Image Synthesizer (3-D animation system ... Saturday morning on your PC!)	$75
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$75
NEW!	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
	lp (flexible printer driver; most popular printers supported)	$65
	Quincy (interactive C interpreter)	$60
	PTree (parse tree management)	$60
NEW!	Icon-Tools (full-featured icon display and editing system)	$60
NEW!	Polyglot TSR Package (includes reminder, bookmark, virus catcher, cache manager, & speech generator)	$50
	HELP! (pop-up help system builder)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
	Make (macros, all languages, built-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
NEW!	Heap I/O (treat all or part of a disk file as heap storage)	$40
	Biggerstaff's System Tools (multi-tasking window manager kit)	$40
	PC-XINU (Comer's XINU operating system for PC)	$35
	CLIPS (rule-based expert system generator, Version 4.2)	$35
	Tiny Curses (Berkeley curses package)	$35
NEW!	Polyglot RAM Disk (change disk size on the fly; includes utilities)	$30
	SP (spelling checker with dictionary and maintenance tools)	$30
	Clisp (Lisp interpreter with extensive internals documentation)	$30
	Translate Rules to C (YACC-like function generator for rule-based systems)	$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
NEW!	Pascal Compiler & Interpreter (P-codes, standard Pascal)	$25
	ICON (string and list processing language, Version 7)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX)	$25
	LEX (lexical analyzer generator; an oldie but a goodie)	$25
	Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
	AutoTrace (program tracer and memory trasher catcher)	$25
NEW!	Data Handling Utilities in C (data entry, validation & display; specify Turbo C or Microsoft)	$25
	Arrays for C (macro package to ease handling of arrays)	$25
NEW!	ANSI Forms (forms manager based on ANSI codes)	$20
	C Compiler Torture Test (checks a C compiler against K & R)	$20
	Benchmark Package (C compiler, PC hardware, and Unix system)	$20
	A68 (68000 cross-assembler)	$20
	List-Pac (C functions for lists, stacks, and queues)	$20
	XLT Macro Processor (general purpose text translator)	$20
NEW!	KeySwap (swaps CapsLock and CRTL key functions on XTs and ATs)	$20
NEW!	OOPS (collection of handy C++ classes by Keith Gorlen of NIH)	$20
	C/reativity (Eliza-based notetaker)	$15

Data

DNA Sequences (GenBank 55.0 including fast similarity search program)	$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$70
Dictionary Words (234,932 words in alphabetical order, no definitions)	$60
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
KST Fonts (13,200 characters in 139 mixed fonts: specify TeX or bitmap format)	$30
USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	$20
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

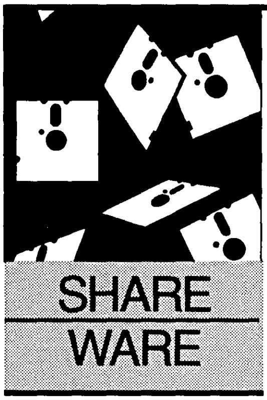
The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

acw!info@uunet.uu.net

Voice: (512) 258-0785
BBS: (512) 258-8831
FAX: (512) 258-1342

Free surface shipping on prepaid orders For delivery in Texas add 7% MasterCard/VISA

Reader Service Number 4



As The Disk Turns

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

There's slow change and there's dramatic change. There's good change and there's bad. When the courts get involved in this industry, it seems the results are usually dramatic — and always bad.

Has the noble undertaking gone sour? While shareware was created as an experiment in economics rather than as an exercise in altruism, user-supported software has always had an extra measure of friendliness and trust. "Share it with a friend." That was how it was supposed to work. Count on honest users to send in their fees.

This column, and others like it, exist because the shareware experiment is a success. The shareware constituency is large and, until recently. Now, unfortunately, the concept has been dragged into the courtroom.

The ARC Wars Escalate

I mentioned last time that Software Enhancement Associates (SEA) had sued Phil Katz of PKWare, alleging that Katz's PKARC and PKXARC infringe on SEA's trend-setting ARC utility. SEA claimed that PKWare had violated SEA's copyright on the ARC program code and misused the "ARC" trademark.

The funniest part of the lawsuit said that PKWare had copied ARC's "look and feel." What "look?" What "feel?" We're talking about programs that operate from the command line of the operating system prompt. If SEA is really serious about the whole, dubious "look and feel" business, they should promptly investigate whether they can also sue Digital Research for PIP and Microsoft for COPY.

Many observers (and your humble columnist) didn't take the part about program code very seriously. Why would Phil Katz borrow code from the sluggish ARC anyway? PKARC was over six times as fast as ARC in a benchmark I ran in my Jan.-Feb. 1988 column. Clearly the PKWare is *much* more efficiently coded.

Despite the obvious differences between the performance of the rival utilities, SEA and PKWare agreed to a settlement that was equivalent to Katz's tossing in the towel. PKWare agreed to withdraw its utilities, effective January 1989; turn its source code over to SEA in return for unspecified royalties; and to immediately rename its programs in the interim to remove the "ARC" nomenclature. (The PKWare products are now called PKPAK and PKUNPAK; by the terms of the agreement, Katz cannot distribute them after the January 1989 deadline.)

The settlement enjoins the two parties to say nothing beyond the joint press release they issued. Naturally there was speculation by interested observers. Rumors suggested that PKWare was caught red-handed with copied code in its products. Competing rumors held that Phil Katz had simply been beaten down by the potential expenses of defending his small company against the much larger SEA.

Of course, with the settlement under a court seal of confidentiality, we'll never know, will we?

Guess Again

The *Milwaukee Business Journal* published a report on the lawsuit that quoted from the confidential settlement agreement. Who was responsible for the leak? Apparently it was SEA — which is all the more remarkable since they were the ones insistent on sealing the outcome.

The papers filed by SEA in the public records of the Milwaukee County courthouse contained the complete settlement documents. They were discovered by Karen Little, a technical writer, who visited the courthouse and asked for the file on *SEA v. PKWare.* (Ms. Little is not a PKWare employee, although she was hired by Katz to prepare his documentation.)

Now the documents have been dumped onto BBSs across the country, where they travel in the company of a lengthening list of BBS sysops who are lining up in support of Phil Katz. The text of the agreement contains nothing to substantiate the claim that Katz borrowed code from ARC. In fact, many people have already pointed out that any similarities between ARC and PKARC could derive from the public domain algorithms that both used in their compression routines (though, as noted, Katz did a much better job of it).

The backlash against SEA is dramatic. Sysops point out that the ARC standard languished until PKWare provided speedy utilities that finally made archiving convenient. Eighty-five system operators have subscribed their names to the September 25 release of FIGHTSEA.ARC, an extensive memorandum that answers points raised in the original SEA lawsuit, criticizes the once-secret settlement, and lambastes SEA's follow-up contempt action.

Editor's note: Look for SEA.TXT on the Micro C BBS (503) 382-7643 and the Issue #45 disk.

Contemptible?

A contempt action? Yes, SEA now says that PKWare is violating the original agreement by continuing to use the terms "ARC" and "archiving."

Have I got news for SEA. When I worked for the California State Treasurer's Office, we used a Burroughs B-20 network that provided operating system utilities to "archive" files. Would SEA like to take on Unisys?

Michael Reaves of Computer Associates (CA) has signed on to the FIGHTSEA memo with the observation that he is the technical lead for CA's ARCHIVER program: "We use the words archiver, archive, and arc throughout the company, in speech, manuals, internal and external docs, etc.... I wonder if they will take on CA (the largest software-only firm in the world)."

This Way To The Ramparts

Judy Getts, contributing editor for *PC World* magazine, has taken a leading role in distributing the sysops' memo against SEA. The file contains the following message from Ms. Getts: "If you are a sysop or software author and would like to add your name to this list of endorsers, please leave a message containing your name, phone number, name of your BBS and/or product, and the name of the city that your board resides in, to Judy Getts on one of the following boards: Exec-PC in Milwaukee at 414/964-5160; Loren Jones' RBBS-PC in Chicago at 312/352-1035; or the Sound of Music in Oceanside, New York at 516/536-8723. We thank you."

In the meantime, you have until January 31st to register your PKWare and preserve your rights to use it — even under the strictest interpretation of SEA's settlement with PKWare. Write to:

PKWare, Inc.
7032 Ardara Avenue
Glendale, WI 53209

and enclose a check for at least $20.

Space Aliens On The ARC

SEA is not indifferent to the reaction to their legal maneuvers. They are now distributing a "policy statement" that sets forth conditions for the use of the term "ARC." One paragraph is particularly expansive: "We hereby grant to the entire world and all sentient creatures in the universe who do not already have an agreement with us to the contrary a perpetual, unlimited, galaxy wide license to read, extract, create, or otherwise manipulate ARC format archives. This does not include any license to use our sources or trademarks."

However, anyone who breathes the word "ARC" had better watch out for trademark infringement. SEA's policy statement provides that the ARC trademark will be licensed only for shareware products whose "fully commented source code" is submitted to SEA. This parallels the briefly confidential agreement whereby SEA acquired PKWare's source code.

Perhaps SEA's counteroffensive will dampen some of the criticism. I wouldn't bet on it. Given the window of opportunity under the terms of the SEA/PKWare agreement, we can all be registered users of PKWare's utilities by year's end and never have to touch a SEA product again.

A Prophylactic Postscript

Thanks to Deborah Rose of New Haven, Conn., for being the first to advise me that CHK4BOMB is the handiwork of Andy Hopkins. Because the program has been the target of crackers who corrupted the file, Hopkins has recently withdrawn CHK4BOMB from bulletin boards. If you don't have one of the older, pristine versions of CHK4BOMB, your best bet would be to contact Hopkins at:

Panda Systems
801 Wilson Road
Wilmington, DE 19803
(302) 764-4722

CHK4BOMB has been repackaged as part of the commercial set of programs called the Dr. Panda Utilities. The complete set of utilities costs only $50. However, you can get CHK4BOMB (now called LABTEST) for $20 (along with a second program called MONITOR). Thanks again, Deborah.

♦ ♦ ♦

CULTURE CORNER

The Rectangle At The End Of The Julia

By Dave Thompson

It was a sunny afternoon, warm, a small snowstorm lightly salting the ground. Larry hadn't noticed the weather, however, for he'd spent the entire day searching for a single practical application for fractals.

In the process, he'd found his way farther and farther down an obscure but attractively freckled arm of a Julia set.

Mathematically there was no end — he knew that. There were only further branches, more arms — deeper until the numbers became so small, the arms (and the freckles) so close together, that the computer could no longer tell them apart. It was then he would turn back, as he always had before.

This time, however, he seemed to be descending in a lateral sort of way. The decimal places were, for no apparent reason, holding their own rather than diminishing. Time after time his screen painted a new vision of organized chaos, each vision different, and yet, in substance, the same. It was deep inside this strange cavernous swirl of unreal numbers taken to imaginary powers that something caught his eye.

"What's the answer to life, the universe, and everything?" Larry typed.

Though it was but a mole on the toenail of the universe, he could tell it was different. A mere pixel on his massive screen, he knew that the pixel was rectangular. Now a rectangle, even an unusually small one, is certainly practical.

Carefully, he calculated new parameters and restarted the image. It was there again. No larger than a grain of sand amidst the crab nebula, it was still a rectangle.

Fifty times, a hundred times, he repeated the image, each time probing farther. Sometimes the rectangle eluded his calculations, other times he found it no closer, but finally at the end of a long, circuitous, but clean-shaven arm, it suddenly appeared. This time, however, it was a large flashing billboard.

Inside the billboard there was a question:

"What is your question?"

He paused. It wasn't often that pure research came up with something this obvious. However, he hadn't neglected other areas of research and so he was ready with the very most practical question.

"What's the answer to life, the universe, and everything?" Larry typed.

"Sounds familiar. Hold on a minute, I'll look it up," the rectangle answered.

"Sure you don't want something a little more interesting? I'd enjoy tackling the Pennant race, being something of an Oakland fan and all, " the rectangle continued.

"I'm sure," Larry responded.

"Okay, it's 43."

"You told Douglas Adams the answer was 42."

"It was. Then."

So now I've related to you Larry's practical application of fractals and, the very practical answer. I was going to have Larry write this up for this issue, but he's already groping his way down another of Julia's arms. I'm not sure what it is he expects to find there, but I'm not about to interrupt. After all, he does have a relatively current answer to the question of life, the universe, and everything.

Now all he needs to know — is what to do with it.

♦ ♦ ♦

KOALA COMPUTER

213-316-5866
9 to 6 PST M - Sat

30-Day Money Back Guarantee
Software sales are final except replacement of defective media. Merchandise must be undamaged for full refund.

!! VGA !! ATI VIP card and VGA Professional Color Monitor - $648

Analog 13" anti-reflective, non-interlaced, flicker free, .31 dot pitch monitor with an ATI improved VIP video card. 800x560 drivers for ACAD, Windows, GEM, and Ventura Publisher included. VGA, EGA, CGA, MDA, HGA selectable. Up to 256 colors on screen at one time with VIP card. 37 lb.

11½"x 14⅞" Greenbar Paper with Carbon Paper - 45 lb. $15

2 part 1500 sets, 4 part 700 sets. Two holes punched for binders. Case approximately 45 pounds. Send self-addressed stamped envelope for free sample.

SUPERIOR XT* STYLE KEYBOARD for the TOUCH TYPIST - $25

Replacement XT* style keyboard with LED in Num Lock and Cap Lock keys. Manufactured by **CHERRY**. Complete with coil cord and plug ready to slip into your case. Our service department uses these when possible instead of repairing the old one as most people appreciate the nice feel of these **made in USA** CHERRY keyboards. 2 lb.

MONOCHROME DISPLAY ADAPTER with PARALLEL PORT - $35

Japanese mfg. board. This is the MDA that can be plugged in with a CGA (Color Graphics Adapter) at the same time. 2 lb.

384K EPANSION WITH IO and RAM - $139

Parallel, serial, game, clock calendar, and 384K of memory installed and tested. **$139 3 lb.**

NO SCREWDRIVER REQUIRED PARALLEL PRINTER CABLE - $10 - $16 - $20

Not the cheap ones but the DB25 with the knurled fastener. **6 foot** $10 one lb., **10 foot** $16 two lb., **15 foot** $20 3 lb.

The COMPLETE DBase* SYSTEM for the SMALL BUSINESS - $4

DBase* command code and operational instructions are contained in this excellent 335 page 8½"x 11" book DBase APPLICATIONS in BUSINESS 2 lb.

SURGE SUPPRESSOR 6 OUTLET POWER STRIP with EMI/RFI FILTER - $15

Full 3 line protection in normal and common modes. Noise filter helps protect against noise interference. "Push to reset" circuit breaker protects against overloads. 6 foot 14/3 SJT power cord. On/Off switch. Continuous surge protection for your valuable electronic equipment. UL listed. 3 lb.

***TERMS** - Cashiers check, immediate shipment - All others must clear. No COD. Prices FOB Torrance, CA. Call us and give your ZIP code and we will quote freight charges. California residents add 6½% sales tax. Add $2 handling to orders under $25.*

KOALA COMPUTER INC.

4306 Torrance Boulevard, Torrance, California 90503 (Offices only)
CALL 213-316-5866 9 to 6 PST M-SAT

*XT trademark IBM, DOS - Microsoft, DBase - Ashton Tate

Reader Service Number 88

ON YOUR OWN

TMI Inc., A Business In Transition

Gary L. Scott
Decision Technology
P.O. Box 5040
Aloha, OR 97006

Over the last few days Gary and I have been discussing this column. I like details. You know, real numbers so you can compare your advertising results with those of the company mentioned. However, Gary pointed out, the company is very concerned about giving away information to competitors.

Gary also noted that marketing companies can be very profitable ways to earn a living. They can be more profitable (but riskier) than consulting practices.

If, like me, you're a voracious reader of technical magazines, you've probably seen a TMI ad. TMI has been advertising its Weather Pro in magazines such as *BYTE*, *Scientific American*, etc., for over eighteen months.

"Imagine having a sophisticated weather station hidden inside your PC."

As an habitual tinkerer, I've wanted a weather station since I was making anemometers out of Dixie cups. TMI's weather station is everything I've dreamed of. With such equipment I could ignore the airhead reading the National Weather Service forecast off a teleprompter.

The Weather Pro

TMI's Weather Pro is a personal weather station. The basic package includes sensors for temperature, barometric pressure, wind speed and wind direction, and an interface card that plugs into any 8-bit PC slot. The Weather Pro software runs as a TSR so data logging doesn't interfere with normal computer operations.

TMI has sold the Weather Pro to a long distance communications company so they can remotely monitor weather conditions at their microwave repeaters. The customer is correlating weather conditions at the repeater stations with transmission brownouts. Another of TMI's customers, an auto maker, uses the Weather Pro to monitor the direction and speed of the wind at its power plants.

TMI is not typical of the businesses that I'll be spotlighting in this column. TMI's founder, Dean Westbrook, was one of the founders of UNIFY Corporation and thus has considerably more experience and resources than most of us. Even so we can learn a lot by looking at TMI.

Choosing Products

As a marketing firm, TMI's first order of business was to find products to market. So they developed guidelines for screening prospective products, coming up with three baseline criteria:

(1) There must be a product to demonstrate,

(2) There must be a quantifiable market for the product, and

(3) The developer must be willing to continue development of the current product and follow-on products.

This may seem like a simple list, but TMI spent over a year looking for products that met all three. They didn't find a single one.

The closest that TMI found became their current product, the Weather Pro, but even this product only met the first two.

With only one product in a small vertical market, TMI has limited growth. Limited growth means that the company may never generate enough cash flow to adequately fund future growth. To grow, any company needs two things, growth of existing markets and entrance to new markets. Both of these take capital.

Limited growth for a small one or two person company might be acceptable. But for a company the size of TMI, limited growth in a small market would be catastrophic. What should they do?

Growth Options

There are three obvious directions. Close the doors. Reduce staff and continue to market a single product. Or, develop new products. TMI has decided to continue to market their existing product, the Weather Pro, while developing a complimentary product in-house.

I apologize to you for being vague about TMI's future designs, but this column will be coming out very close to TMI's announcement of their new product line, and I was asked not to reveal their plans.

What TMI is planning to do is to move from a single vertical market to a product that will allow them to market successfully into multiple vertical markets. To do this, TMI is bringing out a computer-based product that they can market to several vertical markets.

Using their new product, TMI will be able to get into many different markets with only minor rework of their new product. The new product should complement their existing Weather Pro in their current market since it will have many more features and, of course, be more expensive.

Limited growth for a small one or two person company might be acceptable.

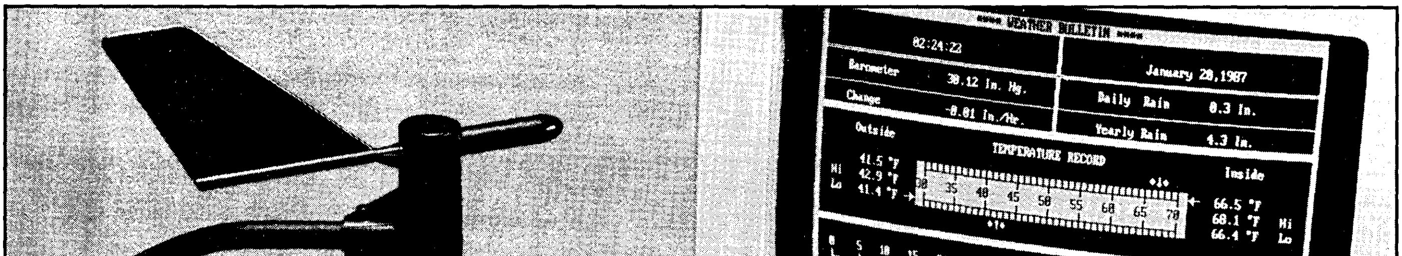
Simply by changing sensors, rewriting software, and rewriting manuals (no changes to the base hardware), TMI will be able to reach many new data acquisition markets.

However, they have to successfully market their product to these new audiences.

Their Focus: Marketing

Westbrook has kept overhead down by focusing on what the company does best, marketing.

TMI doesn't send out its own sales literature — this is done by the same



```
**** WEATHER BULLETIN ****
02:24:22                              January 20,1987
Barometer     30.12 In. Hg.           Daily Rain    0.3 In.
Change        -0.01 In./Hr.           Yearly Rain   4.3 In.
Outside            TEMPERATURE RECORD               Inside
   41.5 °F                                          66.5 °F
Hi 42.9 °F  30 35 40 45 50 55 60 65 70              68.1 °F  Hi
Lo 41.4 °F                                          66.4 °F  Lo
0 5 10 15 20 25 30    Wind Speed   0 MPH
                      Gust Speed   12 MPH
30 35 40 45 50 55 60  Wind Chill   41.5 °F
                      Gust Chill   29.5 °F
```

Display Generated by TMI Weather Station.

Figure 1 — Responses to Ads in a Large Computer Magazine.

			Days			
	Inquiries	Orders	<30	31-60	61-90	>90
Card Deck	500	25	5	9	2	9
'What's New' Mention	139	16	2	7	3	4

Figure 2 — Responses to Ads in a Weather-related Trade Magazine.

			Days			
	Inquiries	Orders	<30	31-60	61-90	>90
Ad #1	17	5	2	1	0	2
Ad #2	17	3	1	1	0	1
PR Article	14	9	3	2	0	4

***

company that handles this kind of distribution for Intel.

TMI doesn't manufacture its own products — building the products is handled by one of the local shops that has excess capacity.

Analyzing Advertising

TMI's sales support system is a two-part system, an advertisement aging portion and a more typical customer/inquiry tracking system. The advertisement tracking portion is unique to TMI.

How many companies know how effective each of their advertisements has been? One well-known high tech company in my area tracks bingo cards by the pound.

Lets look a little closer at the information TMI collects and how it can be used to determine the effectiveness of future advertisement.

Advertisement Response Aging: Every time an inquiry comes in, the person is asked where they heard about TMI's product. This information is then logged in a database and TMI can watch the responses to each advertisement.

Lets look at the responses to three different types of publicity. The responses came from a popular computer magazine such as *BYTE, PC Magazine,* or *PC World.* (The numbers have been changed a bit to protect TMI, but the relationship between the numbers has been maintained.) See Figure 1.

If we knew the number of copies distributed, we could calculate the percentage of readers who inquired and then compare that percentage with the percentages from other magazines. Or, we could use this type of data to compare current ads or quickly check the effectiveness of new ads and new magazines.

Let's the compare computer magazine response with the kinds of numbers produced by a weather-oriented trade magazine. See Figure 2.

The first thing that jumps out about Figure 2 is the much smaller response. Keep in mind though that most trade magazines have much smaller readerships. A closer look at both tables shows that a much higher percentage of inquiries from the trade magazine were turned into sales.

Editor's note: When you look at advertising costs, don't forget the costs of the inquiries. Someone has to answer the mail and the phone, someone has to send out those fancy ($) flyers, someone has to pay for calls on the 800 line, and someone has to pay for all those postage-due response cards.

It's not unusual to pay $2 (to $5 or more) to respond to each inquiry. I've heard stories of ads generating so many inquiries that the companies simply went bankrupt. They didn't have the capital left to make the products and complete the sales.

And, of course, this adds another item to the cost per sale. You'll need to add the inquiry costs to the advertising costs before dividing by the number of sales.

Obviously the response in the trade magazine is too small a sample to draw hard conclusions, but you can see some general trends in responses and sales.

And most often the effectiveness of an ad campaign is determined by a simple division, cost of ad/sale generated. What this simple number won't give you is a profile of the spending habits of your customers.

Marketing Costs Significant

In many small businesses, the cost of product development is insignificant in comparison to the costs incurred in marketing the product. Cost of sales goes on month after month, so determining the most effective methods of advertising is critical.

Is it safe to assume that an advertisement placed in a popular computer magazine will show a more immediate return while an ad placed in a vertical trade magazine will take longer to pay for itself? Will ads placed in trade magazines yield a larger average order? These are the kinds of questions that you can evaluate with a system like this.

Customer Support

When was the last time that you called a company for help with your new fungus-growing kit and got a support person with an IQ of a toadstool? Not that long ago, I'll bet.

As this column goes to press, TMI is looking for an electrical engineer to handle technical assistance. TMI's intent is to help their customers, not frustrate them.

Finally

TMI has laid a solid foundation to grow on. Only time will tell if TMI has chosen the right product, in the right market, at the right time. During the latter part of 1989, I'll do a follow-up look at how TMI has fared with its plans.

Anyone interested in more information on the Weather Pro product can call TMI at (800) 247-5712.

Next issue the column will cover the trials and tribulations of consulting. I would like to thank the people that have called to discuss consulting with me since my initial request for information. Remember this is your column. If you would like your company featured or have an idea for a column, please call or write.

♦ ♦ ♦

GEMS SUPER DISCOUNT* PRICES

MOTHERBOARDS

XT/8088

4.77/8mhz TURBO, 640K,OK	$69
4.77/10mhz TURBO, 640K,OK	76
4.77/10mhz TURBO, 1mb, OK, S/W	79
4.77/15mhz SUPER TURBO, w/1mb MEMORY, 2/3 size, 110% faster than 6mhz IBM AT	420

AT/286 AT SIZE

6/10mhz, 1mb OK, DTK, Ows	229
6/10mhz, 1mb OK, PC CALC, Ows	259
6/12mhz, 1mb OK, PC CALC, ows	289
6/12mhz, 1mb OK, EW, 12mhz CPU, Ows	289
6/16mhz, 1mb OK, PC CALC, Ows	399

AT/286 XT SIZE

6/10mhz, 1mb OK, DTK, Ows	239
6/10mhz, 1mb OK, ELTECH, Ows	229
6/12mhz, 1 or 4mb, OK, MS, Ows	265
6/16mhz, 2 or 8mb, OK, NOVAS	489
6/20mhz, 2 or 8mb, OK, NOVAS	549

AT/386 AT SIZE

16mhz, w / 1mb MICRONICS	1399
20mhz, w / 1mb, MICRONICS	1449
20mhz, w / 1mb, 287 + 387 SOCKET,MS.	1415
20mhz, w / 1mb, up to 16mb on M/B,1 par, 1 ser w / 2nd opt	CALL
20mhz MYLEX motherboard	CALL
25mhz 38C w/1mb	CALL

HARD DRIVES

ST-225 Seagate (DRIVE ONLY)	$215
ST-225 with CONTROLLER	255
ST-238 Seagate (DRIVE ONLY)	229
ST-238 with CONTROLLER	275
ST-251 Seagate 40ms HARD DRIVE	339
ST-251-1 Seagate 28ms HARD DRIVE	419
MR535 Mitsubishi RLL 65mb 28ms	465
ST-125 Seagate 3-½ format 20mb	245
Miniscribe 3650 40mb (61 ms) MFM	315
Miniscribe 3675 63mb (61 ms) RLL	335
MAXTOR....Call For Special Prices	**$CALL**
TOSHIBA MK130 65mb RLL 3-½ form	575
HARD CARD 20mb Plus Development	545
HARD CARD 40mb Plus Development	729

FLOPPY DISK DRIVES

Fujitsu 360K BLACK FACE PLATE	$64
Fujitsu 1.2mb BEIGE FACE PLATE	83
Teac 360K	73
Teac 1.2mb	86
Teac 720K with 5-¼ mount bracket	92
Teac 1.44 3-½ w/ 5-¼ bracket	110
Mitsumi 360K BLACK	63
Sony 720K 3-½ w/ 5-¼ bracket	94
Sony 1.44mb 3-½ w/ 5-¼ bracket	119
Toshiba 720K 3-½ w/ 5-¼ bracket	99
Toshiba 1.44mb 3-½ w/ 5-¼ bracket	119
Mitsubishi 360K	69
Mitsubishi 1.2mb	84
Mitsubishi 720K 3.5	79
Mitsubishi 1.44mb 3.5	95

CASE

XT SLIDE	$27
XT/AT Like Case	30
AT 3 Floppy Drive Front / 2 Button	54
AT 3 Floppy Drive with Digital Disp	72
XT SLIDE Heavy Duty, quality	27
XT/AT like, slide, heavy duty, qual	30
AT DT 3DR FT, 2 button, SOL	48
AT DT 3DR FT, 2 button, AS	54
AT DT 3DR FT, 2 button, DIG DISPLAY	72
AT DT 3DR FT, DIG DISPLAY/PC CALC	75

SPECIAL CASES

Transportable (case PS KB Mono Monit)	$435
Transportable EGA VER	1250
LCD portable 640X400 Bracklite 18lbs	759
Carrying Case for portables	35
Tower Case 230W PWR SPLY, 6 half ht	250
Tower Case w/digital display, 3 Dr. Front will hold 7, ½ht Dr. 220W	225

POWER SUPPLY

150 watt XT Compatible	$36
200 watt XT Compatible	49
200 watt AT Compatible	59
220 watt AT Compatible	63
250 watt AT Compatible	69

KEYBOARDS

84 KEY At Style Keyboard	$36
84 Key At Style Maxi Switch Keyboard	49
101 Key Enhanced Monterey	42
101 Key Tronics Keyboard	47
101 Maxiswitch Enhanced Keyboard	64
101 Tactile Enhanced Keyboard	49

MOUSE

Logitech M-8 Button	$35
Logitech C-7 Button	69
Logitech HI-REZ	86
Microsoft Bus Mouse + Paintbrush/Menu	99
Microsoft Ser Mouse + Paintbrush/Menu	99

VIDEO CARDS

Monochrome W/ Printer Port	$39
Monochrome 132 Column & Print Port	53
Color Graphics Adaptor	39
Color Graphics Adaptor 1/ print port	53
Micro EGA Auto Switching	129
Video-7 VGA	269
Paradise VGA Plus	259
Paradise VGA Professional	389
Chips and Technologies VGA Card	249

MODEM

1200 BAUD with Software	$52
2400 BAUD with Software	95
Zoom HC 2400 w/Procomm Software	116
External 1200 w/PC Talk III S/W	62
External 2400 w/PC Talk III S/W	109
Everex External 2400 Modem	179

I/O CARDS

Multi I/O Ser Par Cal Clk Game Disk	$48
Multi I/O Above (DFI)	66
Magic I/O Ser Par Cal Clk Disk Ctlr 360K, 720K, 1.2mb	49
I/O Plus.. Ser Par Cal Clk Game	42
Parallel Card (XT)	15
Serial Card 2nd Serial Optional	18
Port Serial Card w/2 Port & Opt	66
2nd Serial for XT	17
Game Card (2 Ports)	17
AT I/O CARD par, game, ser, 2nd ser opt	42
2nd Serial Port for AT	19

Floppy/Hard Disk Controller

2 Drive Floppy CTLR (360K/720K)	$17
4 Drive Floppy CTLR (360K/720K)	34
2 Drive Floppy Master (360K/1.2mb)	39
4 Drive FDC (360, 1.2/720, 1.44)SEF	89
4 Drive Super Special FDC W/Cables	89
WD FOX 2 Floppy Drive CTLR For AT	57
WD WX 1 Hard Disk Controller 8 Bit	57
WD XT/GEN Hard Disk CTLR For XT	52
WD 27X RLL Hard Disk Controller	59
WD WA-2 FD/HD CTLR For AT (MFM)	112
DTC 5287 FD/HD CTLR For AT (RLL)	154
NCL 5425 FD/HD CTLR For AT (MFM)	112
ADAPTEC 2372 FD/HD CTLR AT (RLL)	179
16 Bit 4 Floppy/2 Hard Drive Controller Modified 16 Bit DTC CTLR MFM & RLL	295

TAPE BACKUP

Teac 60mb MT2ST/45 (CASS) INT	$539
Everex 60mb Wangtek INT (Cart)	679
Colorado 44mb (Cart) Internal	289

MONITORS

Goldstar Amber Monochrome 720X348	$58
Samsung 1252 Mono 720X348 TIL/SW	66
Samsung 1464 EGB	219
Evervision 14 Amber Flat Screen	119
Amdek 410A	139
Nec GS	199
Taxan Composite Amber/Green	89
Relisys EGA with TILT & SWIVEL	339
Relisys Multi Scan	499
Mitsubishi 1410XC EGA	379
Mitsubishi 1381A Diamond Scan	539
Nec Multiscan II	569
Sigma Designs Laserview 1901-PC	1749

CAD PRODUCTS

MITSUBISHI HA3905 19V/20" MONITOR, ANALOG/TTL, 1024 X 1024, 15.7 35.5khz	$1725
MITSUBISHI HL6905 19" MONITOR, 1280 X 1024, 30⅞ 64khz AUTO TRACKING	2415

HIGH RESOLUTION VIDEO CARDS FOR CAD

QDP VIVA 1280 1290 X 768 16 COLORS	$1289
QDP VIVA 2000 1024 X 1024 16 COLORS	1559
QDP VIVA 2000 2 2024 X 2024 16 COLORS	1895

CHIPS (Prices Subject To Change)

V-20 8mhz (replacement for 8088)	$10
V-20 10mhz	16
8088-2 CPU	5
8087-2 MATH CO-PROCESSOR FOR XT	149
8087-1 MATH CO-PROCESSOR FOR XT	215
80287-8 MATH CO-PROCESSOR FOR AT	229
80287-10 MATH CO-PROCESSOR FOR AT.	279
80387-16 MATH CO-PROCESSOR FOR AT..	425
80387-20 MATH CO-PROCESSOR FOR AT.	CALL
64K 150ns DRAM	1.35
64K 120ns DRAM	3.25
64K 100ns DRAM	4.25
64K 120ns DRAM 4464	13.50
256K 150ns	10
256K 120ns	11.75
256K 100ns	11.95
256K 80ns	12.95
1mb 100ns	33
256K 100ns SIMM'S (SIP's)	125
256K 80ns SIMM'S (SIP's)	145
256K 100ns STATIC COLUMN RAM	17
256K 80ns STATIC COLUMN RAM	18

Desktop Publishing

Hi-Res 1024 15"MONITOR W/Video Controller 1024 X 768 Resolution	$499

SCANNERS

Mitsubishi Hand Scanner Full Page MH216 200 DPI, 8.5" X 11"	$638
Mitsubishi 400 DPI MH 130	CALL
Mitsubishi Paper Feeder For Scanner	176
Abaton 300 FB Full Page Scanner	1349

ACCELARATOR CARDS

MICRO 286-10 OK	$269
SOTA 286i 10mhz OK	298
SOTA 286i 12mhz OK	359

MOTHERCARDS

SOTA 5.0 MOTHERCARDS w/1mb, 10mhz.	$779
SOTA 5.0 MOTHERCARDS w/1mb, 12mhz.	879

Econo XT® Compatible

8mhz Turbo M/B,OK, case,
Keyboard, 150w P/S, FLOP CTLR, 1 360K
Drive, Mono video B.D.
MONITOR **$339**

SPECIAL

Lap Top Computer

The New Mitsubishi MP286L
12mhz,1.44,20mb.....**$2475**

Call for Discounted System Prices

***All prices shown are pre-paid or ordered by VISA or Mastercard. Charge card orders are subject to a 3% surcharge. For C.O.D. or term orders on parts above add 10%.**

XT® AT® & IBM® Are Registered Trademarks of International Business Machines.

GEMS
computers

1-800-332-GEMS

3446 De La Cruz Boulevard
Santa Clara, California 95054

In Business Since 1985
A member of the Better Business Bureau and Chamber of Commerce.

HOURS: M-F: 9 A.M. - 9:00 P.M. EST
7:00 A.M. - 7 P.M. PST / SAT: 8 A.M. - 6:00 P.M. PST

IN CA 408-988-0161 TECH SUPPORT 408-988-0146
FAX 408-988-0609

Shipping: 4% plus $3.00 handling on all part orders (except cases, 9% + $3.00). APO/FPO orders add 8% plus $3.00 on part orders. Call for exact charges on Systems and Monitors. CA residents add 7% tax. Prices reflect 3% cash discount.

One Year Warranty On All Parts And Systems / All Orders Are F.O.B. Santa Clara.

Reader Service Number 130

Around the Bend

Continued from page 4

as I wouldn't go back to pasting up *Micro C's* pages by hand.

Hard Drives, Yet Again

Talking about hard drives is a fast way to get attention; and with this column and Last Page running neck and neck for "most read" (and with Units and Modules and Bruce's hardware pieces coming up fast), it looks like I'll have to do another bit about the little beasties.

Actually, I'm just interested in hard drives. I use two or three every day, and there's no way *Micro C* would function as a floppy based magazine.

Anyway, I did a short phone survey to see if anything had changed radically in the past six months. Here's what I found:

Leo Henderson, Rotating Memory Systems (Service Shop)

"Don't use Seagates, any models. I'd never suggest a Seagate under any circumstances, they were designed to throw away. CMIs, the older ones aren't good. Priam, a little shaky, their best drives are medium quality."

So what would he recommend?

"Miniscribes aren't coming in for repair very quickly, they have a pretty good drive. CDC is also reliable. Maxtors are good. In fact, the older Maxtors are better than anything else in the low capacity field. Conner Peripherals is a new company building drives, they don't have a track record yet but their hardware looks very good."

Leo was the only person who wouldn't give passing consideration to the Seagate 80 meg 8096. When I asked him specifically about this popular full-height unit, he replied:

"Don't use Seagates under any circumstances."

Leo Henderson
Rotating Memory Systems
473 Sapena Ct. #26
Santa Clara, CA 95054
(408) 988-2334

Dale, Cal Abco (Distributor)

"In the 20 meg arena, we're having trouble with the 3 1/2" Seagates. Those are the ones Seagate's hoping will replace the 5 1/4" 225s. Out of 100 125s we sell, 6 to 9 come right back. Things are too tight in the drive market to pay for that many returns. No one's making anything on drives as it is. I'm recommending the Miniscribe 8425F, 20 meg, 40 ms, MFM, half height.

"For 30 megs I'm suggesting that people get the 8438 RLL 68 ms Miniscribe or the 8438F (fast) RLL 40 ms.

"For 40 megs I like the Miniscribe 3053 MFM and the Seagate 251-1 (fast) MFM.

"In the 80 meg arena, I especially like the Miniscribe 6085 full-height 71 meg. Or if you need an inexpensive drive with a full 80 megs, use the full-height Seagate 4096.

"However, the best drive in that ballpark is the Micropolis, 79 meg. You could throw it off the Empire State Building. It's the Cadillac of the high end drives, but you'll pay $300 to $400 more than you would for a 4096. For 150 meg and above, the

Around the Bend

Continued from page 70

drive you want is definitely Micropolis."

Dale Leisen
Cal Abco
6041 Variel Ave.
Woodland Hills, CA 91367
(800) 321-4484 Ext. #295

Brian Garrison, Emerald Microware (Retailer)

"I'm still selling Miniscribes. For 20 megs I'm selling the 8438, MFM, 3 1/2" either 68 ms or 40 ms.

"For 30 megs it's the 8438 with an RLL controller. I've had no trouble with that combination.

"For 40 megs I use the 3650s, half height. Not real fast, 60 ms, but they're very cheap. Or, if someone wants speed I give them the 3053 MFM. At 25 ms it's hot, though it's relatively new.

"For 80 megs I like the 3085. It's 22 ms, half height, 5 1/4" and holds 71 meg.

"For controller cards the Western Digital 1002 WX1s are good. For RLL I'm using DTC and Adaptec. The Adaptec is very fast at 1:1 interleave.

"I've noticed that a lot of the house-brand OEM clones are using Tandon, half height, 3 1/2" hard drives. They must be awfully cheap.

"I'm not handling any Seagates now, at all. Seagate called last week and told me all about their reliability, but I don't know anyone who's using them."

Brian Garrison
Emerald Microware
P.O. Box 1726
Beaverton, OR 97075
(503) 641-0347

A Little Reality

I must add something. If you have Seagates, don't panic. Some of our 225s have run flawlessly for years. And, if you purchase something else, remember that they too ship an occasional lemon.

If your drives have been error free for awhile, they'll probably be error free for a lot longer. Be sure to do a low level reformat (especially on the 225s and 238s) every six months (wouldn't hurt with any drive).

Be sure to keep them as cool as possible. Don't stack a floppy drive right on top of a hard drive, especially a Seagate. After running the system for two hours, open it up enough to place your hand firmly on top of the unit. If it's uncomfortably hot, then move your hand. (I shouldn't have to tell you this, you know.) Also move the drive.

Come to think of it, I should tell you about that big wad of cables you have blocking air flow across your main board. You know, those grey ribbons that hook up with the drives and ports and all. Kinda zaps those fancy ceramic LSI chips, being hot and all. (Remind me to mention it in this column sometime.)

Around the Bend

Continued from page 71

Tape Backup

Speaking of hard drives, I kind of took to heart the stuff I wrote in the July/August issue (very unlike me) and decided to back up the Micro C hard drives.

Tape always seemed like the easiest method. Stick in the cartridge, start the backup program and then split.

Everyone's making tape drives that look vaguely familiar to floppy controllers, so you just plug them in. My experience with an Alloy 40-meg unit demonstrated that there were ways to screw things up.

I fought with that tape unit for six months. Returned it and waited another six months to get my money back.

So it was with great trepidation that I purchased my second 40-meg tape backup and headed back to the office. (Between MicroSphere and Micro C there's the Desert Deli, and they have these large, warm, crispy, gooey, chocolate chip cookies. I bought a dozen to fortify me for the battle.)

However, in under half a cookie I'd pulled the connectors off the B: floppy and slipped them onto the tape unit, inserted the support disk, and fired up the system.

It worked.

But that left me holding the bag (of cookies), so I moved the tape drive to another system. An AT.

It worked. (In another half a cookie.)

Out of desperation I attacked one of our PC Tech X-16s. They didn't use the 765 floppy controller in the X-16, so most backup software doesn't work. (This ought to be a half-bag project.) But in one cookie flat, the Archive was zipping along. Perfectly. (And by the time I'd finished the third of those wonderful zit producers, I was an expert with the tape backup's software.)

The tape drive I purchased was the Archive 5240. It appears to run at 250K bit/sec (like a 360K drive) and stores an average of 1 megabyte per minute. (Works on XTs and ATs.)

They have another model, a 5540 designed for ATs only. It stores 2 megs/minute, so I assume it transfers data at 500K bit/sec like the 1.2 and 1.44 meg drives.

Both retail for $499. (Without chocolate chips.)

Since we have more machines than tape backups, I sometimes move one of the drives to other machines.

In fact, it's so easy and the backup is so effortless, I've got a suggestion:

If you service hard drives, you're often taking data off a drive, reformatting it, and then restoring the data. Ten floppies here, fifteen there. You spend hours feeding disks into machines (dum-da-dum-dum). Why not cable up one of these little tape units and go have lunch, dinner, even a dozen warm, crispy, chocolate chip cookies? (You can get them to go.)

Archive Corporation
1650 Sunflower Ave.
Costa Mesa, CA 92626
(714) 641-1230

Reader Service Number 59

Reader Service Number 37

Around the Bend

Continued from page 72

Work Time

From reading *Micro C*, you probably assume we goof off all the time. You figure I'm up in the air somewhere in the Stinson, Larry's on vacation, and the rest of the staff is holding an eternal SOG. Not so. Why this very moment I'm sitting in front of my computer slaving away on the world's second longest editorial (the longest ran last issue).

I also happen to be sitting on a log, in the middle (not near an edge, anyway) of some very thickly wooded mountains.

I'm in a very narrow cut in the hills, a small creek tumbles by about 20 yards below me; occasionally I hear a bird or two over the rush of the water.

It's a lot of work putting out a magazine. A lot of work.

Micro Channel Is Dead, Long Live...

When I first saw IBM's micro channel, I was bummed. Here we go, follow the leader again, only this time IBM had raised the stakes and it would get a cut from every one of those stakes. It looked like they'd found a way to turn the micro marketplace into one gigantic profit center.

However, something happened to micro channel. I think it was the marketplace.

What is IBM left with?

(1) A PS/2 series that it says is five times as dependable as the original ATs. (Though I understand the PS/2s still aren't as dependable as generic clones.)

(2) Machines that can't keep up with many clones (despite the "high-speed" bus).

(3) A motherboard that won't accept 90% of the video, drive, I/O, and other cards currently on the market.

(4) A system that people aren't writing custom software for.

(5) A system that's almost as compatible with the original XT/AT software (the kind of software people are writing) as the clones.

Meanwhile:

Why do math coprocessors cost $150 - $450? Why has Intel refused to second source the 80386? Why am I asking these dumb questions?

Intel, it appears, is trying to take full advantage of its chip design prowess and its corner on the market. However, VM Technologies (a Japanese consortium) is laying out $1 billion (probably in the form of yen) to ramp up manufacturing of its VM8600S. The chip will run 386, 286, and 8086 code at 20 MHz and it's supposed to run 8086 code faster at 20 MHz than an 80386 at 20 MHz. Plus, if I know the Japanese, the chip won't cost $300 a copy.

Of course, Intel could push for an import ban so the Japanese wouldn't dump VM8600s on the U.S. market.

Magazine Review

Borland's *Turbo Technix* and Kaypro's *Profiles* have called it quits.

Turbo Technix had been given a year to make a profit. *Profiles* got five years.

But magazines aren't the only things the companies have dumped. Borland's taken a new, very conservative tack, reportedly cutting advertising from $6 million/year to $2.2 million (you don't see them in *Micro C* any longer) and hacking away at their direct sales staff.

Kaypro appears to be belt-tightening everywhere. Not only are they cutting out frills such as the magazine, but they're also cutting management (I understand David Kay resigned last Friday). I've heard that the problem stems from ongoing losses and a $12 million note that comes due the middle of October. Which leaves the chilling question: Does Kaypro Corp. have a ghost of a chance of making it to Halloween?

Micro/Systems Journal RIPs Again

Those of you who are old enough to remember S100 will no doubt remember Sol Libes and *Microsystems. Microsystems* was purchased from Sol by Ziff-Davis and Ziff ran it for a year or two before folding it.

Sol restarted the magazine as *Micro/Systems Journal* and two years ago sold it to M&T (the company which publishes *Dr. Dobb's*). M&T planned to strengthen the hardware/software mix, move strongly into the PC market, and go head to head with *BYTE*.

Well, M&T's contract with Sol has expired, so it is shutting down *Micro/Systems Journal* and is starting a whole new magazine called *LAN Technology*.

Sol may be tired enough of magazines to let dead journals lie.

Meanwhile, I understand *BYTE* has dropped Ciarcia (as of the December '88 issue). *BYTE* is trying to become yet another *PC Magazine*. (But you knew that.)

Around the Bend

Continued from page 72

So at *Micro C*, we just get ourselves out of a puddle and into the mainstream and what happens? Everyone else bails out. Is there something about hardware that we don't know? (Is there something about hardware that they don't know?)

By the way, speaking of puddles, I've confluenced the Kaypro Column with CP/M Notes.

The Editor's Shorts:

Stat Graphics

Tom Gettys called to say there's a bug in Stat Graphics. The program ignores more than six columns of data unless you set your system clock to April 1, 1988, or earlier.

Kinda makes me wonder who wrote their program. He may no longer be there, but his legacy definitely lives on.

Perfect Vision

I was just loaned a board by Perfect Vision which converts composite video (TV, video camera stuff) into computer graphics. This board grabs pictures from any video tape recorder, or video camera, and displays them. Then it lets you store the images on disk and display them later. More importantly, it also translates images into PCX files for use in graphics and desktop packages. (Hey, how about a spread of Olympics pictures — the first *Micro C* centerfold?)

The current model only stores shades of grey, but it does let you translate each of those shades into an incredible number of colors. Boy, the posterizations are interesting.

However, there's no way I know of to grab the colors other than taking a picture of the screen. When I import the monochrome versions of these files into Ventura, for instance, the image is very poor quality. Garbage and jitter that I didn't notice on the TV screen look awful when captured.

But even what appeared to be reasonable quality when captured, was, at best, marginal as a PCX file. At this point I think the technology is still relatively crude, probably hampered more by the low resolution of television transmissions than an inability to convert images.

Also, this new product has its share of rough edges. Images weren't always translated properly into PCX files. Image contrast and brightness were difficult to set. Error messages were cryptic, and the way it operated wasn't always the way I expected — though it was generally easy to use.

Perfect Vision $349.95
SunRise Industries
3801 Old College Rd.
Bryan, TX 77801
(409) 846-1311

12 Volt Book

Okay, okay. I give up. I assumed that an ISBN number would be sufficient to connect you up with this informative book. I was wrong. So here's all the dope on this load of information on batteries, generators, and refrigerators. (Refrigerators? Sure, to store your Jolt Cola so you can program.)

Anyway, David Smead and Ruth Ishihara put together a sailboat that's entirely electrically self-sufficient, and this book is about that experience.

Living On 12 Volts (344 pgs, $25)
Rides Publishing Company
2442 NW Market St. #43
Seattle, WA 98107
(206) 789-5758

Call them and they'll sail it to you.

Advertise In Russia

I read a short piece in the *Wall Street Journal* that I have to pass along. It turns out that the Russians (with the help of *Computerworld's* publisher) have started a magazine called *PC World USSR*.

This wonderful new Russian magazine has an innovative new feature — advertising. (You heard it here second.)

Anyway, their U.S.-based salesman sold 30 pages of ads at $3,000 per page, but when the first issue came out it contained only four ads. It turns out that the Soviet editors removed any ads that "didn't contain enough technical information" and decided the facts in 11 others repeated information already in editorial copy. Apparently they didn't want to waste the advertisers' money.

Waste? We're talking advertising, not ethics.

Do Svidán'ya (It's goodby in Russian, I hope.)
David Thompson

♦ ♦ ♦

UNITS & MODULES

Out Of Sorts?

And: Can You Spell Me?

By Michael S. Hunt
845 E. Wyeth
Pocatello, ID 83201
(208) 233-7539

Michael writes the fundamental units you need to create and use a real, honest to Webster, dictionary. Great fun.

Last issue I showed you a program and asked what it does. The identifiers were changed to protect the identity of the program. If you say it's a sort routine, then you get your Junior Programmer Badge. This program (see Figure 1) is called The Sort From Hell. Now let's see if I can explain how it works.

The program first calls SortInput. SortInput calls Number with the prev parameter set to MAXINT. This insures that no input value is greater than the prev value for the initial call to Number. On the first call Bubble has no effect, and Number makes a recursive call to itself. Each time Number is invoked it requests an integer n, swaps n with the previous n if the current n is larger by calling Bubble, and then invokes itself.

When Bubble is called, it performs the swap if necessary. If the swap was necessary, Bubble also calls PrevBubble. PrevBubble is Bubble from the previous recursive call. The first time Number is called, PrevBubble is the empty procedure EndBubble. EndBubble is used because no swap is required for the first call to Number. The combination of Bubble and PrevBubble keeps the list of integers in ascending order.

When eoln(input) becomes true, the integers are printed out in descending order as Number falls out of recursion.

I hope my description is clear, but it's hard to describe something this convoluted and recursive. If you have a language that supports procedure passing, try running it.

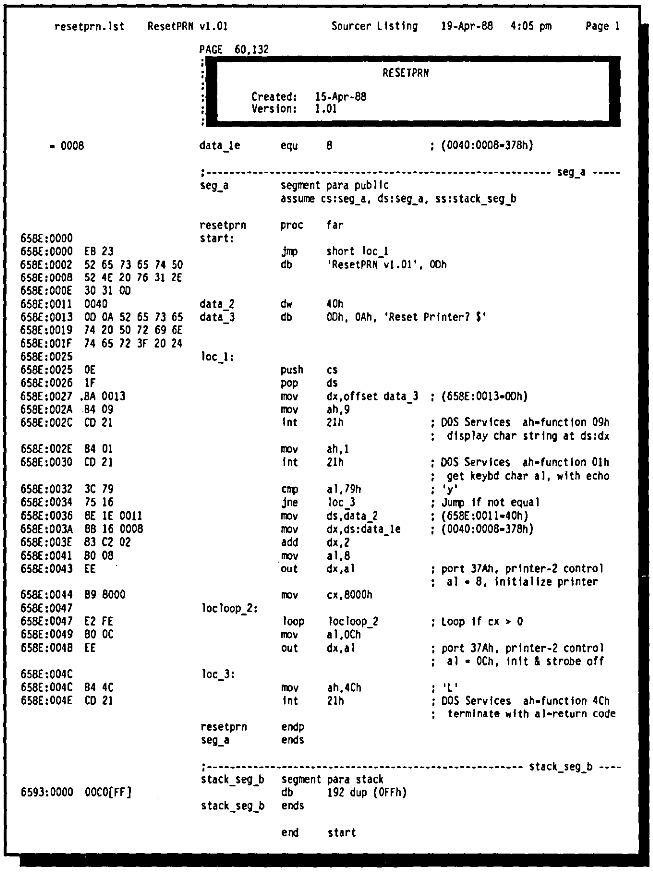
This Time

Last time I introduced the problem of storing large amounts of information. In this issue I will describe a dictionary unit and how it works. The example unit (see Figure 2) uses a disk based dictionary. A memory based unit and the Modula-2 code are available on the Issue #45 disk.

You might think that a spelling dictionary is only useful for writers, but a dictionary can store and retrieve many types of information. Any list of items, such as identifiers, passwords, or anything else, can be stored in the dictionary for reference.

Figure 1 — WhatAmI From Last Issue

```
Program Sort(input,output);
(* David Wall, IFRICS July 87 *)

procedure Number(var prev: integer;
                 procedure PrevBubble);
var n : integer;

  procedure Bubble;
  var temp : integer;
  begin {Bubble}
    if n > prev then
      begin
        temp := n;
        n := prev;
        prev := temp;
        PrevBubble;
      end; {if}
    end; {Bubble}

begin {Number}
  if not eoln(input) then
    begin
      read(n);
      Bubble;
      Number(n,Bubble);
      write(' ',n:1);
    end; {if}
end; {Number}

procedure SortInput;
var endNumber:integer;

  procedure EndBubble;
  begin
  end;

begin {sortInput}
  endNumber := MAXINT;
  Number(endNumber,EndBubble);
  writeln
end; {SortInput}

begin
  SortInput
end.

***
```



```
resetprn.lst    ResetPRN v1.01              Sourcer Listing    19-Apr-88    4:05 pm      Page 1

                                  PAGE  60,132
                                  ;
                                  ;                          RESETPRN
                                  ;
                                  ;        Created:   15-Apr-88
                                  ;        Version:   1.01
                                  ;

 = 0008                           data_1e       equ     8                   ; (0040:0008=378h)

                                  ;------------------------------------------------------------ seg_a -----
                                  seg_a         segment para public
                                                assume cs:seg_a, ds:seg_a, ss:stack_seg_b

                                  resetprn      proc    far
658E:0000                         start:
658E:0000  EB 23                                jmp     short loc_1
658E:0002  52 65 73 65 74 50                    db      'ResetPRN v1.01', 0Dh
658E:0008  52 4E 20 76 31 2E
658E:000E  30 31 0D
658E:0011  0040                   data_2        dw      40h
658E:0013  0D 0A 52 65 73 65      data_3        db      0Dh, 0Ah, 'Reset Printer? $'
658E:0019  74 20 50 72 69 6E
658E:001F  74 65 72 3F 20 24
658E:0025                         loc_1:
658E:0025  0E                                   push    cs
658E:0026  1F                                   pop     ds
658E:0027 .BA 0013                              mov     dx,offset data_3  ; (658E:0013=0Dh)
658E:002A  B4 09                                mov     ah,9
658E:002C  CD 21                                int     21h               ; DOS Services  ah=function 09h
                                                                          ;  display char string at ds:dx
658E:002E  B4 01                                mov     ah,1
658E:0030  CD 21                                int     21h               ; DOS Services  ah=function 01h
                                                                          ;  get keybd char al, with echo
658E:0032  3C 79                                cmp     al,79h            ; 'y'
658E:0034  75 16                                jne     loc_3             ; Jump if not equal
658E:0036  8E 1E 0011                           mov     ds,data_2         ; (658E:0011=40h)
658E:003A  8B 16 0008                           mov     dx,ds:data_1e     ; (0040:0008=378h)
658E:003E  83 C2 02                             add     dx,2
658E:0041  B0 08                                mov     al,8
658E:0043  EE                                   out     dx,al             ; port 37Ah, printer-2 control
                                                                          ;  al = 8, initialize printer
658E:0044  B9 8000                              mov     cx,8000h
658E:0047                         locloop_2:
658E:0047  E2 FE                                loop    locloop_2         ; Loop if cx > 0
658E:0049  B0 0C                                mov     al,0Ch
658E:004B  EE                                   out     dx,al             ; port 37Ah, printer-2 control
                                                                          ;  al = 0Ch, init & strobe off
658E:004C                         loc_3:
658E:004C  B4 4C                                mov     ah,4Ch            ; 'L'
658E:004E  CD 21                                int     21h               ; DOS Services  ah=function 4Ch
                                                                          ;  terminate with al=return code
                                  resetprn      endp
                                  seg_a         ends

                                  ;------------------------------------------------------- stack_seg_b ----
                                  stack_seg_b   segment para stack
6593:0000  00C0[FF]                             db      192 dup (0FFh)
                                  stack_seg_b   ends

                                                end     start
```

Figure 3—Dictionary Tree Structure

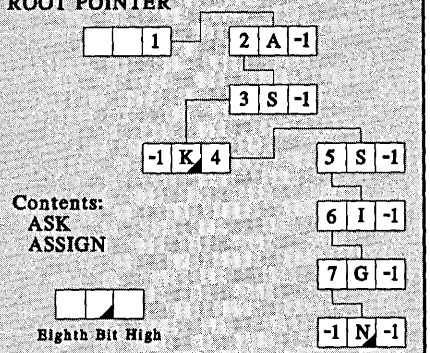


Figure 4—Pointer Overhead

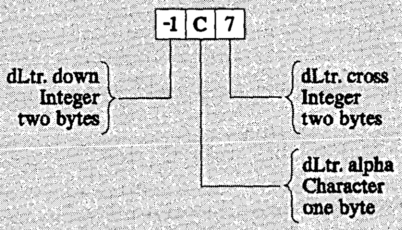


Figure 5—Words Within Words

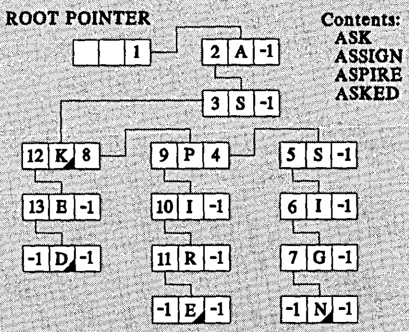


Figure 2 — Dictionary Code

```
unit Dict;

interface

type  str255 = string[255];
      fileName = string[79];

var dictOpen : BOOLEAN;

procedure OpenDict(fName : fileName);
procedure CloseDict;
function WordFound(wrd : str255) : BOOLEAN;
procedure AddWord(wrd : str255);
function WordDelete(wrd : str255) : BOOLEAN;
procedure BuildDict(wordFile : fileName);

implementation

const  recsize = 5;

type   dLtrPtr = integer;               (* ptr to disk record *)
       dLtr = record
         down : dLtrPtr;                (* ptr to nth char level *)
         alpha : char;                  (* char stored *)
         cross : dLtrPtr;               (* ptr to nth+1 char level *)
       end;

var  dictFile : filename;
     dFile : file;                      (* untyped file for dictionary *)

function FileExists(fName : fileName) : BOOLEAN;
var f : file;    (* from Turbo Pascal 4.0 manual page 473 *)
begin
  {$I-}
  Assign(f,fName);
  Reset(f);
  Close(f);
  {$I+}
  FileExists := (IOResult = 0) and (fName <> '');
end; (* FileExists *)

function UpCaseWrd(wrd : str255) : str255;
var I : integer;                        (* returns upper case word *)
begin
  for I := 1 to Length(wrd) do wrd[I] := UpCase(wrd[I]);
  UpCaseWrd := wrd
end; (* UpCaseWrd *)

function SevenBitCh(b : char) : char;  (* sets bit eight to zero *)
begin
  if ord(b) >= 128 then
    SevenBitCh := chr(ord(b)-128)
  else
    SevenBitCh := b
end; (* SevenBitCh *)

function EightBitCh(b : char) : char;  (* sets bit eight to one *)
begin
  if ord(b) < 128 then
    EightBitCh := chr(ord(b)+128)
  else
    EightBitCh  := b
end; (* EightBitCh *)

procedure ReadRec(var f : file;pos : dLtrPtr;var rec : dLtr);
begin
  Seek(f, pos);                         (* performs direct read *)
  BlockRead(f, rec, 1)                  (* on untyped file *)
end; (* ReadRec *)

procedure WriteRec(var f : file;pos : dLtrPtr;rec : dLtr);
begin
  Seek(f, pos);                         (* performs direct write *)
  BlockWrite(f, rec, 1)                 (* on untyped file *)
end; (* WriteRec *)

procedure OpenDict(fName : fileName);  (* opens dictionary file *)
begin
  Assign(dFile, fName);
  if FileExists(fName) then             (* if it exists *)
```

Continued on page 78

If the dictionary were stored as a sequential file, it would have to be searched from the beginning for each word. If the sequential file were sorted in ascending order, the search could terminate when it encountered a word greater than or equal to the search word. Searches for words towards the end of the list would take a long time.

I chose a tree structure for the dictionary because it gave me direct access to the information. Direct access is much like opening a (book) dictionary and quickly locating your word. Figure 3 illustrates a tree structure containing the words "ask" and "assign."

> If every character in the word list required a node, the dictionary would be five times the size of the word list. Fortunately you can save space by taking advantage of character repetitions.

Trees have overhead in the form of pointers to other nodes in the tree. The overhead for the dictionary tree is high, for each node there is one byte for the character and two bytes for each of the two pointers to other nodes in the tree (Figure 4). The ratio of node size to stored information is 5:1.

The record dLtr is the node type and has three components. The alpha variable contains the stored character. The cross variable points to another letter allowed at this nth position in the word. The down variable points to allowed letters at the nth+1 position in the word.

If every character in the word list required a node, the dictionary would be five times the size of the word list. Fortunately you can save space by taking advantage of character repetitions.

Since "ask" and "assign" share the

same first two characters, only seven characters have to be stored to represent the two words (Figure 3).

When one word encompasses another, like "asked" does with "ask" (Figure 5), it's not clear where the first word ends. All of the alphabet can be represented by only seven bits of a byte. The eighth bit is unused. When a word terminates in the dictionary the eighth bit is set high (set to 1) for the last character.

The dictionary unit contains procedures, functions and variables that:

1) Add a word.	AddWord
2) Delete a word.	WordDelete
3) Find a word.	WordFound
4) Build dictionary from a word list.	BuildDict
5) Denote open dictionary.	dictOpen
6) Open dictionary.	OpenDict
7) Close dictionary.	CloseDict

Building The Dictionary

The first task is to build the dictionary from a word list. Where would you find a list of several thousand unique words?

First I tried using the screen cut and paste from within Desqview, but that was too cumbersome for individual words. Then I remembered that the Word+ spelling checker for my CP/M system created a unique word list as an intermediate step when spell checking a file. I rummaged around through a couple of boxes of disks until I found a 100K+ document. In five minutes Word+ had created a list of about 2500 unique words.

Now to get those words stored into a dictionary. The basic step for building the dictionary is to add the words one at a time. Procedure BuildDict opens the text file passed in the parameter list, then reads in a word at a time and calls procedure AddWord for each word in the list. BuildDict expects the text file to have only one word on each line. It also expects a dictionary to be opened.

Adding Words

AddWord inserts new words into an existing dictionary. A word may exist in the dictionary that shares the first n characters with the word to be added. AddWord traces the path of repetitive characters.

At the position that the two words differ, AddWord will begin inserting new records into the dictionary. The word to be inserted may be contained

```
    begin
      if dictOpen then Close(dFile);
      Reset(dFile,recsize)             (* open it at record 0 *)
    end
  else
    begin
      Rewrite(dFile,recsize)           (* else create the file *)
    end;
  dictOpen := TRUE
end; (* OpenDict *)

procedure CloseDict;
begin
  if dictOpen then Close(dFile);       (* close dictionary file *)
  dictOpen := FALSE
end; (* CloseDict *)

function WordFound(wrd : str255) : BOOLEAN;   (* find wrd in dictionary *)
var curPtr : dLtrPtr;
    tLtr : dLtr;
    wrdIdx : integer;
begin
  if FileSize(dfile) = 0 then          (* if empty file, no word *)
    WordFound := FALSE
  else
  begin
    wrd := UpCaseWrd(wrd);
    ReadRec(dFile, 0, tLtr);           (* get root pointer *)
    curPtr := tLtr.cross;
    wrdIdx := 1;
    while (curPtr >= 0) AND (wrdIdx < Length(wrd)) do
      begin
        ReadRec(dFile, curPtr, tLtr);
        while (SevenBitCh(tLtr.alpha) < wrd[wrdIdx])
              AND (tLtr.cross >= 0) do
          begin
            curPtr := tLtr.cross;      (* transverse level n *)
            ReadRec(dFile, curPtr, tLtr)
          end;
        if (SevenBitCh(tLtr.alpha) = wrd[wrdIdx]) then
            curPtr := tLtr.down        (* move to nth + 1 level *)
        else
            curPtr := -1;
        Inc(wrdIdx)
      end;
    if (curPtr >= 0) then
      begin
        ReadRec(dFile, curPtr, tLtr);
        while (SevenBitCh(tLtr.alpha) < wrd[wrdIdx])
              AND (tLtr.cross >= 0) do
          begin
            curPtr := tLtr.cross;      (* drop to bottom of column *)
            ReadRec(dFile, curPtr, tLtr)
          end;
        if (EightBitCh(tLtr.alpha) = tLtr.alpha) then
          WordFound := TRUE
        else
          WordFound := FALSE
      end
    else
      WordFound := FALSE
  end
end; (* WordFound *)

procedure AddWord(wrd : str255);
var curPtr, prevPtr : dLtrPtr;
    tLtr : dLtr;
    wrdIdx, I : integer;

  function NewList(ch : char) : dLtrptr;
  var curLtr : dLtr;
  begin                                (* create a new cross list *)
    curLtr.down := -1;
    curLtr.cross := -1;
    curLtr.alpha := ch;
    WriteRec(dFile, FileSize(dfile), curLtr);
    NewList := FileSize(dFile)-1
  end; (* NewList *)
```

Continued on page 80

entirely by a word already in the dictionary. In this and all other cases, the eighth bit of the last character to be inserted is set to 1 at the end of AddWord. This is the flag for a valid end of word.

Finding Words

After the dictionary is built, a function is needed to tell us whether a word exists in the dictionary. Function WordFound returns a BOOLEAN value, and lets us use the existence of a word to control program flow.
Example:

```
if WordFound(wrd) then
  write(wrd,' found')
else
  write(wrd,' not found');
```

WordFound returns FALSE if the path it follows terminates before the end of the word or the eighth bit is not set to one for the last character. WordFound only returns TRUE if the path doesn't terminate and the eighth bit of the last character is set to one.

Deleting Words

WordDelete removes words from the dictionary. It sets the eighth bit of the last character to zero. With the eighth bit set to zero, WordFound can't detect the end of the word. It will return FALSE for all checks on that word.

Restrictions

The data structure I used for the dictionary is not optimum. The unit dict is designed as a general storage and retrieval system. You can, no doubt, find a better algorithm and data structure for word storage. For instance, digging into the relationship between the roots of words and their prefixes and suffixes might lead to a better method of storing words. A short list of common words (to check first) might also reduce search time.

The dictionary did not save storage space for the test data. The 25K word list became a 42K dictionary. The small list of words contained little repetition, so storage requirements increased. A large list of words with a greater number of character repetitions would yield a dictionary equal to or smaller than the word list.

The record pointers are two byte integers. You'll get an error if you use a record number higher than MAXINT. The dictionary tree can have up to 32767 nodes. This yields a maximum file size of 32767*5+5=163840 bytes. The

```
  function InsertLtr(prev, next : dLtrPtr;ch : char) : dLtrPtr;
  var curLtr, prevLtr : dLtr;
  begin                              (* insert char in a cross list *)
    ReadRec(dFile, prev, prevLtr);
    if (prevLtr.cross = next) then (*determine where to insert char*)
      prevLtr.cross := FileSize(dfile)
    else
      prevLtr.down := FileSize(dFile);
    WriteRec(dFile, prev, prevLtr);
    curLtr.cross := next;
    curLtr.down  := -1;
    curLtr.alpha := ch;
    InsertLtr := FileSize(dfile);
    WriteRec(dFile, FileSize(dfile), curLtr)
  end; (* InsertLtr *)

begin
  wrd := UpCaseWrd(wrd);
  wrdIdx := 1;
  if FileSize(dfile) = 0 then
    begin                            (* initialize a new dictionary *)
      prevPtr := NewList(' ');
      prevPtr := InsertLtr(0,-1,wrd[wrdIdx]);
      curPtr := -1;
      wrdIdx := 2
    end
  else
    begin
      ReadRec(dFile, 0, tLtr);         (* get root pointer *)
      prevPtr := 0;
      curPtr := tLtr.cross
    end;
  while (curPtr >= 0) AND (wrdIdx <= Length(wrd)) do
    begin
      ReadRec(dFile, curPtr, tLtr);
      while (SevenBitCh(tLtr.alpha) < wrd[wrdIdx]) AND
            (tLtr.cross >  0) do
        begin
          prevPtr := curPtr;           (* transverse level n *)
          curPtr := tLtr.cross;
          ReadRec(dFile, tLtr.cross, tLtr)
        end;
      if (SevenBitCh(tLtr.alpha) = wrd[wrdIdx]) then
        begin
          prevPtr := curPtr;           (* drop to nth + 1 level *)
          curPtr := tLtr.down
        end
      else
        begin
          if (SevenBitCh(tLtr.alpha) < wrd[wrdIdx]) then
            prevPtr := InsertLtr(curPtr,tLtr.cross,wrd[wrdIdx])
          else                (* start new column, vertical run *)
            prevPtr := InsertLtr(prevPtr,curPtr,wrd[wrdIdx]);
          curPtr := -1
        end;
      Inc(wrdIdx)
    end;
  for I := wrdIdx to Length(wrd) do
    begin
      curPtr := NewList(wrd[wrdIdx]);  (* create new column of *)
      ReadRec(dFile, prevPtr, tLtr);   (* matched characters *)
      tLtr.down := curPtr;
      WriteRec(dFile, prevPtr, tLtr);
      prevPtr := curPtr;
      Inc(wrdIdx)
    end;
  ReadRec(dFile, prevPtr, tLtr);
  tLtr.alpha := EightBitCh(tLtr.alpha); (* set bit 8 / end of word *)
  WriteRec(dFile, prevPtr, tLtr)
end; (* AddWord *)

function WordDelete(wrd : str255) : BOOLEAN;
var curPtr : dLtrPtr;
    tLtr : dLtr;
    wrdIdx : integer;
begin
  if FileSize(dfile) = 0 then          (* if empty file, quit *)
    WordDelete := FALSE
  else
    begin
      wrd := UpCaseWrd(wrd);
      ReadRec(dFile, 0, tLtr);         (* get root pointer *)
      curPtr := tLtr.cross;
      wrdIdx := 1;
      while (curPtr >= 0) AND (wrdIdx < Length(wrd)) do
        begin
          ReadRec(dFile, curPtr, tLtr);
          while (SevenBitCh(tLtr.alpha) < wrd[wrdIdx])
                AND (tLtr.cross >= 0) do
            begin
              curPtr := tLtr.cross;    (* transverse level n *)
              ReadRec(dFile, curPtr, tLtr)
            end;
          if (SevenBitCh(tLtr.alpha) = wrd[wrdIdx]) then
            curPtr := tLtr.down        (* drop to nth + 1 level *)
          else
            curPtr := -1;
          Inc(wrdIdx)
        end;
      if (curPtr >= 0) then
        begin
          ReadRec(dFile, curPtr, tLtr);
          while (SevenBitCh(tLtr.alpha) < wrd[wrdIdx])
                AND (tLtr.cross >= 0) do
            begin                      (* transverse last level *)
              curPtr := tLtr.cross;
              ReadRec(dFile, curPtr, tLtr)
            end;
          if (EightBitCh(tLtr.alpha) = tLtr.alpha) then
            begin                      (* remove eighth bit *)
              tLtr.alpha := SevenBitCh(tLtr.alpha);
              WriteRec(dFile, curPtr, tLtr);
              WordDelete := TRUE
            end
          else
            WordDelete := FALSE
        end
      else
        WordDelete := FALSE
    end
end; (* WordDelete *)

procedure BuildDict(wordFile : filename);
var  wrd : str255;
     wFile : text;
begin
  if NOT dictOpen then
    writeln('Dictionary not open...')
  else
    begin
      Rewrite(dfile,recsize);        (* create new dictionary file *)
      if FileExists(wordFile) then
        begin
          Assign(wFile,wordFile);
          Reset(wFile);                (* open word list file *)
          while (NOT Eof(wFile)) do
            begin                      (* add each word in list to *)
              readln(wFile,wrd);       (* the dictionary *)
              AddWord(wrd)
            end;
          Close(wFile)
        end
      else
        writeln('Word file ',wordFile,' not found')
    end
end; (* BuildDict *)

begin
 dictOpen := FALSE                     (* initialize dictOpen *)
end.

***
```

extra five bytes is the root pointer (record zero). Of course, you can change the record pointers, dLtrPtr, to four byte longints.

If you change to longints, be sure to change recsize to 9. This change will increase the node size from five bytes to nine bytes. The node size to info ratio will jump to 9:1.

The end-of-word flag uses the eighth bit of the stored character. If the character you need to store requires eight bits, an additional flag will have to be added to the dLtr record.

All words are stored in upper case so the dictionary is not case sensitive. If your needs require case sensitivity or the stored information is not cased, remove the calls to UpCaseWrd in the first lines of AddWord and WordFound.

The dictionary works well but is not bullet proof. It does not have extensive error checking. It is vulnerable to I/O errors like disk full, and it doesn't check for record sizes greater than 32767.

News Flash

It's here. Yes, it's actually here. Turbo Pascal 5.0 came yesterday. I finally broke down and upgraded from Turbo Pascal 2.0. Wow, what a difference. There's all these new goodies, units, procedure passing, linking, EXE files, debugger, 8087 support, limited EMS support and BGI graphics. I have lots to say next time.

Next Time

The theme of the next issue is software tools. Almost any unit presented here could be considered a software tool. But I've got BGI graphics now so look for graphics tools to enliven your user interface. Stay tuned...

♦ ♦ ♦

Continued from page 6

knowledge of data communications (I am well into my own commercial telecommunications program).

While I don't pretend to know all there is to know about computers, I have helped many people with the vast range of questions generated by people like Mr. Pettitt. I hope that next time Mr. Pettitt goes to his favorite computer store, the clerk on duty is not as intolerant as he.

As for you: well, you are forgiven for this oversight and I will continue to praise your magazine to all my customers. It is my personal favorite and I thank you for all your hard work in getting this fine publication out on the stands for all to enjoy.

John Mulligan
129 Lakeview Ave.
Syracuse, NY 13204

Editor's note: I assumed that Mr. Pettitt had been patronizing the wrong computer store. Certainly, I've found knowledgeable, forthright sales folk to be the exception. (I was ejected from the local ComputerLand after insisting that the assembler they had on floppy would run just fine on my hard drive. They not only wouldn't sell it to me, but I was persuasively escorted to the front door.)

However, Don Thompson, who's owner, salesman, and chief deliveryperson at MicroSphere, is very knowledgeable (and he's my brother), so I seek him out when I'm stumped.

Sunshine PROM Burner Fix

Forwarded to us from EASTNET is the following.

A recent note suggested that the Sunshine EPROM Burner may not burn several brands of EPROMs above 4000H. I definitely noticed that with mine. At the time, I could find no rhyme or reason to which parts would successfully burn and which ones wouldn't. However, I did find that by switching to a second program supplied with the burner, all problems with burning parts went away.

So, if you're having problems above 4000H, try using the program EPROM4.EXE instead of plain old EPROM.EXE. You may just have buggy software from Sunshine. Software 2000 may not be as devious as originally thought.

If you don't have EPROM4, go see your dealer. He should have it or be able to get it. Don't ask me for it... the version I have may not work with your burner.

from Mark — W3CCV

More Code!

Hello from down under. I, unfortunately, only discovered *Micro C* earlier this year. I've been using computers now for over eight years. I started with an OSI Superboard 2 running 65D, moved to a local hybrid system called a Rabble 65+ (an OSI C4/C8 clone with a Z80 CP/M system grafted on), then a Turbo XT clone, and now an AT clone.

But I digress. You should have *Micro C* put on the endangered species list. It seems to be the last of the computerist mags (hacker is a four letter word down here) and needs protection. Please keep it up. There's a lot for me to chew on.

I have a minor gripe about incomplete listings in the mag as your bulletin board is not a practical proposition from here. Having to order a disk each time would also get expensive. Have you considered putting out an archive disk? Maybe at a price premium with the year's listings on it. That would certainly be attractive to those far away. Curse the tyranny of distance.

R. M. Hess
P.O. Box 33
Bulleen 3105 Australia

Editors note: I understand the problem. Australia isn't anywhere near Bend. We're trying our best to run complete listings. (Assuming there's room for the editorial, of course.) Otherwise, we try to run the key portions so you can see what it is the author thinks he's saying.

More, More Code!

Why do people buy magazines? I buy them for entertainment and information. Why do I buy your magazine? Because *Micro C* is one of the few computer magazines written by people with a sense of humor, and it *was* one of the best values around. Unfortunately, its value is decreasing.

Why is this? Because of a practice I call source code extortion. Here's how it works. A magazine publishes an article describing a nifty new way to make a computer do something. The article gives enough information to interest readers, but not enough for them to write their own version. Then, somewhere in the text, either the author or the editor tells the readers the source code for the program is available. For a price!

I realize you are in business to make money and you are not the only magazine doing this. You are certainly not the worst offender. But it's sad to see one of the best computer magazines going downhill.

I'm sorry I've had to write this letter and hope that you change your policies. But if I don't see any improvement, I won't renew. I'm tired of paying to read article-long advertisements for source code disks.

Scott Rankin
2609 W. Southern Ave. #366
Tempe, AZ 85282

Editor's note: Great Scott. I know, I know. We're trying, very trying. People are welcome to download our software for other boards. Next week I'm going to start checking into the possibility of getting a local node so people with PC Pursuit can reach us.

Take a look at this issue. I think it's better.

Paradox Users Group

Thank you for your kind words about our users group in your recent Paradox review (*Micro C* Issue #43, p.34). The recognition we receive from people like you gives us a lot of encouragement.

The 800 number you listed for support is incorrect. The correct address and phone number for our group is shown below. I called the 800 number since you say it provides information about "third party support, including user groups." They wouldn't tell me who they were, but they certainly had never heard of us!

Brian J. Smith
President — LAPALS
901-A N. Pacific Coast Highway
Suite 200
Redondo Beach, CA 90277
(213) 374-0151

PT68K Update

I'd like to thank you and author Jack Crenshaw for the excellent review of the PT68K computer and SK*DOS in the November-December issue. I got a chuckle out of your comments about "warts" in products.

As it turns out, Jack had sent us a preliminary copy of his review last spring, and during the summer of 1988 we removed virtually all of the SK*DOS warts he'd found, although we did keep the few we felt qualified as beauty marks. Beauty is, after all, still in the eye of the beholder. It's important to remember that SK*DOS is *not* MS-DOS; so differences are not necessarily warts.

Here's what we did about the warts:

(1) SK*DOS now implements the "*" wild-card character for file names, extensions, and even directory names, although in a slightly different way from MS-DOS.

(2) When SK*DOS deletes a file, or copies a file over an existing file, it does indeed ask you for confirmation. This is something which we have no intention of changing (and I'm told that PC-DOS 4.0 now does the same). Nevertheless, there's an easy way for adventurous (foolhardy?) users to defeat the system — simply add "<Y" on the command line and all the prompts will automatically be answered with "Y".

(3) Unlike MS-DOS, SK*DOS runs on a variety of very different computers. Hence, as Jack puts it, we "developed a truly elegant system of I/O control, suitable for device-independent control of a wide variety of devices." We provide a sample driver to show people how to use I/O, but have let users or vendors write their own drivers.

(4) SK*DOS disk I/O on the PT68K is slower than MS-DOS for several reasons. We don't have DMA, and SK*DOS always defaults to "verify on." We have now changed the verify scheme so that in some instances verification will be done on an entire file or an entire disk. This has really sped things up.

(5) We've added subdirectories, but not in the same way as MS-DOS. SK*DOS lets you specify a default "system" drive/directory for loading programs, and a "work" drive/directory for data files. System commands can search within one directory, or across many using the directory wild-card character.

(6) The PT68K-2 now fully implements keyboard typeahead.

As I indicated above, SK*DOS is severely hampered by the need to adapt to a wide variety of hardware. In that respect, SK*DOS is more like the old CP/M — it has to run on many different hardware configurations, and so it takes a little work to take advantage of things like function keys.

On the other hand, it means that our user base also includes people who have other computers — like Marion Systems MS68K, NCR 7950, and Mustang 68020 machines. They all help to develop software to share through our Users Group, via our support BBS at (914) 241-3307, or via Mike Evenson's BBS at (817) 488-8398.

Peter A. Stark
President—Peripheral Technology
1480 Terrell Mill Rd.
Suite 870
Marietta, GA 30067

Teac Help Request

Anybody seen this one?

On an AT clone with a WD controller and 1.2M Teac for A:, 360K Teac for B:, and Award Bios; I boot from A: with DOS 3.3, no TSRs or cache. I remove the boot disk, put another disk into A: and do a directory read. The drive LED goes on and the disk spins, and there is no unexpected delay.

What I get is a normal directory display, but it's the directory of the boot disk which I had taken out before I did the DIR. If I DIR again, I get the boot disk directory again. If I B:<CR> A:<CR> DIR<CR>, it sees the new disk and behaves normally. If I swap the 1.2M drive into another AT, the fault moves with the drive. No other symptoms.

Another 1.2M Teac does not show the fault, but is probably an earlier model. I know that only some FD-55 drives have a "Disk change circuit ... used only on some models with disk sensor." (From p. 345 of the FD-55 manual.)

Do recent Teac 1.2Ms have some sort of cache for the directory? If not, where is it being held? Does the drive or the WD controller have a jumper to fix or disable this? Any ideas?

John Innes
120 MacPherson St.
Cremorne NSW, 2090 Australia

Editor's note: I could say something weird like "Your system has really become attached to its boot disk." But, of course, I won't. There is no memory in the 1.2M floppy drive. Your system stores the directory in a buffer. All systems do that but yours doesn't detect the disk change. I think there's a door-open signal your system isn't getting. Right? Help?

Seagate Solution

Here's yet another letter on the Seagate 225.

After reading all the recent 225 material in your magazine plus some stuff on the local BBS, I installed a new 225 and soon heard the endless seeking for track zero that others have described. I remembered the article mentioned leaving the front PC board screw loose.

Mine was just barely snug as received from the dealer so I assumed that the case wasn't as rigid as it appeared. I then found that the seek problem could be made better or worse by applying pressure to various parts of the case. The solution was to turn the drive over with the PC board down and use the side mounting holes. This seemed to tweak the case just the right amount.

Since going through this procedure, I haven't had a single incident of track zero seek trouble. In fact, this is the quietest and fastest inexpensive drive I've used.

You have a fine magazine; I depend on it for the real truth about small computers. Keep up the good work.

Tom Prouty
18639 Manhattan Place
Torrance, CA 90504

♦ ♦ ♦

CP/M NOTES

Who's Alive In CP/M?

By Samuel Vincent
CP/M Special Interest Group
Pittsburgh Area Computer Club
P.O. Box 6440
Pittsburgh, PA 15212

Samuel and his friends have done a great service by compiling this list of active CP/M software suppliers. When I spoke with him on the phone, he stressed the fact that this isn't a static project: any help maintaining the list will be greatly appreciated.

This issue we begin part 1: the CP/M commercial software vendors. In the future we'll also list public domain software suppliers and Z-Nodes that sell CP/M products. We've edited things slightly to fit the mag. A complete version (with prices when available) lives on the Micro C BBS and the Issue #45 disk.

This list, produced by the CP/M SIG of the Pittsburgh Area Computer Club (PACC), is based on a listing released previously by others; however, we've greatly expanded it.

While far from a complete listing of CP/M companies, vendors, and user groups, we hope it will become a major reference source for all CP/M users. We also hope that this list will continue to expand and be maintained so that it will be worthwhile re-releasing it annually or semi-annually.

Finally, while every effort has been made to verify the information contained in this list, the CP/M SIG of PACC cannot be held responsible for errors in the list, nor can we make any guarantees on the continued availability or suitability of the software described.

Please send 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).

$Ware Tools for Investors
P.O. Box 645
San Luis Rey, CA 92068
(619) 941-1241
* Verified 8/31/88 *

M.A.G.I.C.-Stock market analysis program.
Free brochure available
Demo/tutor package also available.

Advent Products
3154-F LaPalma Ave.
Anaheim, CA 92806
(714) 630-0446
* Verified 4/15/88 *

Specializes in Kaypro add-ons. Selling out current inventory - mostly games.
No Catalog.

Agricultural Software Consultants, Inc.
1706 Santa Fe
Kingsville, TX 78363
(512) 595-1937
* Verified 4/15/88 *

Catalog available.

AGS Software
94 Cranberry Hwy.
P.O. Box 28
Orleans, MA 02653
(617) 255-0510
* Verified 4/15/88 *

Astrological software
Catalog available.

Alcor Systems

See MIX Software

Alexander & Lord
DRI Distribution Center
P.O. Box 2110
Carmel Valley, CA 93924
(408) 659-2203
* Verified 8/15/88 *

Complete line of DRI products for CP/M-80, CP/M-86,CP/M-68K,CCP/M, Concurrent DOS, and MS-DOS.
Free brochures.

Alpine Data
635 Main St.
Montrose, CO 81401
(800) 525-1040
(303) 249-1400
* Verified 4/15/88 *

Income Tax program, and Depreciation Schedule program.
Free brochures.

Alpha Systems Corp.
711 Chatsworth Place
San Jose, CA 95128
(408) 297-5594
(408) 432-0821 (Z-Node Central BBS 300/1200/2400)
* Verified 7/15/88 *

Z-System software and books. Z-80/Z-280/HD-64180 programming tools
Free catalog available.

Analytical Products

See Anapro

Anapro
6905 El Camino Real #4
Atascadero, CA 93422
(805) 466-1589
* Verified 7/29/88 *

Dealer-Heath/Zenith
8-bit soft and hardware
CP/M to DOS transfer
WordStar 4.0 for CP/M
SMART CHECKBOOK, etc.
Free catalog available.

Anderson Techno-Products
947 Richmond Rd.
Ottawa, Ontario
Canada, K2B 6R1
(613) 722-0690
* Verified 8/15/88 *

POOR MAN'S NETWORK
LAN for use with 2 computers.
Free brochure available

Application Engineering
2238 Grey Fox Court
Orange Park, FL 32073
(904) 264-4619
* Verified 8/31/88 *

TAXPAD-personal income tax software template for use with SuperCalc
Free brochure available

Ashton-Tate
20101 Hamilton Ave.
Torrance, CA 90502-1319
(213) 329-8000
(213) 329-9989 (Customer service)
* Verified 8/15/88 *

dBASE II, version 2.43*
Limited quantities for upgrades only. Contact cust service for info.

BD Software, Inc.
42 Fresh Pond Place
Cambridge, MA 02138
(617) 576-3828
* Verified 8/15/88 *

BDS C compiler, comes with RED full screen editor and debugger.
No literature available

Borland International
4585 Scotts Valley Dr.
Scotts Valley, CA 95066
(800) 543-7543
* Verified 6/15/88 *

TURBO Pascal-only v 3.0 available for CP/M.
TURBO Data Base ToolBox also available for CP/M
Selling out inventory.

C.C. Software
1907 Alvarado Ave.
Walnut Creek, CA 94596
(415) 939-8153
* Verified 8/31/88 *

Source Code Generators for CP/M 2.2, CP/M Plus and TURBO Pascal.
Masterful Disassembler
Free brochure available

CDE Software
4017 39th SW
Seattle, WA 98116
* Verified 8/15/88 *

CHECKS&BALANCES-single entry accounting.
Free brochure available

CMB3 Technologies, Inc.
P.O. Box 3061
Walnut Creek, CA 94598
(415) 685-7737
* Verified 4/15/88 *

WORDPATCH - Patches WordStar

Centennial Software, Inc.
1873 S. Bellaire St.
Suite 510
Denver, CO 80222
(303) 691-5586

REALFOCUS - Real estate package

<-- phone rang, but no answer

Central Computer Products
330 Central Ave.
Fillmore, CA 93015
(800) 533-8049
(800) 624-5628 (CA only)
(805) 524-4189
* Verified 6/15/88 *

More than 100 software and hardware products for CP/M computers.
Disk format conversion service also available.
Free catalog available.

CompuMAGIC, Inc.
P.O. Box 437
Severn, MD 21144
(301) 969-8068
* Verified 8/31/88 *

Loan amortization.
CompuMAGIC Utility Package, now includes SEARCH.
Free catalog available.

Computer Editype Systems
509 Cathedral Parkway
Suite 10A
New York, NY 10025
(212) 222-8148
* Verified 4/15/88 *

MAGICPRINT - Desktop publishing

Computer Professionals,Inc
1508 West Blue Ridge Dr.
Greenville, SC 29611
(803) 294-0494
* Verified 7/29/88 *

Accounting - G/L, A/R, A/P, and Payroll. 45 days free phone support
Free catalog available.

CompuView Products
1955 Pauline Blvd., Ste. 200
Ann Arbor, MI 48103
(313) 996-1299
* Verified 7/15/88 *

VEDIT Plus - Multifile Word Processor. Also avail - Print Formatter and Spelling Checker
Free catalog available.

Condor Computer Corp.
1490 Eisenhower Place
Ann Arbor, MI 48108
(800) 451-3267
(313) 971-8880
* Verified 8/15/88 *

CONDOR 3, Relational Database Management System, Release 2.11.07
Free brochure available

Cranberry Software Tools
P.O. Box 681
Princeton Junction, NJ
08550
* Verified 7/29/88 *

ALPHA TEXT - word processor. Other CP/M programs. Write for free brochure.

Digital Marketing Corp.
P.O. Box 2010
Walnut Creek, CA 94595
(800) 826-2222
(415) 947-1000
* Verified 8/15/88 *

RANDOM HOUSE PROOFREADER, grammar checker project planner, time/equipment scheduler, etc.
Free catalog available.

DynaComp, Inc.
DYNACOMP Office Building
178 Phillips Road
Webster, NY 14580
(800) 828-6772(outside NY)
(716) 245-4040
* Verified 6/15/88 *

Scientific, engineering financial, statistical, educational, and other programs. Disk conversion service also available. Catalog $2.

Echelon, Inc.

See Alpha Systems Corp.

Elliam Associates
1230 Bay View Heights Dr.
Los Osos, CA 93402
(805) 528-5871
* Verified 8/31/88 *

FORMS 4-forms entry program. Other programs for CP/M and Z-system. PD software. Send SASE for a free brochure.

Ellis Computing
5655 Riggins Ct. Suite 10
Reno, NV 89502
(702) 827-3030
* Verified 4/15/88 *

Nevada BASIC, FORTRAN, Pascal, and PILOT.
Selling out CP/M inventory. Cobol no longer available.

List will be continued next issue.

011
100
100
TIDBITS

Errors And Traps

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

There are traps everywhere you turn nowadays. However, Gary's traps are good traps, not those spring loaded, hair triggered, 13-year-old traps you find watching you across the dinner table.

One big difference between a program you write for yourself and one you intend others to use is the degree of error checking you include in the finished application.

If *you* break your program, you can usually fix it. If your client or customer breaks it, you might find yourself on a Mafia hit list.

At least a dozen critical errors can get you in trouble —

(0) Attempt to write on a write-protected disk;
(1) Unknown unit;
(2) Drive not ready;
(3) Unknown command;
(4) CRC error in data;
(5) Bad drive request structure length;
(6) Seek error;
(7) Unknown media type;
(8) Sector not found;
(9) Printer out of paper;
(10) Write fault;
(11) Read fault;
(12) General failure;

These errors are often unavoidable (you can't make a client shut his drive door!). But you can catch (and fix) them before the system crashes.

The secret is to trap the error and let your application deal with it.

Turbo Prolog Trap

I've been trapping bunches of errors with a handy dandy Turbo Prolog standard predicate (or function to you C'ers and Pascal types) called (appropriately) "Trap."

Trap takes 3 arguments: 2 predicate calls and 1 integer variable —

```
trap(P1,I,P2)
```

P1 is the predicate we want to trap. P2 is the predicate we want to call if there's an error. I is the error code returned by trap, if there's an error. If there isn't an error, trap is transparent. The predicate, P1, will execute as if trap weren't there.

For example, suppose we try to open a non-existent file (see Figure 1).

Without trap, the system crashes when it can't find the file. With trap, and instructions to handle the error (see Figure 2) —

The predicate fails but the application doesn't. We write an error message and move on.

Most languages allow programmers at least some control over runtime errors. At a low level, you can call the DOS critical error handler (at interrupt 24h), which will handle all the errors I've listed.

But trap has a subtle beauty — besides allowing you to handle errors at a very high level, it also resets the stack, global stack, and trail (used by reference variables) to the values they had before the predicate (P1) was trapped. So we can use trap to catch memory overflow errors.

For example, suppose we write a recursive predicate (or function) that generates an integer overflow (see Figure 3) —

If we don't tell the compiler to look for overflows, and in addition don't trap the overflow, the program will crash.

If we do toggle the Compiler/Option integer check to On and trap, we can write an error message and recover. (See Figure 4.)

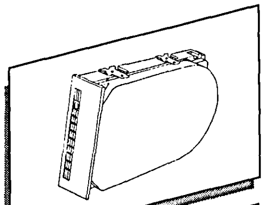
Trap is an incredibly powerful feature included in Turbo Prolog 2.0. I'd be lost without it.

I don't have any data, but I'd guess that at least 1/3 of a programmer's development time goes into running down errors. Unfortunately, some errors simply won't appear until a creative user finds them, so critical error handling at the level of trap is an excellent solution.

Trapping SOG Errors

Every time I make a list thanking folks for helping out, solving a problem, or clarifying one of my mistakes, I omit someone. This time it was my SOG VII list (in Tidbits, Issue #44).

I wanted to thank all our SOG speakers for generously donating their time to our annual event. I thanked (I think) all but one. Mike Donovan, I'm sorry you were the one.

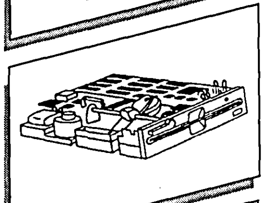


20 MEG HARD DRIVES

SEAGATE Model ST-225
(Full 6 Mo. Warranty)*less face plate* $219.95

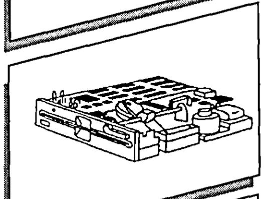
Western Digital Controller For Model ST-225 Drive
$69.95 w/cable

Yes!! These are for IBM compatible machines.



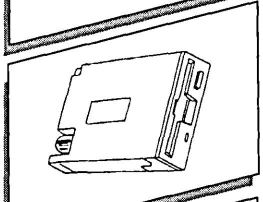
SHUGART # 465

QUAD DENSITY, 720 K, 96 TPI
1/2 HEIGHT
BLACK FACE PLATE
(NEW BUT HAVE COSMETIC BLEMISHES)
SOLD AS IS $49.95



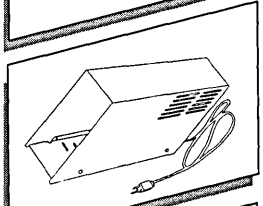
TOSHIBA 5 1/4" FLOPPY DRIVES

DSDD, 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

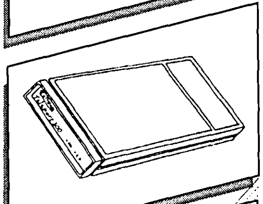
FULL 90 DAY WARRANTY $119.95



5.25" DISK DRIVE CABINETS
(with power supply)

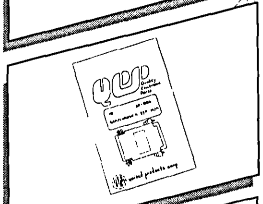
For 1 full height or 2 half height
#CAB-25V5

$99.95 each



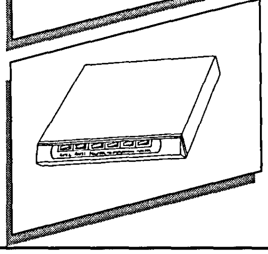
300 BAUD SMART MODEM (DIRECT CONNECT)
LOW COST SERIAL MODEM
7 x 10 x 1 1/2

$19.95



CABLES-CONNECTORS-SEX CHANGERS

"XT" STANDARD PARALLEL (10', DB-25M TO CN36M)	#10-004	$12.95
AC POWER CORD (6' COMPUTER STANDARD, MOLDED RUGS)	#10-001	2.95
RS232 MODEM ADAPTER (DB-25 M-F, PINS 2 & 3 REVERSED)	#20-001	9.95
SEX CHANGER, DB-25 F TO F	#20-005	9.95
SEX CHANGER, DB-25 M TO M	#20-006	9.95



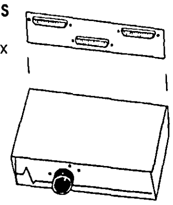
COMPUTER POWER SUPERVISOR (A LOW COST INSURANCE POLICY
PROVIDES SURGE/SPIKE PROTECTION
FOR YOUR COMPUTER!
ALL INCOMING POWER IS MONITORED & CONDITIONED BEFORE ALLOWING IT TO POSSIBLY TRASH YOUR COMPUTER OR OTHER SUCH VUNERABLE & EXPENSIVE EQUIPMENT.

- 5 OUTLETS, EACH WITH A LIGHTED SWITCH
- 1 MASTER ON/OFF SWITCH
- SYTLISH CABINET PROVIDES FULL SHEILDING
- RATED 15 AMP, 125 VAC, 1875 WATTS

DIM.12.5" x 2.5" x 14" Regular price $69.95 Your price $49.95

5.25" HARD DISK
CONTROLLER CARD
FOR ANY MODERN 5.25" HARD DRIVE

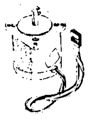
$89.95 W/CABLE



COMPUTER A-B SWITCH BOXES

#CN36 A-B Parallel Centronix
$24.95 each

#DB25 A-B Serial RS-232
$24.95 each



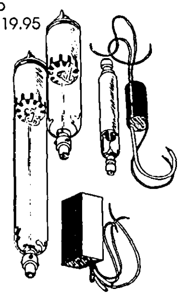
STEPPER MOTORS

Copal #SP-57
1/4" Shaft, 7.5 deg./step, 36 Ohm, 12VDC $6.95
3/16" Shaft, 35 oz." torque, 2.1A., 5VDC
WITH BRASS GEAR, 20 TEETH, 1/2" DIAMETER $9.95

LASERS

5 MW Laser Tube $89.95
Power Supply Kit (115VAC)$69.95
Power supply (wired) (12VDC)$119.95

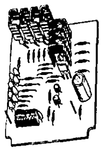
1 MW Laser Tube$119.95
Power Supply (12VDC)$99.95



(These lasers are brand new and guaranteed to have a cosmetic defect or not meet manufactures full specifications. All are tested in our lab to insure your satisfaction.)

WARNING: Voltages present and used by lasers can be lethal...Permanent eye damage could result from direct exposure to an on coming laser beam. Only those persons qualified to handle such potentials should do so...

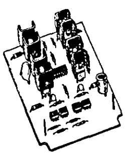
TECHNA-KIT



D-C Motor Controller

- Control 2 D-C motors with a computer or other logic source
- For motors rated 6-24 VDC
- Control forward/reverse/run/cw/ccw/stop
- Up to 6 Amp starting surge, 4 Amp cont.
- Dynamic breaking (capable)
- Will also run most 4-lead stepper motors

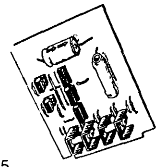
$29.95



USMD-C

- Control standard 6-lead stepper motors with a computer or other logic source
- For motors rated 1.7 - 12.0 VDC
- Optical isolation
- Control: forward/reverse/step rate/stop
- Industry standard 22 pin edge card connector

$29.95



COMPUTER AUTOMATE

- Use your computer to provide automation
- 8 separate driver ports per card
- 8 TTL/Cmos inputs
- 1 user defined sense switch
- 6-24 VDC
- 4 Amps/driver (max current)

$29.95

united products corporation
DISTRIBUTORS OF ELECTRONICS SINCE 1968

1123 VALLEY STREET • SEATTLE, WA 98109-4425

VISA
AMERICAN EXPRESS

PHONE: (206) 682-5025
FAX: (206) 682-5593
M-F 9-6 SAT 9-5

Reader Service Number 45

Thanks for an illuminating discussion of Pseudo Concurrency in Turbo Pascal. Now, please stop sending me those threatening notes.

Errors Of Our Ways

In Fort Worth, Texas, an ex-programmer, Donald Burleson, became the first person to be convicted of creating a computer virus.

Don, apparently, annoyed at being fired by his employer, an insurance company, planted a time-bomb which activated itself two days after his firing. The bomb wiped out 168,000 records (including backups?). Incidentally, the prosecutor referred to his handiwork as a "rogue program" — first time I've heard that terminology.

The District court jury deliberated six hours before delivering the first conviction under the three year old Texas computer sabotage law. The key to the conviction — Don's virus program was introduced as evidence. (What can you trust nowadays?)

The charge, harmful access to a computer, is a third degree felony bearing a penalty of up to ten years in prison and up to $5,000 in fines.

Boy did he make a mistake.

Editor's note: I wonder if they'll let him teach programming classes in the pen.

I'm out of here.

♦ ♦ ♦

Figure 1 — Attempt To Open Nonexisting File

```
DOMAINS

file = dd

PREDICATES

process
main

CLAUSES

process:-
      writedevice(dd).  % This file doesn't exist.

main:-
      process.  % process fails crashing the system.
main:-
      write("\nWithout trap we can't get here.").

GOAL
      main.

***
```

Figure 2 — Trapping Error From Figure 1

```
DOMAINS

file = dd

PREDICATES

process
error(INTEGER)
main

CLAUSES

process:-
      writedevice(dd). % This file doesn't exist.

error(E):-
      writef("\n\n>> Error code=%", E),
      beep.

main:-
      trap(process,E,error(E)).
main:-
      write("\nThere's an error, but it's ok.").

GOAL
      main.

***
```

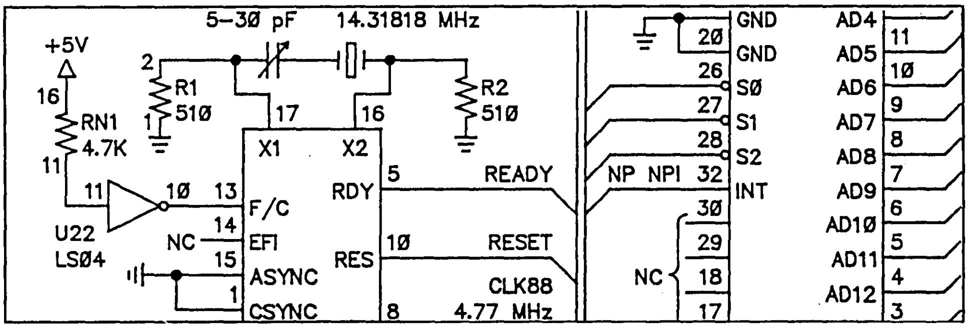
Figure 3 — Overflow Condition

```
DOMAINS

   file = dd

PREDICATES

process
loop(INTEGER)

CLAUSES

loop(A):-
      B = A + 1,
      !,loop(B).

process:-
      loop(30000). % process crashes
                   % integer overflow
process:-
      write("\nWe don't get here.").

GOAL
      process.
***
```

Figure 4 — Trapping Overflow Condition

```
DOMAINS

   file = dd

PREDICATES

process
error(INTEGER)
loop(INTEGER)

CLAUSES

loop(A):-
      B = A + 1, % Must have compiler option
                 % integer check on.
      !,loop(B).

error(E):-
      writef("\n\n>> Error code=%",E),
      beep.

process:-
      trap(loop(30000),E,error(E)).
process:-
      write("\nWaiting after error"), % No crash.
      readchar(_).

GOAL
      process.
***
```

XT SCHEMATIC



At last you can plumb the mysteries of your computer with this single sheet schematic of the IBM XT's main board. A wealth of information for both True Blue *and* clone owners.

Need to know just how a non-maskable interrupt occurs (and how to mask it)? Is your keyboard dead (or do you just want to know how to disable it)? A trip through our schematic will answer your questions.

Although clones use slightly altered board layouts and different chip location names, they're close enough to the original for this schematic to be very useful. As an example — you have a dead clone. Lil sucker won't even beep. A look at the schematic shows the location of parallel port A. You know that the power on self test loads a checkpoint number into port A before each test. So now all you have to do is read port A with a logic probe to see how far the system went before it puked.

We'll include a list of these checkpoint numbers and some other pertinent trouble shooting information with the schematic.

IBM PC-XT Schematic **$15.00**

CP/M KAYPRO SCHEMATICS

Of course, we still provide a complete schematic of the processor board in your CP/M Kaypro. It's logically laid out on a single 24" by 36" sheet and comes complete with an illustrated theory of operation that's keyed to the schematic. You get detailed information available nowhere else.

For instance, those of you with the 10 and newer 84 systems get a thorough run down of the processor board's video section complete with sample driver routines. All packages contain serial and parallel port details and programming examples. Also coverage of the processor, clock, I/O, and disk controller (information that's not even available in Kaypro's own dealer service manual!).

Kaypro II & IV (pre-84)	**$20.00**
Kaypro 10 (without modem)	**$20.00**
Kaypro 2, 4, and 10 (84 series)	**$20.00**

NOTE: These packages cover *only* the main boards. You're on your own when it comes to disk drives, power supplies, video cards, etc.

Phone Orders: (503) 382-5060 or 1-800-888-8087 Mon-Friday 9 am-5 pm PST

Mail Orders: P.O. Box 223, Bend, Oregon 97709

Techtips

DOS Stream Filtering

Regarding your article, "Hercules Graphics Printer Dump" in Issue #44: the problem with control-Zs in the graphics print stream is easily explained. MS-DOS is the culprit filtering the output stream.

Normally, when you do input or output to a file with DOS function calls (as I'm sure the Turbo C putc() function does), all stream data including ^Z is left intact. However, when the output goes to a device (i.e., COM, PRN, LPT, AUX), stream filtering is normally done by DOS, although I can't imagine why.

You can demonstrate this simply by copying a file with embedded ^Z characters to a device, using the MS-DOS COPY command. When the copy operation reaches the first ^Z it stops because DOS considers that to be an end-of-file character. In some versions of MS-DOS (notably the versions distributed by Heath/Zenith), the command "COPY /B" will do a straight binary copy without filtering.

There are two solutions to this filtering problem ...

One is to avoid using MS-DOS for output. This can be done by using the ROM BIOS routines (as you suggested in your article), or by communicating directly with the output port.

The problem with avoiding MS-DOS altogether is that now the program is not portable to other DOS machines. In your screen dump example, this isn't important because the Hercules card only fits in a PC compatible computer. But many times portability is a chief concern.

A more elegant solution to the filtering problem is to use MS-DOS function call 44H (I/O Control for Devices) to place the output device channel in the "raw" mode. Raw in this case means unfiltered, not uncooked. This can be done with the code in Figure 1.

Of course you would also want to check for a legal device name, device ready status, and so forth, but you get the idea.

Paul F. Herman
Software Graphics Tools
3620 Amazon Dr.
New Port Richey, FL 34655

Figure 1 — I/O Device Control

```
mov     ax, 3C00H
mov     dx, offset _device      ; get pointer to device name
mov     cx, 0
int     21H                     ; open output device
jc      ERROR                   ; if error, exit
mov     bx, ax                  ; get device handle in BX
mov     ax, 4400H
int     21H                     ; get device status in DX
or      dx, 0A0H                ; set raw bit in status word
mov     ax, 4401H
int     21H                     ; put device in raw mode
ret

***
```

Extra COM Ports

Old IBM PCs have space for two communication ports, COM1 and COM2. The base addresses for these ports are 03F8h and 02F8h, with interrupts IRQ4 and 3, respectively.

The BIOS POST (Power On Self Test) routine goes out and checks for the ports' addresses and writes those addresses to locations 0000:0400 through 0000:0407. (Remember if you want to use DEBUG to dump out those addresses, they'll be displayed in low byte-high byte order. COM1's address of 03F8h will be displayed as F8 03.) These memory addresses are used by the BIOS and some other programs

Micro Ads

A Micro Ad is the inexpensive way to reach over 22,000 technical folks like yourself. To place a Micro Ad, just print out your message (make it short and sweet) and mail it to Micro C. We'll typeset your ad (no charge) and run it in the next available issue. You can also send camera ready copy. Rates: $99 for 1 time, $267 for three times, $474 for 6 times (a best buy at only $79 per insertion). Full payment must accompany ad. Each ad space is 2 1/4 inches by 1 3/4 inches.

S-100 LIQUIDATION

Intercontinental Micro

CPZ-48006	$95	SAS-200	$10
CPS-B8A	$50	RPB-200	$5
CPS-B6A	$40	MPB-100	$4

Other Items

OMTI-5300	$85	MD-1013	$35

Order/Info: **215-233-4199**
Craig Tiano, 9303 Stenton Ave.
Erdenheim, PA 19118

Reader Service Number 122

F L A S H

THE **DISK ACCELERATOR**

PC MAGAZINE EDITORS' CHOICE

- EASY to Install
- Cache up to 3 MEGS of EXTENDED and or EXPANDED
- Buffers up to 26 DEVICE driven drives
- Comes with 2 FREE utilities!!!!!

ORDER NOW ~~$69.95~~
(800) 25-FLASH $19.95*

SOFTWARE MASTERS 6352 North Guilford Ave.
Indianapolis, In 46220 / (317) 253-8088

*To receive discount price, DEMAND IT!!
$5.00 Shp/hnd in USA & CANADA, $15.00 overseas.

CROSS ASSEMBLERS

PseudoCode releases version 2 of its cross assemblers. Assemblers for the 8048, 8051, 8096, 8085, z80, HD64180, 6301, 6303, 6502, 1802, 6800, 6805, 6809, and 68000 microprocessor families are available. Macros, Conditional Assembly, Include Files plus extensive expression handling. Virtually no limit to program size. For IBM PC's and true compatibles with MS—DOS 2.0 or greater, and 256K memory. Complete with printed manual for $35.00. Each additional is $20.00. (Michigan residents add 4% tax). Shipping and handling $5.00 USA, $10.00 Canada, $15.00 elsewhere. Visa/MC. Order from distributor: **KORE Inc., 6910 Patterson, Caledonia, MI 49316, 616-791-9333.**

Reader Service Number 75

snOOp - World's Best Disassembler. Turns any program to assembler source, comments each line of code. 8086 to 80386 processors and coprocessors. DOS calls, ports, EMS functions, topview calls and gets - all supported and unsupported codes. Batch or interactive. Built in code-sensitive help. Best way to: Learn Assembler, modify software when source is unavailable, find and disarm viruses, or just snOOp. $49.95 TriDOS

4004 SW Barbur • Portland, OR 97201
VISA/MC, AMEX, COD 800-237-9111

Reader Service Number 117

Get Inside Your Program Today!

Analyze and improve your program . . . function by function! Inside follows your program through execution counts, minimum, maximum, and total elapsed times with microsecond accuracy! Get inside your programs with unprecedented detail now!

Inside! Turbo C
Inside! Turbo Pascal
Inside! Quick C
Inside! Quick Basic
Inside! Microsoft Pascal
Inside! Microsoft Fortran
Inside! Lattice C
Inside! Logitech Modula-2

$75.00 Each

Visa/Mastercard Accepted
(800)537-5043
Paradigm Systems
P.O. Box 152 Milford, MA 01757

Reader Service Number 113

ONELINERS

for IBM PCs and compatibles

A pop-up (or command line) utility that instantly alphabetically lists both the current directory and the 1st line of all ASCII text files in a full screen window.

Automatically translates Wordstar and Wordperfect 4.2 files (including summary boxes) to text.

Indispensible for quickly identifying data, program source, word processor, etc, files from inside a running application.

ONLY $15

GNM Endeavors, Inc.
1910 Fieldwood Drive
Northbrook, IL 60062

DID YOU TYPE "CD" TODAY?

DOS users, why crawl around your hard disk with CD when you can JUMP instead? JUMP offers instant four-way access to any area of your hard disk: via menu choice, shorthand or symbolic name, or pathname. JUMP also has remote program users. Introductory price just $34.95 with special bonus! Write for free info on our complete line of DOS and CP/M programming and word processing products.

Cranberry Software Tools
P.O. Box 681
Princeton Junction, NJ 08550-0681

68000 SOFTWARE

• K-OS ONE operating system uses MS-DOS disks with source code	$50
• K-OS ONE manual	$10
• HT68K SBC w/K-OS ONE	$395
• Screen Editor Toolkit	$50
• HT-FORTH	$100
• BASIC	$149

Free Newsletter & Spec Sheets

HAWTHORNE TECHNOLOGY
1411 S.E. 31st Ave., Portland, OR 97214
(503) 232-7332

Reader Service Number 34

ACCELERATE YOUR AT'S MATH

Most AT's and clones run their 80287 math coprocessors at 4 to 6 Mhz. So if you have an 8 or 10 Mhz 80287 in your system, it's **loafing!** The Solution: Speed-up your AT's 80287's clock rate to it's maximum frequency with a coprocessor daughter-board. 8 or 10 Mhz versions available for $29.95 (80287 not included). Simple installation.

Sierra Circuit Design
18185 West Union Road
Portland, Oregon 97229
(503) 645-0734

8051 Z8 / Super8 C COMPILER

*** Call today for a FREE technical bulletin ***
MICRO COMPUTER CONTROL
P.O. Box 275 – Hopewell, NJ 08525 USA
Telex 9102404881 MICRO UQ
(609) 466-1751

Reader Service Number 100

OPT-TECH SORT/MERGE

Extremely fast Sort/Merge/Select utility. Run as an MS-DOS command or CALL as a subroutine. Supports most languages and filetypes including Btrieve and dBase. Unlimited filesizes, multiple keys and much more! MS-DOS **$149.** XENIX **$249.**

(702) 588-3737
Opt-Tech Data Processing
P.O. Box 678 - Zephyr Cove, Nv 89448

Reader Service Number 64

16 Megabytes EMS and/or Extended Memory

- Works on 8 or 16 bit bus
- 16 bit transfer on AT bus
- Single board design
- Includes RAM disk and extensive diagnostics
- Quantity/OEM discounts

XT and AT Compatible

Designed, Manufactured, Sold and Serviced by R-Tech

907 North 6th St. Lake City, MN 55041 (612)345-4555

Reader Service Number 54

to talk to the ports.

Even though only two COM ports are defined, DOS sets aside space in memory for the addresses of four. The space for the latter two hasn't been taken advantage of until the release of DOS 3.3.

Until the release of the PS/2, IBM and almost all other manufacturers did little about any extra COM ports. A few manufacturers created expansion cards with port addresses of 03E8h and 02E8h for COM3 and COM4. Most of these boards allowed the selections of IRQ2, 3, 4, or 5.

However, at least one chose IRQ10 and 11 to use in the AT. (Remember that ATs have a second slaved interrupt controller for a total of 15 IRQs instead of the 8 in the PC/XT. On the PS/2, IBM chose to share IRQ3 for COM2 through 8.

If you were to manually write the addresses for a third and forth COM port to the RAM BIOS area, MODE could redirect parallel output to these addresses. MODE doesn't care what the IRQ is and at work we set up COM1 and COM3 on the same IRQ. Then we redirected LPT1 to COM3 with MODE at 9600 baud and used COM1 at 9600 baud with a terminal emulation program to a minicomputer. We proceeded to log onto the mini and sent all incoming data on the screen to the printer. We saw no loss of characters after listing out a six or seven page document.

A co-worker, Bill Hitch, who is very good at assembly, wrote a program to check for the two standard and two extended addresses on the cards we ordered. I disassembled his code to find the part necessary to check if a serial port exists:

```
mov  dx,3FAh
in   al,dx  ;port 3FAh
test al,0F8h
jnz  label  ;port not there
```

The equipment list in the RAM BIOS area (0000:0410-11h) requires updating.

In our case we need to update bits 9-11 in the second (most significant) byte with a count of COM ports (0-4).

Ports	Bit Pattern
0	xxxx000x
1	xxxx001x
2	xxxx010x
3	xxxx011x
4	xxxx100x

MODE doesn't care if this area has been updated or not, but other pro-

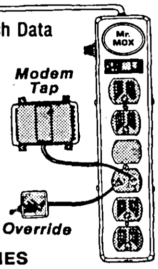
Figure 2 — Set Up COM Ports

```
seg_c           segment para public
        assume  cs:seg_c, ds:seg_c

start:
        mov     ax,0
        mov     es,ax
        mov     bx,0            ; initialize port count
        mov     dx,3FAh
        in      al,dx           ; read port 3FAh
        test    al,0F8h
        jnz     port_2          ; port not there
        inc     bx              ; it's there, count it
        mov     word ptr es:400h,3F8h  ; base address to memory
port_2:
        mov     dx,2FAh
        in      al,dx           ; read port 2FAh
        test    al,0F8h
        jnz     port_3          ; port not there
        inc     bx              ; it's there, count it
        mov     word ptr es:402h,2F8h  ; base address to memory
port_3:
        mov     dx,3EAh
        in      al,dx           ; read port 3EAh
        test    al,0F8h
        jnz     port_4          ; port not there
        inc     bx              ; it's there, count it
        mov     word ptr es:404h,3E8h  ; base address to memory
port_4:
        mov     dx,2EAh
        in      al,dx           ; read port 2EAh
        test    al,0F8h
        jnz     done            ; port not there
        inc     bx              ; it's there, count it
        mov     word ptr es:406h,2E8h  ; base address to memory
done:
        mov     cl,1
        shl     bx,cl           ; shift w/zeros fill
        mov     al,es:411h      ; get the equipment flag MSB
        and     al,0F1h         ; zero out bits 9-11
        or      al,bl           ; insert port count in bits 9-11
        mov     es:411h,al      ; write data to equipment flag MSB
        int     21h
seg_c   ends

        end  start

***
```

Figure 3 — DEBUG Session To Alter Floppy Step Rate

```
C>debug
-a                              ; invoke DEBUG's assembler
????:0100 xor ax,ax             ; zero the AX register
????:0102 mov ds,ax             ; DS = 0
????:0104 mov bx,522            ; offset of DPT
????:0107 mov byte ptr [bx],ef  ; new step rate (ef = 4 msec) to DPT
????:010A int 13                ; reset FDC (AH still = 0 from above)
????:010C int 20                ; return to DOS
????:010E                       ; <cr> here exits assembler
-nfaststep.com                  ; name file and set up FCB
-rcx                            ; cx register gets file length
CX 0000                         ; current value of cx
:e                              ; change to e, length of file
-w                              ; write the file to disk
Writing 000E bytes
-q                              ; quit DEBUG

***
```

more Micro Ads...



grams may. (PCTOOLS doesn't care, but DIAGS does.)

I tried to reinvent a simple program to accomplish the same end result. I'm a beginner when it comes to assembly language so I'm sure there are more efficient ways of doing this. I have, therefore, named this program COM_POST. (See Figure 2.) Of course you should change the port 3 and 4 base addresses according to your system.

Rob Aprato
323 75th St. #17
Everett, WA 98203-5561

High Steppin' Drives

About a hundred years ago (*Micro C* issue #30) I wrote an article explaining how to speed up the step rate of clonal floppy drives, and, in the process, quiet those noisy puppies.

Frank Schoettler called the other day to say that the code wasn't working in his AT. "Impossible," I said. "My code runs on anything, even a Veg-A-Matic." But a quick check showed he was right.

Backing up for a moment, parameters for the floppy drive controller (FDC) live in the disk parameter table (DPT) at address 0000:0522. The first byte of the DPT controls the step rate. But simply changing that byte (as in issue #30) isn't enough. We also need to reset the FDC to force it to use the new parameters.

Figure 3 shows the new DEBUG session including the FDC reset code. An interrupt 13h with AH set to zero does the reset. Notice that AH still holds zero from the first XOR so the only added line is the int 13. Figure 4 does the same thing in Turbo C.

Take a look at Figure 4 for step rate options. My Teacs run well at 4 msec while Mitsubishis work at 2 msec. I haven't tried this method on 3 1/2" drives, but it does work with one mystery 5 1/4" 1.2 MByte drive at 4 msec. (No, I'm not interested in opening up an AT box at this hour of the morning just to satisfy your curiosity.)

You'll need to experiment with your drives to find the optimum value. Don't worry, they'll let you know when you've gone too far (gag!).

Larry Fogg—Micro C Staff

♦ ♦ ♦

Figure 4 — Turbo C Step Rate Code

```
/* step.c - alters disk parameter table to vary floppy step rate */

#include <stdio.h>
#include <dos.h>

const int step_rate = 0xef;                          /* 0xcf = 8 msec */
                                                     /* 0xdf = 6 msec */
                                                     /* 0xef = 4 msec */
                                                     /* 0xff = 2 msec */
                                                     /* 0xgf = 0 msec */
main ()
{
  pokeb (0x00, 0x522, step_rate);    /* set new step rate in DPB */
  _AH = 0x00;                                /* set up for reset */
  geninterrupt (0x13);                /* reset floppy controller */
}

***
```

Is There A Gap In Your Info?

Fill in your Back Issues of Micro C today!

ISSUE #1 (8/81)
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
FORTHwords Begins
16 pages

ISSUE #4 (2/82)
Keyboard Translation
More 4 MHz Mods
Modems, Lync, and S10s
Undoing CP/M ERASE
Keyboard Encoder
20 pages

ISSUE #5 (4/82)
Word Processing
Two Great Spells
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)
BBII EPROM Program
Relocating Your CP/M
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)
256K for BBI
Bringing Up BBII
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
Cheap Fast Modem
Nevada COBOL Review
BBI Printer Interface
Kaypro Reverse Video Mod
44 pages

ISSUE #14 (10/83)
BBII Installation
The Perfect Terminal
Interface To Electronic
Typewriter
BBI Video Size
Video Jitter Fix
Slicer Column Begins
Kaypro Color Graphics Review
48 pages

ISSUE #15 (12/83)
Screen Dump Listing
Fixing Serial Ports
Playing Adventure
SBASIC Column Begins
Upgrading Kaypro II To 4
Upgrading Kaypro 4 To 8
48 pages

ISSUE #16 (2/84)
Xerox 820 Column Restarts
BBI Double Density
BBII 5"/8" Interface Fix
Kaypro ZCPR Patch
Adding Joystick To Color
Graphics
Recovering Text From Memory
52 pages

ISSUE #17 (4/84)
Voice Synthesizer
820 RAM Disk
Kaypro Morse Code Interface
68000-Based System Review
Inside CP/M 86
56 pages

ISSUE #18 (6/84)
Kaypro EPROM Programmer
I/O Byte: A Primer
Kaypro Joystick
Serial To Parallel Interface
Business COBOL
60 pages

ISSUE #19 (8/84)
Adding Winchester To BBII
6 MHz On The BBI
Bulletin Boards
Track Buffering On Slicer
4 MHz For The 820-I
64 pages

ISSUE #20 (10/84)
HSC 68000 Co-Processor
DynaDisk For The BBII
Serial Printer On BBI Sans S10
Cheap & Dirty Talker For Kaypro
Extended 8" Single Density
72 pages

ISSUE #21 (12/84)
Analog To Digital Interface
Installing Turbo Pascal
Low Intensity BBI Video
Turbo Pascal, The Early Days
80 pages

ISSUE #22 (2/85)
Xerox 820-II To A Kaypro-8
Sound Generator For the
STD Bus
Reviews Of 256K
RAM Expansion
In the Public Domain Begins
88 pages

ISSUE #23 (4/85)
Automatic Disk Relogging
Interrupt Drive Serial Printer
Low Cost EPROM Eraser
Smart Video Controller
Review: MicroSphere RAM Disk
Future Tense Begins
86 pages

ISSUE #24 (6/85)
C'ing Into Turbo Pascal
8" Drives On The Kaypro
48 Lines On A BBI
68000 Versus 80x86
Soldering: The First Steps
88 pages

ISSUE #25 (8/85)
Why I Wrote A Debugger
The 32-Bit Super Chips
Programming The 32032
Modula II
RS-232C: The Interface
104 pages

ISSUE #26 (10/85)
Inside ZCPR3
Two Megabytes On DSI-32
SOG IV
The Future Of Computing
Graphics In Turbo Pascal
104 pages

ISSUE #27 (12/85)
SOLD OUT

ISSUE #28 (2/86)
Pascal Runoff Winners
Rescuing Lost Text From
Memory
Introduction To Modula-2
First Look At Amiga
Inside The PC
104 pages

ISSUE #29 (4/86)
Speeding Up Your XT
Importing Systems
From Taiwan
Prototyping In C
C Interpreters Reviewed
Benchmarking The PCs
104 pages

ISSUE #30 (6/86)
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

ISSUE #31 (8/86)
RAM Resident PC Speedup
Practical Programming In
Modula-2
Unblinking The PC's Blinkin'
Cursor
Game Theory In PROLOG
and C
104 pages

ISSUE #32 (10/86)
Public Domain 32000:
Hardware And Software
Writing A Printer Driver for
MS-DOS
Recover A Directory By
Reading & Writing Disk
Sectors
96 pages

ISSUE #33 (12/86)
SOLD OUT

ISSUE #34 (2/87)
SOLD OUT

ISSUE #35 (4/87)
SOLD OUT

ISSUE #36 (6/87)
Mouse Control
Build A Midi Interface
For Your PC
Designing A Database, Part 2
Interrupts On The PC
Hacker's View of MS-DOS
Vs 3.X
Digital To Analog Conversion,
A Designer's View
96 pages

ISSUE #37 (9/87)
Desktop Publishing On A PC
Build Your Own Hi-Res Graphics
Scanner For $6, Part 1
Designing A Database, Part 3
Controlling AC Power
From Your PC
Expanded Memory On The
PC/XT/AT
Uninterruptable Power
Supply For RAM Disks
96 pages

ISSUE #38 (11/87)
Parallel Processing
Laser Printers, Typesetters
And Page Definition
Languages
Magic In The Real World
Build A Graphics Scanner
For $6, Part 2
Writing A Resident Program
Extractor In C
96 pages

ISSUE #39 (1/88)
PC Graphics
Drawing The Mandelbrot And
Julia Sets
Desktop Graphics
Designing A PC Work-
station Board
Around the TMS-34010
96 pages

ISSUE #40 (3/88)
The Great C Issue
11 C Compilers
Writing A Simple Parser In C
C++, An Object Oriented C
Source Level Debugger For
Turbo C
96 pages

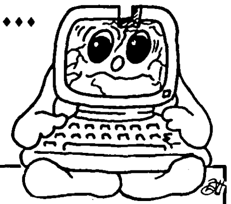
ISSUE #41 (5/88)
Artificial Intelligence
3-D Graphics
Neural Networks
Logic Of Programming
Languages
Applying Information Theory
96 pages

ISSUE # 42 (6/88)
Maintaining PCs
Keeping Your Hard Drives
Running
Troubleshooting PCs
XT Theory of Operation
Simulating A Bus
Ray Tracing
96 pages

ISSUE #43 (9/87)
Building Databases
Build a C Database
Selecting a dBase III
Compatible Compiler
Working with Paradox
Designing Custom PC Cards
Accessing dBase III Plus
Records from Turbo Pascal
96 pages

ISSUE#44(11/88)
Object-Oriented Programming
A Taste of Smalltalk
Actor
Thinking Objectively
Building MicroCad
Peripheral Technology-
PT68K-2
Hercules Graphics Printer
Dump
96 pages

♦♦♦



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.

ADVERTISERS INDEX

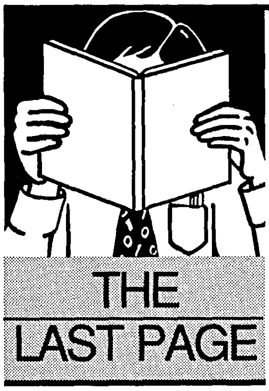
Issue 45

Reader Service		Page Number
72	Acquired Intelligence	32
107	American Cosmotron	31
132	Aristo Computers	71
04	Austin Codeworks	61
15	Cascade Electronics	53
**	CC Software	71
105	Computerized Processing Unltd.	63
07	CompuView	Inside Front Cover
09	Ecosoft	38
10	Emerald Microware	57
93	Erac Company	37
112	Garrison, Peter	15
130	GEMS	69
11	Halted Specialties	25
22	Integrand	45
**	JPAM	79
88	Koala Computers	65
17	Manx Software Systems	5
42	McTek Systems	59
**	Micro Cornucopia	74, 89,94
37	Microprocessors Unltd	72
24	microSOLUTIONS	15
02	Microsphere	13,39
120	Micro Star	29
59	National Advancement Corp	72
110	NuMega	2
03	PC Tech	Back Cover
129	Research Group	7
127	SemWare	11
19	Slicer	70
108	SofSolutions	6
40	Star-K Software	33
131	Tab Books	21
109	Quarterdeck	1
45	United Products	87
62	V Communications	76
39	Xenosoft	73
70	Zortech, Inc.	Inside Back Cover

** Contact Advertiser Directly.

Coming in Issue #46, Software Tools

- Secrets of Disassembly
- Interrupt Service Routines in C
- Modifying Sprint for the TI34010
- Designing a Neural Network in C++
- Turning an XT into a Controller



By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Factoring Monsters

There are certain mathematical facts you must know. First: factors aren't made in factories. Second: you can now get the bank's number. I mean the real number.

Whether or not we can factor very large numbers (of 100 or more digits) might not seem like a big deal. In fact, an interest in such a topic might seem down right eccentric. But to intelligence agencies, banks, and other big businesses who depend on the difficulty of factoring monster numbers, it's more than a big deal — it's their security.

A procedure called the oblivious transfer lies at the heart of the matter, and in one variation or another "secures" many of the computer data banks in the world.

Transfers of certified electronic mail and money, the signings of contracts via computers, secret exchanges of information, and numerous other deals are made and closed via the oblivious transfer, which, in effect, guarantees both parties that their messages or funds have arrived intact.

Simulating A Toss

In its simplest form, the oblivious transfer could be used for a safe coin toss.

Suppose you and your ex-partner (let's call her Eliza) live in different cities, and you want to divvy up some property. Whoever wins the toss gets first choice.

(Or if you're more sports-minded, consider two football teams playing a game on different fields and needing to toss in order to decide who kicks off.)

In a coin-toss simulation, we assume that the two tossers distrust each other. Otherwise, one could toss, the other could call, and both would accept the result.

Here's how a toss might work —

Eliza starts by selecting 2 primes: say, 7 and 13. (Note: in a real-world situation, these primes will be very large, say 60-80 digits.)

She multiplies them and sends the product (91) to you, keeping her numbers (7 and 13) secret.

You win the toss if you can factor 91. First, you make sure Eliza isn't cheating by checking to see if the number is a prime (let's assume you have a little program that does this; i.e., the program checks for primes).

Now, you randomly (or by some mysterious formula) pick a number between 1 and 91, say 11. You square it, get 121, divide it (using modulo division) by 91, and get the remainder (30). You send 30 to Eliza, keeping your number (11) secret.

Eliza knows the original number (91), so she looks for all numbers less than 91 that generate a remainder of 30 when divided into her number.

She discovers two pairs (±11 & ±24). If she sends 11, she wins the toss because you won't have received any new information, and thus won't be able to factor her number. If she sends 24, you'll add 11 (your number) to 24, and get 35. The greatest common divisor of 35 and 91 is 7, so you can now factor Eliza's number. You win the toss.

Critical Details

In the high-tech world, the oblivious transfer depends on two crucial mathematical details —

- that we can determine whether a large number (of 60 digits or more) is a prime;
- and that factoring large numbers is hard.

Determining whether a large number is prime (divisible only by itself and by the number 1) is more or less trivial, even for large numbers.

Factoring large numbers is hard. You can prove it to yourself by using "trial division" or "baby divide" on a small number (N). Try dividing N by 2, then 3, then 5, then 7, etc., until one of them divides cleanly. You only have to try primes (to save time) and you don't have to try any prime greater than the square root of N. Sounds easy? It is. But only for a small N.

As N reaches 50, 60, or more digits, the computing time becomes astronomical. It takes a Cray a day or so to crack an 80 digit number. And until just recently cracking a 100 digit number had not been done. The record was 96 digits — a record banks counted on.

Enter Arjen Lenstra, a computer scientist at the University of Chicago, and his cohorts. He organized (and linked) a team of computers in the U.S., Europe, and Australia, from PCs and workstations to a Cray, and cracked the first 100 digit number (into factors 41 and 60 digits long) in 26 days.

They did it by using a technique called distributed processing. They split the problem into little bits and processed it in parallel. Then they compiled the results on minicomputers at a Digital Equipment lab in Palo Alto, California.

A Cray operating alone would have taken at least two months to factor a 100 digit number.

The number, by the way, is —
9,412,343,607,359,262,946,971,172,136,
294,514,357,528,981,378,983,082,541,
347,532,211,942,640,121,301,590,698,
634,089,611,468,911,681.

Its factors are —
86,759,222,313,428,390,812,218,077,095,
850,708,048,977
and
108,488,104,853,637,470,612,961,399,842,
972,948,409,834,611,525,790,577,216,753

The so-called security factor (the impossibility of factoring monsters) banks, etc., have counted on seems to no longer exist. Anyone with a better idea for securing all that loot could make a mint.

References

Peterson, Ivar., *The Mathematical Tourist*. 1988. Freeman & Co. ♦ ♦ ♦

LAST CHANCE AT

$99.95

Expires Feb. 1st
Regular price $149.95

C++

FROM ZORTECH INC.

WALTER BRIGHT – Programmer

"This is a revolutionary product which reflects the fact that Walter Bright is one of the world's best programmers."
Andy Redfern, Personal Computer World

"I find it amazing that Walter continues to find ways to improve his already excellent product."
Scott Robert Ladd, Micro Cornucopia

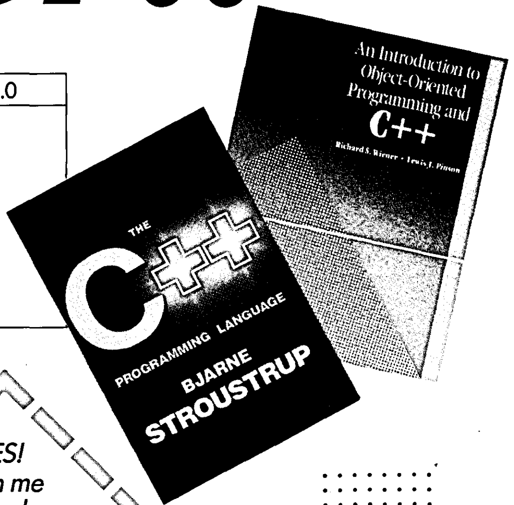
C is great! C++ is better!!

Performance benchmarks

Benchmark	Zortech C	Zortech C++	Turbo C 1.5	Quick C 1.0
Sieve	20.49	20.54	23.62	22.72
Rsieve	20.49	20.54	23.62	22.03
Integer'	1.32	1.38	6.31	6.49
Float	0.17	0.22	52.29	51.03
*Float	32.73	37.74	52.39	51.63
Pointer	17.91	17.96	17.13	16.87
Rpointer	17.79	17.91	17.14	16.64
Loop	3.90	3.90	3.90	3.90
Optimize	0.49	0.60	8.46	8.79

Benchmarks were run on an 80286 based IBM compatible at 6Mhz with no 8087.

*The float benchmark was re-run without optimization.


An Introduction to Object-Oriented Programming and C++
Richard S. Wiener · Lewis J. Pinson
THE C++ PROGRAMMING LANGUAGE
BJARNE STROUSTRUP

Zortech C++ is the first native-code C++ compiler for MS-DOS compatible machines. This means you don't need to spend a lot of money on a C compiler. In fact, Zortech C++ comes with a C++ compiler, a C compiler, a linker, a librarian, a fully integrated editing environment, context sensitive help and the fastest graphics library you've ever seen!

With Zortech C++, you can mix and match code in C++, C and assembler and link them all together with a single command!

Zortech C++ comes with a complete C++ and C library, including compatibility with the ANSI C draft. Unique to Zortech C++ is support for mice, fast screen writes, re-entrant floating point and speaker support.

Now you can move your Microsoft and Turbo C code over to Zortech C++ – we've included compatible library functions! Also, we have made Zortech C++ compatible with Microsoft's Codeview debugger!

Enter the world of Object Oriented Programming (oops) with Zortech C++!

Sure, C is great – but C++ is better!

*YES!
Rush me
C++ as shown
below:*

*□ Zortech C++
$99.95**

□ C++ Book $29.95 □ Intro C++ Book $27.95**

VISA/MC/COD CHECK ACCEPTED

Name..........

Address..........

.... Phone..........

VISA or MC#:.......... Exp. date..........

ZORTECH

To: ZORTECH INC. 366 Massachusetts Ave., Arlington, MA 02174. Tel: 617-646-6703. Fax: 617-643-7969.

CALL THE ORDER HOTLINE 1-800-848-8408

**Prices do not include shipping*

Reader Service Number 70

MI-2

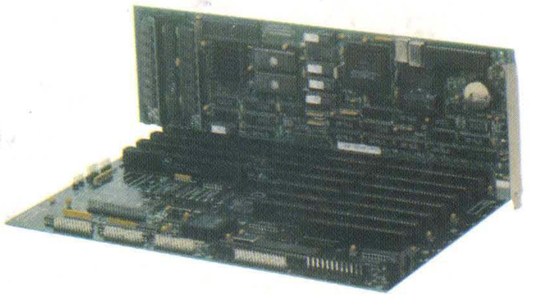
VERY HIGH PERFORMANCE

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 4 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.



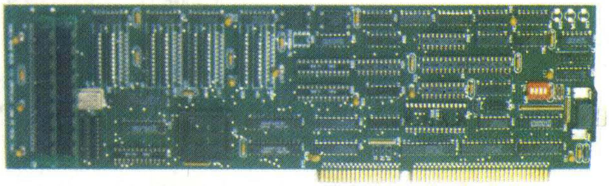
PC Tech X24 and ASMB

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 tracer/debugger. Our standard product includes support for extended CGA, MDA and Hercules emulation 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

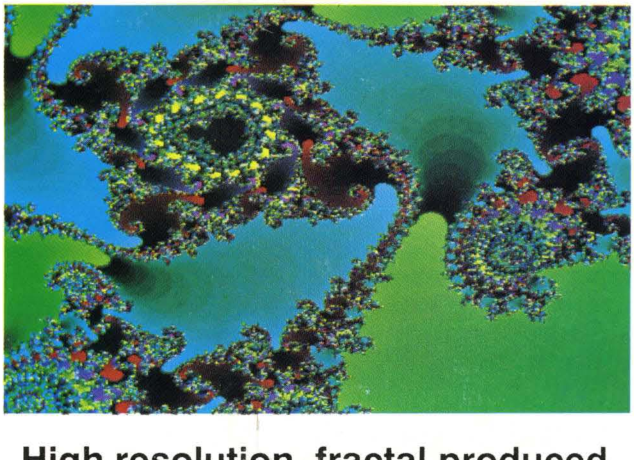
PC Tech will license most products for non-exclusive manufacture. We will also customize any of our designs to better meet your needs on our in-house CAD systems. All of our standard products are available in private label versions.

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)



High resolution fractal produced on the PC Tech COLOR 34010

Reader Service Number 3

PC, XT, AT, DGIS, Hercules, and Award BIOS are trademarks or registered trademarks of their respective companies.