

MICRO CORNUCOPIA

FINAL ISSUE

Micro Controllers

In this, our hysteric final issue, we cover one of our favorite topics, micro controllers and their cousins, the data loggers.

Designing A Micro Controller page 8

Designing a controller is simple when you start out with the right micro.

Building A Two-chip Terminal page 26

The perfect accessory for your micro controller.

Waylaid By Snakes page 20

A low-down look at objects (in the grass).

Save The Floppies page 54

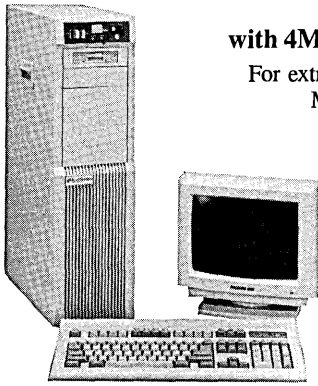
Routine, complete with code, for recovering bad sectors on floppies.

And More . . .

- Shareware/Favorite Tools
- Initializing Variables in Turbo Pascal
- Faster Neural Networks



GREAT VALUES FROM MICROSPHERE!



SUPER 25Mhz 386 SYSTEM! with 4MB RAM and Rotary Voice Coil Hard Drive

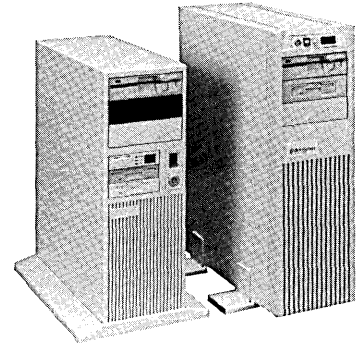
For extra speed and reliability we've included a 66MB Mitsubishi MR535 voice coil hard drive with an access time of 28ms, and a super fast 1:1 interleave HD controller card.

- 25Mhz 80386 CPU, AMI Bios, Full Size MB
- 4MB of RAM, expands to 8MB on MB
- 1.2MB and 1.44MB Floppy Drives
- 12" Amber Monitor w/Mono Graphics Card
- 101-Key Enhanced Keyboard, Focus or Maxiswitch
- 2 Serial, 1 Parallel, 1 Game, Clock/Calendar
- 200-Watt Power Supply
- Socket for 80287, 80387 or Weitek
- Full 1 Year Warranty

\$2350 (Mega Tower)

w/Std AT Case.....\$2250
w/1MB.....\$250 Less
20Mhz 386.....\$100 Less

Tower Cases w/Power Supply



Mini Tower \$225
Standard Tower \$269

AT /386S SYSTEMS

Includes: 1MB RAM, 1.2MB & 1.44MB FD, 40MB Hard Drive (28ms), Fast Hard Drive Controller, Mono Graphics Video Card, 12" Amber Monitor, 101 Key Keyboard, Serial(2)/Parallel(1)/Game(1) Ports, Clock/Calendar, Full 1 Year Warranty. **FREE** assembly and testing.

6/10Mhz 286/AT.....\$1249
8/12Mhz 286/AT.....1295
8/16Mhz 286/AT.....1395
16Mhz 386-SX.....1449

Video upgrades for our systems (includes video card and monitor)

ATI Graphics Solution.....39
CGA Color.....175
EGA Color.....349
VGA Color (Analog, 16 bit w/Amazing monitor).....379
CGA/EGA/VGA (Multisync).....450

FLOPPY DRIVES

360K.....69
1.2MB.....77
3 1/2" 720K.....79
3 1/2" 1.44MB.....85

HARD DRIVES

XT 20MB Miniscribe/Kalok
8425 (65ms).....229
8425 w/controller.....269
XT 30MB Miniscribe/Kalok
8438 (65ms).....229
8438 w/controller.....269
AT 44MB Mitsubishi MR535,
Rotary Voice Coil (28ms).....429
AT 71MB MFM
Voice Coil (28ms).....595

VIDEO CARDS

Mono Graphics w/Parallel port.....37
ATI Graphics Solution
Herc w/CGA Emulation.....79
CGA/EGA/VGA 8-Bit.....119
CGA/EGA/VGA 16-Bit.....129

The Most Cost Effective Way to Speed Up your AT or 386 System!

Upgrade your hard drive controller. Discover the NEXT generation of hard drive controller cards. As the chart below demonstrates, the speed improvements are incredible!

Controller Card Type	Data Xfer Rate
Standard IBM AT, 3:1 Interleave, MFM	167 KB/Sec.
DTK WA2, 2:1 Interleave, MFM	261 KB/Sec.
AT/386 1:1 Interleave, MFM	522 KB/Sec.
Western Digital 1:1, RLL, w/Cache	799 Kb/Sec.

Note: Test results using 10Mhz AT, Mitsubishi MR535 Hard Drive and SpinRite Disk Optimizer.

VGA Color ONLY \$379...



When you buy a complete MicroSphere Computer System. Upgrade includes: 16 bit VGA video card and 14" Analog VGA Monitor with a .28 mm dot pitch.

MOTHERBOARDS

XT/Turbo 4.77/10Mhz.....\$75
AT 6/10Mhz.....175
AT 8/12Mhz.....195
AT 8/16Mhz.....249
386SX 16Mhz w/C&T chip set.....359
386 8/20Mhz w/Phoenix Bios,
holds up to 8MB on board.....679
386 16/25Mhz w/AMI Bios
holds up to 8MB on board.....795
386 33Mhz.....Call

POWER SUPPLIES

150-Watt XT Power Supply.....49
200-Watt AT Power Supply.....59
230 Watt AT Power Supply.....75

EXPANSION CARDS

Clock.....18
Game (Joystick).....14
Parallel Port (LPT1, 2 or 3).....18
Serial Port, 2 ports, 1 installed,
(COM1 or 2).....18
2nd Serial Port Kit.....18
Serial Port, 4 ports installed.....99
Multi Drive Controller, up to
2 drives, Supports 360K,
720K, 1.2MB & 1.44MB.....39
3 & 4th Floppy Controller.....79

AT/386

AT Multi-IO, 2 Serial, 1 Parallel,
1 Game.....35
AT 3MB EMS Memory Card (0K).....99

CHIPS/ETC.

Memory 64K, 256K, 1MB.....Call
Math Coprocessors.....Call
XT Components & Systems.....Call

CONTROLLER CARDS

AT/386 1:1 MFM Controller.....\$105
WD 1006-SR2 1:1 RLL w/Cache....139

CABINETS

XT Slide Case, Lock, LED.....35
AT Slide Case, Lock, LED
3 half ht., 1 full ht. drives.....59
Baby Tower, w/200 watt PS,
2 half ht., 2 - 3 1/2" Drives,
Baby Motherboard ONLY.....129
Mini Tower, w/200 Watt PS,
3 half ht, 2 - 3 1/2" Drives,
holds Full or Baby size MB.....195
5 Bay Std. Tower, 220 Watt PS,
3 half ht., 1 full ht drives.....269
6 Bay Mega Tower, 230 Watt PS.....229

MONITORS

12" Amber Monochrome TTL.....88
CGA Color RGB.....249
EGA/CGA Autoswitch.....362
CGA/EGA/VGA
Multisync (.31 dot).....489
VGA Analog
(Amazing .28 dot).....369

KEYBOARDS

Maxiswitch (AT Only) 101-Key.....\$59
Keytronic* Enhanced 101-Key.....55
Focus* 101-Key Tactile, Switchable
Control/Caps Lock, Dust Cover.....59
(#1 Find by Micro C Staff)
* Keyboards are XT/AT switchable

SOFTWARE

MS-DOS 3.3 w/GW Basic.....95
DR-DOS 3.3 w/GEM.....45
SpinRite II Disk Optimizer
by Gibson Research.....79
386Max Memory Manager for
386 Systems.....69

MicroSphere INC.
COMPUTERS "HARDWARE MANUFACTURER SINCE 1983"

Orders Only Please! **1-800-234-8086**

Tech Calls: **(503) 388-1194** Hours: Mon-Fri 9:00-5:30
855 N.W. WALL • BEND, OREGON 97701

*Prices are subject to change without notice. Shipping CHARGES will be added. *1-year warranty/30-day money back (subject to restrictions)

ZORTECH

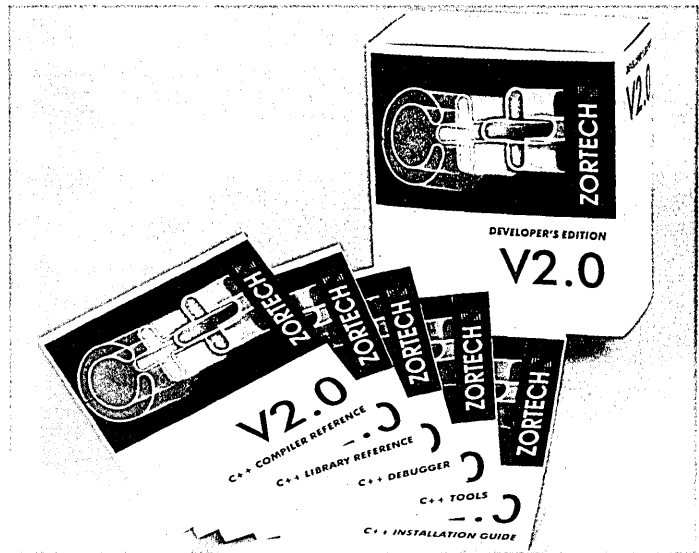
NEW! AT&T C++ RELEASE 2.0 SPECIFICATION

NEW! MS WINDOWS COMPATIBILITY

NEW! EASY PORTABILITY FROM MICROSOFT C

NEW! C++ DEBUGGER & EXPANDED C++ TOOLS

NEW! OS/2 UPGRADE AVAILABLE NOW!



We listened carefully to what you wanted in a next generation MS DOS C++ compiler. The answer is Zortech C++ V2.0 Developer's Edition.

You wanted the latest AT&T V2.0 features with the power offered by multiple inheritance and type safe linkage, so here it is.

You wanted compatibility with MS WINDOWS, we added it.

You repeatedly asked for easier portability from Microsoft C, we got the message, and have written the library functions you need.

You wanted the world's first MS DOS C++ source level

DEBUGGER, and now the wait is over.

You wanted expanded and improved documentation, we both listened and delivered.

You wanted to be able to upgrade to an

OS/2 version compiler supporting Presentation Manager, you did not want it to cost a fortune, so it's available for \$150.

You want to look at the standard library SOURCE CODE, so we are including it.

SAVE \$200
Get the Developer's Edition for only \$450 (compared to \$650) comprising:

C++ Compiler	\$199.95
C++ Debugger	\$149.95
C++ Tools	\$149.95
Library Source	\$149.95
Total Value	\$649.80

Here is our list of highly recommended C++ books:

C++ Language/Stroustrup	\$32.25
C++ Answer Book/Hansen	\$26.95
C++ for C Programmers	\$29.95
C++ Primer/Lippman	\$30.25

Ask about our new C++ Video Tutorial

For many, EMS programming support, built into the compiler is important, so it's in there too.

You were happy using the 18 classes provided in C++ TOOLS, but we revised and expanded it anyway.

You never asked for a free TSR library to be included, but we knew you'd love to use our neat little package, so we included it free.

You liked our FLASH GRAPHICS package for its speed, but wanted a C++ Class interface, so we've written it.

How To Order:

Already own Zortech C++? Call the order hotline for details of our low cost upgrades.

To order Zortech C++ for the first time, just call the order hotline. We accept payment by Mastercard/Visa/COD.

Alternatively, mail the coupon below with your check or credit card details.

ZORTECH INC.,
1165 Massachusetts Avenue, Arlington, MA 02174, USA
Voice 617-646-6703 Fax 617-643-7969

ZORTECH LTD.,
106-108 Powis Street, London, SE18 6LU, ENGLAND.
Voice (44)-1-316-7777 Fax (44)-1-316-4138

CALL 1-800-848-8408

Yes! Please rush me the following C++ V2.0 items:

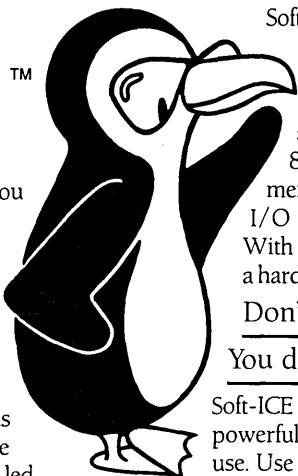
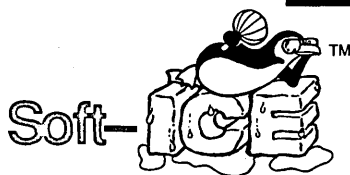
Name _____
Address _____
City _____ State _____ Zip _____
Visa/MC# _____
Exp. Date _____ Tel _____

- | | |
|---|---|
| <input type="checkbox"/> DEVELOPER'S EDITION \$450 (Save \$200) | <input type="checkbox"/> OS/2 COMPILER UPGRADE \$149.95 |
| <input type="checkbox"/> C++ COMPILER \$199.95 | <input type="checkbox"/> C++ VIDEO COURSE \$499.95 |
| <input type="checkbox"/> C++ DEBUGGER \$149.95 | <input type="checkbox"/> C++ Language /Stroustrup \$32.25 |
| <input type="checkbox"/> C++ TOOLS \$149.95 | <input type="checkbox"/> C++ Answer Book/Hansen \$26.95 |
| <input type="checkbox"/> LIBRARY SOURCE CODE \$149.95 | <input type="checkbox"/> C++ for C Programmers/Phal \$29.95 |
| <input type="checkbox"/> COMPILER & LIBRARY SOURCE \$299.95 | <input type="checkbox"/> C++ Primer/Lippman \$30.25 |
- For US orders please add \$5.05 shipping Overseas orders at international mail rates.

All MicroSoft trademarks are acknowledged.

FINALLY. A debugging tool tough enough to handle the DOS Nasties.

New Version 2.0



Nasty over-write? No sweat!

Soft-ICE memory range break points help you track down memory over-write problems whether you are doing the over-writing or another program is over-writing you.

Hung program? No problem!

When the system hangs, you now have hope. With Soft-ICE you can break out of hung programs no matter how bad the system has been trashed. And with Soft-ICE's back trace ranges you can re-play the instructions that led up to the crash.

Program too large? Not with Soft-ICE!

Soft-ICE runs entirely in extended memory. This means you can debug even the largest DOS programs. And since your program runs at the same address whether Soft-ICE is loaded or not you can find those subtle bugs that change when the starting address of your code changes.

System debugging? Soft-ICE is a natural!

Soft-ICE is ideal for full source level debugging of TSRs, interrupt service routines, self booting programs, DOS loadable device drivers, real-time kernels, non-DOS O/Ss and ROMs. Soft-ICE can even debug within DOS & BIOS.

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 the 80386 to provide real-time break points on memory locations, memory ranges, execution, I/O ports, hardware & software interrupts. With Soft-ICE you get all the speed and power of a hardware-assisted debugger at a software price.

Don't want to switch debuggers?

You don't have to!

Soft-ICE can run stand-alone or it can add its powerful break points to the debugger you already use. 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 automatically.

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.

"These may be the only two products I've seen in the last two or three years that exceeded my wildest expectations for power, compatibility and ease-of-use."

— Paul Mace
Paul Mace Software

Soft-ICE	\$386
MagicCV	\$199
MagicCV for Windows	\$199
Buy Soft-ICE & MagicCV(W)	—Save \$86.
Buy MagicCV and MagicCVW	—Save \$100.
Buy All 3	—Save \$186.

30 day money-back guarantee
Visa, MasterCard and
AmEx accepted



New Soft-ICE 2.0 features

- Back Trace Ranges
- Symbolic & Source level debugging
- EMS 4.0 support with special EMS debugging commands
- Windowed user interface



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

RUN CODEVIEW IN 8K

MagicCV



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

NEW VERSION FOR C 6.0
Attention Windows Developers!
Version available for CVW.

P.O. BOX 7607 ■ NASHUA, NH ■ 03060-7607

MICRO CORNUCOPIA

MAY 1990—ISSUE NO. 53

FEATURES

8 H. Ward Silver
Designing A Microcontroller
Ward puts together a very simple data logger and microcontroller based on the 68HC11.

20 Bruce Eckel
Waylaid By Snakes
A fun look at C++ if you don't object to serpents.

26 Karl Lunt
Building A Two-chip Terminal
Here's a simple, low-power terminal with a two-line by twenty-column display.

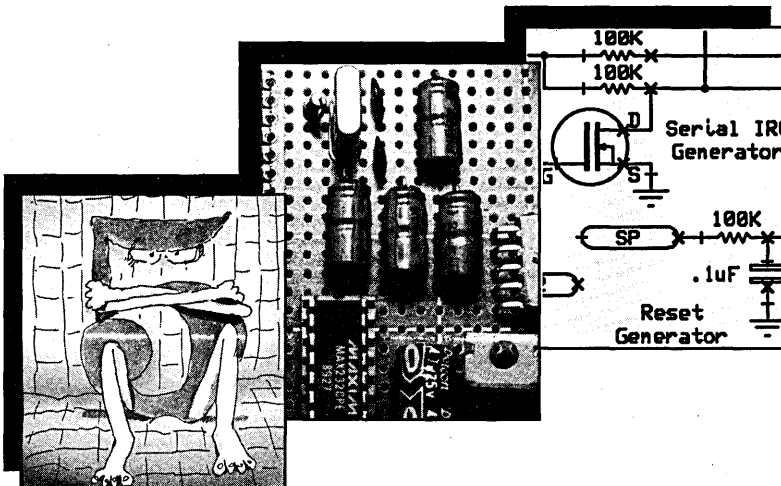
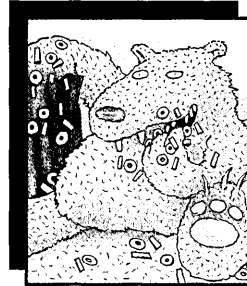
34 Lance Dannon Bresee
A Roundoff Roundup
Tired of round-off errors? This should help your calculations.

40 Dave Gwillim
Initializing Variables In Turbo Pascal
An elegant fix for a popular compiler.

46 D. Gilbert Lee
Fast Neural Networks

54 Larry Fogg
Save The Floppies

70 Stuart R. Bell
Build A Composite Video Adaptor



COLUMNS

57 86 World

64 Culture Corner

65 On Your Own

81 Units and Modules

88 Shareware

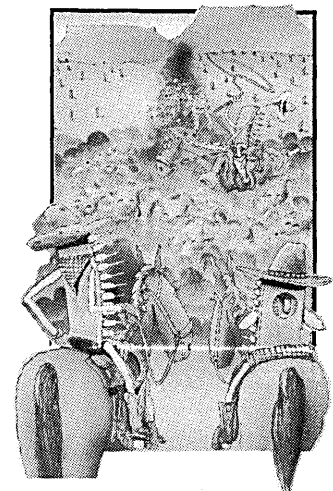
90 Techtips

FUTURE TENSE

85 Tidbits

96 Last Page

COVER



Cover Illustration by Greg Cross.

Editor and Publisher
David J. Thompson


Associate Editors
Gary Entsminger
Larry Fogg
Cary Gatten

Contributing Writers
Bruce Eckel
Michael S. Hunt
Alfred E. Newman
Karl Lunt

Accounting
Sandy Thompson

Advertising & Distribution
Reader Services
Nancy Ellen Locke & Laura Shaw

Graphic Design & Production
Carol Steffy

MICRO CORNUCOPIA (ISSN 0747-587X) was 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.  Recycled Paper

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 (surface)	\$36.00
1 yr. Foreign (airmail)	\$50.00

Make orders payable in U.S. funds on a U.S. bank.

CHANGE OF ADDRESS:

Please send your old label and new address to:

MICRO CORNUCOPIA

P.O. Box 223
Bend, Oregon 97709

READER SERVICES:

For orders and subscription problems call (503) 382-5060, 9 am to 5 pm, Pacific time, M-F. FAX your VISA or MC order to us, our FAX number is (503) 389-6833.

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

Copyright 1990 by Micro Cornucopia, Inc.
All rights reserved
ISSN 0747-587X

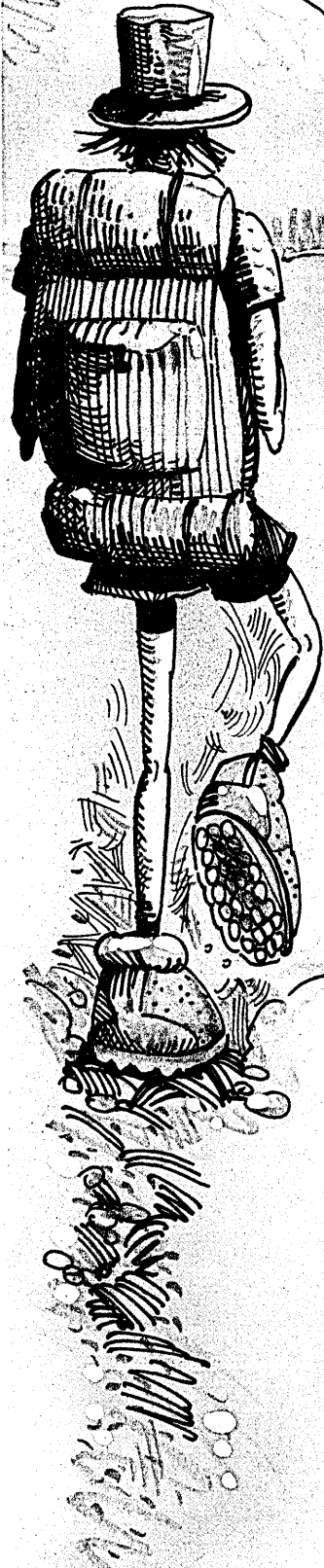


The
Audit
Bureau

AROUND
THE BEND

By David J. Thompson

The Last Hoorah



I'm not sure how to begin. I've started this editorial so many times, in my daydreams, in my sleep, in the shower.... But nothing I've come up with really feels like I feel.

I'm closing down *Micro C* and I don't know what I'll be doing next. (Wait till mom reads this.... "Dad! Our little David's quitting his job. Quick, clean up the spare room.")

You may be wondering why it's closing rather than moving on to other ownership. Good question. Three groups have come forward, interested in taking on *Micro C*. After long discussions, none felt they could put together both the finances and the talent to make it fly.

When a magazine folds (a strangely appropriate word), other magazines often pick up the subscribers. As far as I know, the only subscribers no one wanted were the ones taking *Profiles* (aimed at Kaypro owners) and *CP/M Review*. So we've had offers from several magazines wanting to finish your subscriptions. How did I choose?

Some were willing to pay for your names. Some weren't. Some were already popular with *Micro C* subscribers. Some weren't. Some were very interested in picking up the articles already in the pipe for future issues. One even wanted to continue "Around The Bend" in its entirety. (Led to an, uhm, lengthy discussion about sanity.)

After considering all the options, I selected *Computer Language*. To finish out your *Micro C* subscription, you'll get your choice of *Computer Language*, *AI Expert*, or *Embedded Systems Programming*. If you already subscribe (25% of you already take *Computer Language*), they'll extend your subscription. Just fill out the form you'll receive in April.

If you've got a year of *Micro C* coming, you'll get a year of *Computer Language*—and you'll be getting it at a really good discount (CL's normally \$29 a year, *Micro C*'s just \$18). Plus, *Computer Language*'s standard policy is: if you decide you don't like their magazine, they'll send you a refund for all remaining issues. All you have to do is let them know.

Look, I know that *Computer Language* isn't *Micro C*. When I looked closely the past few months, I was surprised how far we'd moved away from the other com-

Continued on page 74

Don't Buy Anybody's Disassembler Until You Read This.

Trying to modify a program without its source code is tough enough without having to use an inferior disassembler.

That's why you should look at DASM™ before you invest in any disassembler product.

Most disassemblers complicate the process with errors: giving you incorrectly placed labels, incorrect and misplaced code and data, data treated as code, or code treated as data.

DASM™ makes the modification process easy by keeping track of segment register usage and mapping code in its execution sequence. It determines which parts of the file are code, which are data, and what labels exist. You simply edit the assembly language output, make the necessary changes to the program, and reassemble.

Japan's #1 Seller - Made in America

DASM's simplicity, accuracy and power have made it the top-selling disassembler in Japan for two straight years. The reason - DASM™ does what other disassemblers can't:

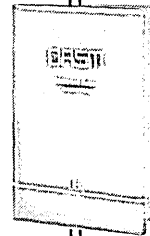
- Handle problematic segment and instruction address register loads through interactive mode
- Track segment register usage
- Generate appropriate ASSUME's and segment maps
- Handle multiple entry points
- Track memory references
 - Direct
 - Via register
 - Implied in OS function requests
- Handle transfer vectors
- Handle .EXE, .COM and .BIN files up to 200K

Try DASM™ Risk-Free!

DASM™ is available through the Programmer's Shop for only \$225. And it's guaranteed to outperform any disassembler on the market.

Order your copy now and throw your worst disassembly problem at it. We think you'll be impressed. If not, just return DASM™ for a full refund.

Why settle for anything less?



Jb

Software

To Order Call

301-752-1348

For more information, call (301) 752-1348.

To demonstrate DASM's superior performance, we gave the same disassembling task to both DASM™ and Sourcer™ (Version 2.18).

Compare For Yourself

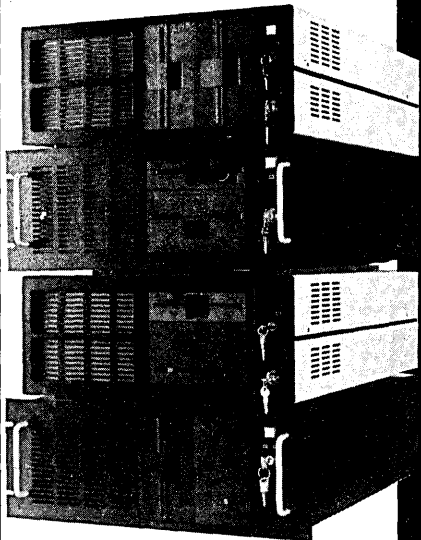
While DASM™ generated the correct assembly code, Sourcer made at least six major errors.

DETAIL	SOURCER™	DASM™
<p>① Sourcer™ has incorrectly Assumed DS to the segment address of the code. (DS is actually pointing to the PSP on entry.)</p> <p>② Sourcer's error at this point renders all labels generated relative to the DS incorrect.</p> <p>③ Sourcer™ continues to have the incorrect Assume for the DS, even after being loaded with a segment value referenced in the program header.</p> <p>④ Sourcer™ does not reference the segment value symbolically, so any changes to the program which alter its location will not be reflected.</p> <p>⑤ Since Sourcer™ generates no segment labels relative to the DS segment, changes in the DS segment will not be reflected.</p> <p>⑥ Sourcer™ can't handle jumps or calls via register, and does not generate the code segment label (or reference).</p> <p>In short, if you're not using DASM™, be prepared to spend a long time editing the file to add and fix the missing and incorrect labels - and expect a lot of bugs!</p>	<pre> data_1e equ 0 data_3e equ 50h seg_a segment para public assume cs:seg_a, ds:seg_a data_4 dw 6F15h loc_1: mov dx, data_1e mov ah, 9 int 21h mov al, byte ptr ds:data_1e+15h mov bx, 11h jmp bx db 0B4h, 4Ch, 0CDh, 21h t2 proc far start: mov cl, ds:data_3e mov ds, cs:data_4 jmp short loc_1 t2 endp seg_a ends seg_b segment para public assume cs:seg_b, ds:seg_b db 'Print string message\$' db 'P' seg_b ends end start </pre> <p><i>Handwritten notes:</i> - "not a label" with arrow to 50h - "wrong" with arrow to 6F15h - "DS changed not indicated by Sourcer" with arrow to ds:seg_b - "no labels" with arrow to seg_a</p>	<pre> D2SEG SEGMENT at 0FFF0H org 50H D2B50H LABEL BYTE D2SEG ENDS C1SEG SEGMENT ASSUME cs:C1SEG, ds:D2SEG, es:D2SEG C1W0H LABEL WORD dw SEG D3SEG ASSUME ds:D3SEG C1N2H: mov dx, OFFSET D3B0H mov ah, 9 int 21H mov a1, D3B15H mov bx, OFFSET C1B11H jmp bx C1N11H: C1B11H LABEL BYTE mov ah, 76 int 21H ASSUME ds:D2SEG C1F15H LABEL FAR mov c1, D2B50H mov ds, C1W0H ASSUME ds:D3SEG jmp C1N2H C1SEG ENDS D3SEG SEGMENT D3B0H LABEL BYTE db 'Print string message\$' D3B15H LABEL BYTE db 'P' org 20H D3SEG ENDS END C1F15H </pre> <p><i>Handwritten annotations:</i> - Circled numbers 1-6 corresponding to the detail text. - Arrows pointing from detail text to specific lines in the DASM output.</p>

Trademark/Owner: DASM™/JBS Software;
Sourcer™/V Communications.

Rack & Desk PC/AT Chassis

Integrand's new Chassis/System is not another IBM mechanical and electrical clone. An entirely fresh packaging design approach has been taken using modular construction. At present, over 40 optional stock modules allow you to customize our standard chassis to nearly any requirement. Integrand offers high quality, advanced design hardware along with applications and technical support *all at prices competitive with imports. Why settle for less?*



Rack & Desk Models

Accepts PC, XT, AT Motherboards and Passive Backplanes

Doesn't Look Like IBM

Rugged, Modular Construction

Excellent Air Flow & Cooling

Optional Card Cage Fan

Designed to meet FCC

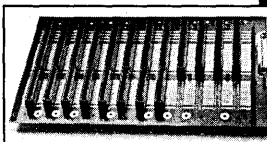
204 Watt Supply, UL Recognized

145W & 85W also available

Reasonably Priced

Now Available

Passive Backplanes



INTEGRAND

RESEARCH CORP.

Call or write for descriptive brochure and prices:
8620 Roosevelt Ave. • Visalia, CA 93291

209/651-1203

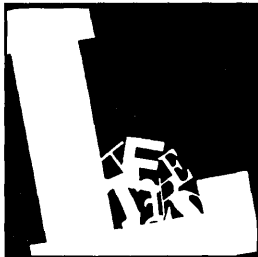
TELEX 5106012830 (INTEGRAND UD)

FAX 209/651-1353

We accept Bank Americard/VISA and MasterCard

IBM, PC, XT, AT trademarks of International Business Machines.
Drives and computer boards not included.

Reader Service Number 22



Letters

So Long

Today I heard that *Micro Cornucopia* is to be no more. This greatly saddens me even as I recognize the need for change in all things. Your magazine filled a niche in the journalistic world that was very special and unique. You represented all of us plain folks, who did not hide behind the shirt and tie but who loved working with computers at every level.

You spoke to us, as one enthusiast to another, not using technojargon to hide a lack of real understanding so common in other journals. You held gatherings where we laughed and joked with the literary mentors who taught us and guided us on our own paths to knowledge, even while we wiped the barbecue sauce off our lips.

You became family. You will be missed. My best wishes and thoughts go out to each and every one of you at *Micro C* who have touched my life. Godspeed And Thank You.

Al J. Szymanski (Big Al)
8991 Edcliff Ct. SE
Aumsville, OR 97325

Editor's note: Thanks Al. The entire staff remembers, only too vividly, your smashing presence during SOG volleyball. Godspeed to you, also.

Bend Memories

I have just sent in my subscription renewal. Late. I can't believe that just because I forgot to renew, you stopped sending me the magazine! Didn't you know that I still wanted it?

Seriously, I think *Micro C* is the best computer mag available. You can imagine my shock when I first saw it a couple years ago. I couldn't believe such a great mag was published in Bend, of



all places. I mean, I used to live in Bend when I was 14, in the year 19-mumble-mumble. I don't suppose Bend has changed any since then? As I recall, Bend was very small.

I say, Bend was very small. (Did you say, "How small was it?") It was so small that if you wanted a business phone number, you looked it up on the yellow page. It was so small that every time I turned on my radio, the electric trolley slowed down. It was so small we didn't have a regular village idiot; we all took turns. Bend had a newspaper, though. ("Really?") Yeah, a traveling salesman left it at the feed store.

I may be exaggerating slightly, but the most interesting thing I did all year there was ride my bike up the dead volcano outside town. I had just seen *The Last Days of Pompeii* and, being 14, I had hopes it might erupt again and liven things up a bit. No such luck.

Sounds like I missed something by not attending the Rocky Mt. SOG. But I had been to Gunnison before, and having recently seen *Deliverance*, I

Letters continued on page 68

Good Luck!

*Dave & Sandy
Jennifer & Erin*

The \$25 Network

Try the 1st truly low cost LAN

- Connect 2 or 3 PCs, XTs, ATs
- Uses serial ports and 5 wire cable
- Runs at 115 K baud
- Runs in background, totally transparent
- Share any device, any file
- Needs only 14K of ram

Skeptical? We make believers!

Over 15,000 sold worldwide



Information Modes

P.O. Drawer F

Denton, TX 76202

817-387-3339 or 1-800-628-7992

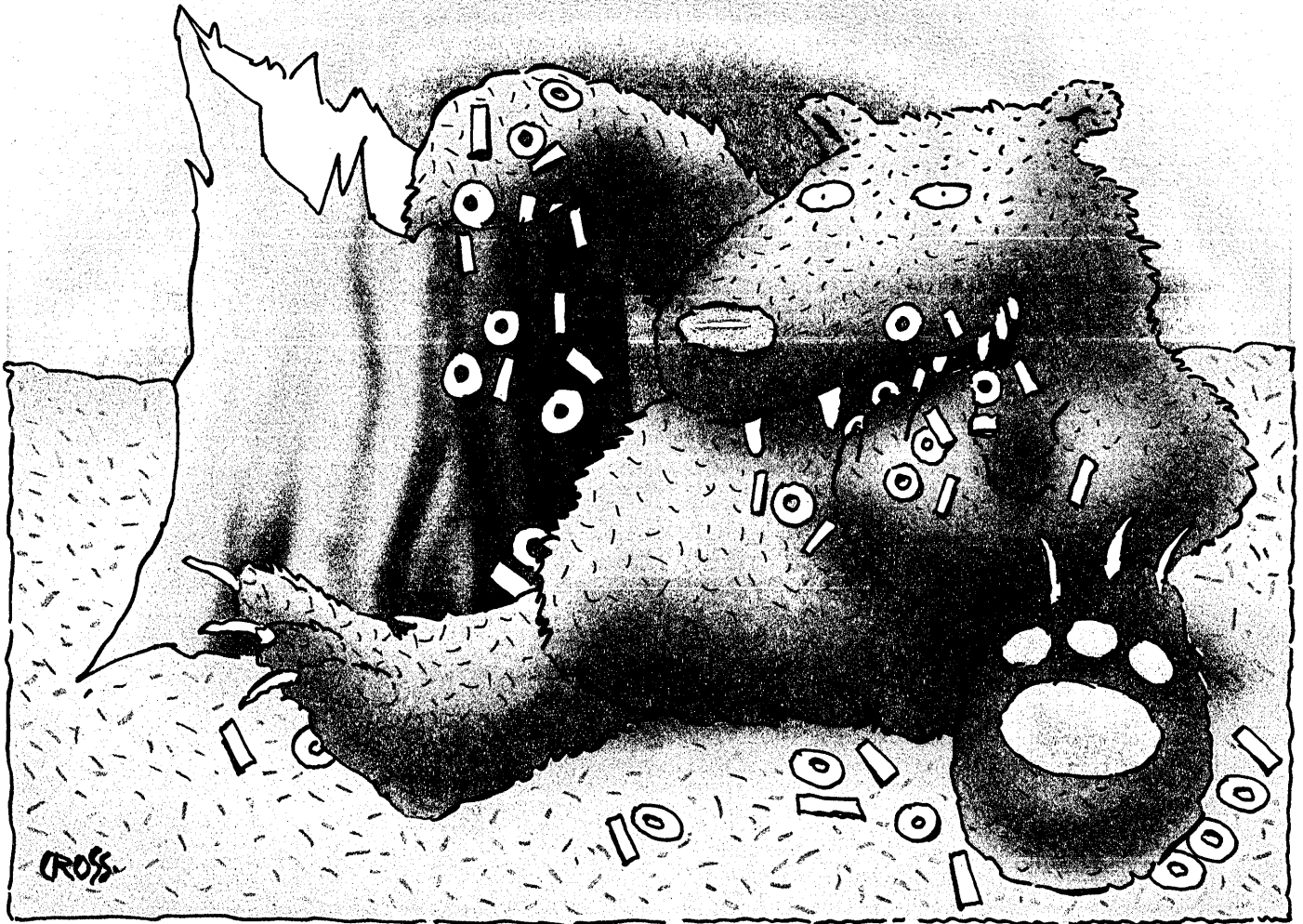
Why waste money on simple file transfer systems?

Love,

Don & Kim

Donald & Courtland

Reader Service Number 149



Designing A MicroController

The MC68HC11 In Action

Ward walks us through a microcontroller design project based on the 6811. If you're thinking about designing a monitor to watch the real world or control a process this project, the DataBear, should make great reading. (Now that I've spilled the name of his board, you suspect I'm going to make one of my unbearable puns. Right?)

The best way to describe a particular microprocessor is to show how it's used. In this article, I'm using the Motorola MC68HC11 in a battery-powered data logger, the DataBear from Langan Products, Inc., of San Francisco.

The 6811 is one of the more flexible and powerful eight-bit microcontrollers.

See Figure 1 for a list of features and Figure 2 for the way it's organized.

Not only is the 6811 well supported by software and hardware development tools (including C compilers and emulators), it's quite cheap. Unlike some of the latest whiz-bang processors, a 52-pin version with 512 bytes of EEPROM costs only about \$12 in onesies. Plus, you can easily interface inexpensive 8-bit support chips to the 6811.

See Figure 3 for a list of variations on the 6811.

What Is A DataBear?

I designed the DataBear as a small, simple unit, powered by just about anything and capable of monitoring ordinary analog data at very ordinary rates.

Of course, I wanted to include as many features of more expensive loggers as I could while keeping down the chip count and printed-circuit size. The 6811 makes that possible.

The DataBear packs the 6811, 8K or 32K of static RAM, 8K or 32K of EPROM, bus logic, real-time clock, serial interface, temperature sensor, and two external channels of analog input onto a board 2.6x3.0 inches. With the battery packs, the whole unit is only 1x4x6 inches. It's portable and powerful, thanks to the 6811.

Slow And Steady Continuous Mode

I set up the DataBear to collect data over a long period of time, from once per second to once every 18 hours. Each

Figure 1—Basic 6811 Features

Internal Memory: 256 bytes static RAM
512 bytes EEPROM
0 - 12K bytes mask ROM (programmed at factory)
External Memory: 64K byte address space with mux'd bus
Serial Communications Interface (SCI), to 131K baud
Serial Peripheral Interface (SPI), clock speed to 1/2 crystal freq.
Parallel I/O: up to 38 bits, some dedicated Input or Output
16-bit Timers: Four-Stage Prescaler
3 Event Capture Counters (ICF)
5 Counter Controlled Outputs (OCF)
Pulse Accumulator Input (PAI)
8 Channels of 8-bit Ratiometric A/D Conversion
2 Hardware External Interrupts: XIRQ (non-maskable) and IRQ (maskable)
Programmable Real-Time Interrupt and Watchdog Timer
WAIT/STOP Low Power Modes
6801-based instruction set plus: 16x16 Integer and Fractional Divide
Bit Manipulation Instructions
Direct, Extended, and Indexed (2 16-bit Index Registers) Addressing

♦ ♦ ♦

sample is an average of eight rapid samples, thus minimizing high frequency noise.

You'll probably use Continuous Mode for most field, shipping, or remote applications. When you're sampling at a very slow rate, the processor can shut down between samples, saving power.

Programmable Fixed Mode

A second mode uses the real-time clock to take bursts of data at preset times. This is called Fixed Mode and it lets you control the time resolution of the data.

Real Time Mode

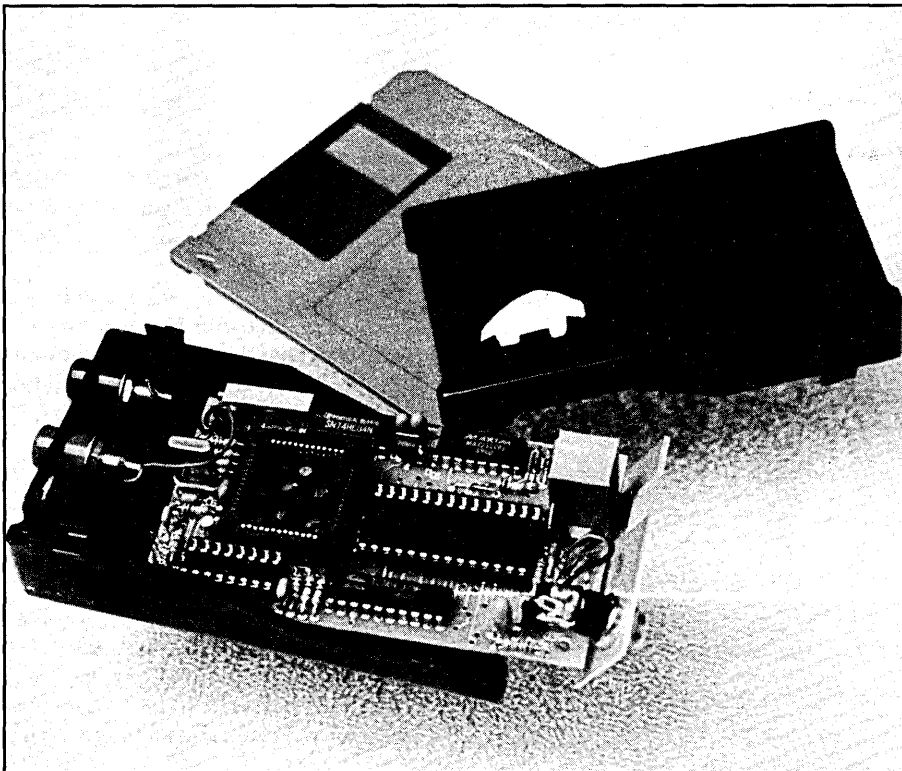
The third logging mode samples on request by a host computer. A command string from the host triggers the sampling. The Bear then reports its data in either binary or ASCII. The maximum rate is approximately 20 samples per second, limited mainly by the speed of the serial link.

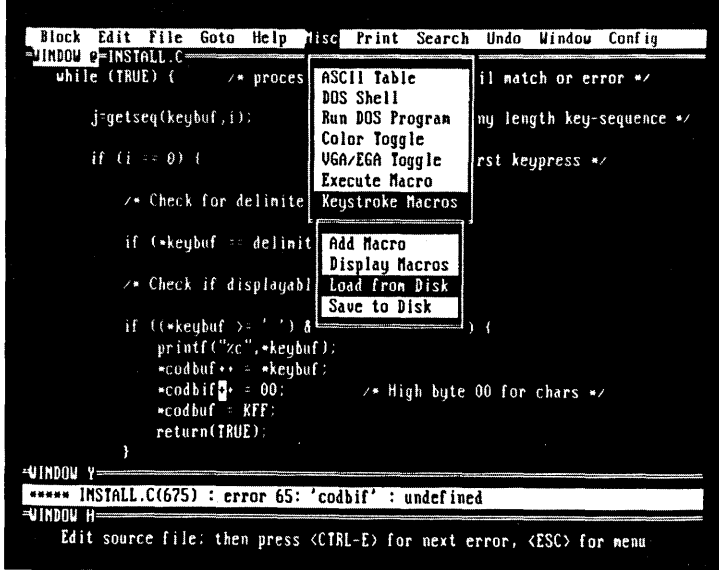
The DataBear's time-controlled operation makes heavy use of the 6811's SPI port to service clock interrupts and read the clock time and date. Because of the SPI's high speed, these transactions don't take long, providing for low power consumption and a high sampling rate.

Analog Conversion Numbers

The DataBear has three analog inputs. The 6811 has an eight-channel A/D Converter, but I used two to monitor the power and left three for expansion. Of the three left, I used one channel to monitor internal temperature. You get the other two for data. The converter resolution is eight bits, its accuracy limited by the voltage reference and by the performance of your sensor.

Eight bits of resolution is acceptable resolution for the vast majority of field data logging requirements. For example,





Introducing . . .

The 1st Family of Low Cost, Powerful Text Editors

VEDIT Jr. \$ 29
VEDIT \$ 69
VEDIT PLUS \$185

Finally, you can choose the best editor for your needs without compromising performance or paying too much. And organizations that want the "same" editor for everyone can pick VEDIT® for most users and VEDIT PLUS for their power users.

The new family of VEDIT text editors are upwards compatible, easy to use and offer exceptional performance, flexibility and stunning speed. (3 to 30 times faster than the competition on large files where speed really counts.)

Call for your free evaluation copy today. See why VEDIT has been the #1 choice of programmers, writers and engineers since 1980.

VEDIT Jr. — Unmatched performance for only \$29.

All VEDIT editors include a pull-down menu system with "hot keys," context sensitive on-line help, pop-up status and ASCII table, a configurable keyboard layout and flexible, unlimited keystroke macros. Edit files of any size and any line length. Perform block operations by character, line, file or column. Undo up to 1000 keystrokes— keystroke by keystroke, line by line, or deletion by deletion. Automatic indent, block indent and parentheses matching speed program development. Word wrap, paragraph formatting, justification, centering, adjustable margins and printing for word processing. Run DOS programs.

VEDIT—A best value at only \$69.

Simultaneously edit up to 36 files and split the screen into windows. Search/replace with regular expressions. Includes the best compiler support available—menu driven, easy selection of compiler options, supports "Include" files and MAKE utilities.

VEDIT PLUS — Ultimate programmer's tool for only \$185.

VEDIT PLUS adds the most powerful macro programming language of any editor. It eliminates repetitive editing tasks and permits creating your own editing functions. The macro language includes testing, branching, looping, user prompts, keyboard input, string and numeric variables and control over the size, position and color of windows. Source level macro debugging with breakpoints and tracing. Macros developed with VEDIT PLUS also run under VEDIT.

30 day money-back guarantee. Call for pricing of XENIX, OS/2 and FlexOS versions. Very attractive quantity pricing is available for schools, hardware and software vendors.

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. Norton Editor is a trademark of Peter Norton Computing Inc. QEdit is a trademark of SemWare.

*Supports IBM PC, XT, AT, PS/2 and clones with CGA, MGA, EGA, VGA, Wyse 700, Amdek 1280 and other displays. Also supports Concurrent DOS, DESQview, Microsoft Windows, PC-MOS/386 and most networks.

*Also available for MS-DOS (CRT terminals), T1 Professional and others.

*Free evaluation disk is fully functional and can edit small files.

FREE Evaluation Copy* Call 1-800-45-VEDIT

	VEDIT	BRIEF 2.10	Norton 1.3	QEdit 2.07
Pull-Down menus	Yes	No	No	Yes
Pop-Up ASCII table	Yes	No	No	No
Keystroke macros	100 +	1	No	100 +
Regular Expressions	Yes	Yes	No	No
"Cut and Paste" buffers	36	1	1	100
Text (book) markers	10	10	No	No
Undo keystroke by keystroke	Yes	Yes	No	No
Undo line by line	Yes	No	No	No
Normal/max Undo levels	500/1000	30/300	—	—
Variable tab positions	Yes	Yes	No	No
Configurable keyboard	Yes	Yes	No	Difficult
Integrated mouse support	Yes	No	Yes	No
FILE LIMITS				
Edit files larger memory	Yes	Yes	Difficult	No
Maximum line length	> 8096	512	65,535	512
Maximum lines/file	8,388,607	65,535	> 65,535	20,000
COMPILER SUPPORT				
Menu driven	Yes	No	—	—
Select Compiler options	Menu	Difficult	—	—
Support "Include" files	Yes	No	—	—
BENCHMARKS 50K FILE				
Simple search	0.2 sec	1 sec	1 sec	0.3 sec
Save and continue	1 sec	2 sec	2 sec	1 sec
1000 replacements	3 sec	19 sec	17 sec	2.5 sec
BENCHMARKS 3 MEG FILE				
Simple search	1:40 min	1:36 min	Cannot	Cannot
Save and continue	1:05 min	3:23 min	Cannot	Cannot
60,000 replacements	3:18 min	1:44 hour	Cannot	Cannot
Block-column copy (40 x 200)	2 sec	30 sec	Cannot	2 sec
Insert 1 Meg file in middle of 1 Meg file	1:11 min	15:13 min	Cannot	Cannot
PRICE	\$69	\$195	\$75	\$54.95

CompuView

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

Reader Service Number 7

Figure 3—MC68HC11 Family Members

Device Number	ROM	EEPROM	RAM	CONFIG ³	Comments
MC68HC11A8	8K	512	256	\$0F	Family Built Around this Device
MC68HC11A1	0	512	256	\$0D	Same Die as 'A8 but ROM Disabled
MC68HC11A0	0	0	256	\$0C	Same Die as 'A8 but ROM and EEPROM Disabled
XC68HC11B8	8K	512 ¹	256	\$0F	Early Experimental Version
XC68HC11B1	0	512 ¹	256	\$0D	'B8 with ROM Disabled
XC68HC11B0	0	0	256	\$0C	'B8 with ROM and EEPROM Disabled
MC68HC11E9	12K	512	512	\$0F	Four Input Captures and Bigger RAM and 12K ROM ⁴
MC68HC11E1	0	512	512	\$0D	'E9 with ROM Disabled ⁴
MC68HC11E0	0	0	512	\$0C	'E9 with ROM and EEPROM Disabled ⁴
MC68HC811A2	0	2K ²	256	\$FF	No ROM Part for Expanded Systems
MC68HC11D3	4K	0	192	N/A	Economy Version, No A-D and Smaller Memories ⁴

Notes:

1. The EEPROM on B Series parts requires an external 19-volt supply for programming and is not byte erasable.
2. This 2K EEPROM is relocatable to the top of any 4K memory page. Relocation is done with four bits in the CONFIG register.
3. CONFIG register values in this table reflect the value programmed prior to shipment from Motorola.
4. Available in 1988.

only guaranteed for 10 nanoseconds following the trailing E-clock edge. Either you carefully watch the number of gate delays between the processor E-clock and data receiving device or you generate an intermediate clock edge. 6809 users have the quadrature (Q) clock edge available to help grab data.

You can also operate the 6811 in the Single-Chip Mode if you install the program in the internal masked ROM (great for high-volume developers).

For the best of both worlds, Motorola makes a companion chip, the 68HC24, which restores lots of parallel I/O to an external memory design. This device connects to the 6811's address/data bus and restores the 16 bits of parallel I/O lost to the bus. It's as if the parallel ports were built into the 6811! So you can have it all.... For a price.

SCI Software

The 6811 has an on-chip SCI for serial communications. The SCI can generate interrupts or set status flags and it senses Framing and Overrun errors.

Though the SCI won't automatically generate and check parity, it will send and receive parity bits supplied by software. Plus, you can use incoming data to wake up the processor after you've left it in WAIT or STOP (low-power) mode.

Since baud rates get generated by dividing the processor's clock, you'll want to choose the processor frequency carefully. I selected 8.0 MHz so the Bear can talk at all the standard baud rates.

The 6811's SCI output is logic-level, not the ±12V standard for RS-232. I chose not to use the familiar 1488/89 RS-232 interface because of their hefty power needs and their use of ±12V.

Instead I used the Maxim MAX232. It has two line receivers and two line drivers and it has a charge pump for generating RS-232 line-output voltages. By controlling the amount of power to the charge pump, I control power consumption.

Real-Time Clock And Serial Peripheral Interface (SPI) System

During the initial development of the

DataBear, I used the National MM58167 clock chip. This venerable device worked well enough, but it didn't offer absolute year clocking. And it required four I/O lines plus control lines. On a 2.5x3.0 inch board, that's expensive!

Then I found the RCA 68HC68T1 clock family, now second sourced by Motorola. The '68T1 family offers a 6811-compatible SPI port, absolute year, weekday, time to hundredths of a second, and 32 bytes of static RAM in some family members. I selected a 32 KHz tuning fork watch crystal as the time-base, although you can use crystals up to 8 MHz with higher power consumption.

A caveat for clock circuit designers: tuning fork watch crystals have a poor temperature coefficient compared to the standard HC-6/U quartz plate crystals. A typical watch crystal can be adjusted to 32,768.0 Hz ±0.1 Hz. However, even at that, the clock can gain or lose several seconds per week.

It's funny; tell people that you have 3 ppm accuracy and they love it, but several seconds per week is awful! How

do watches keep time so well? It helps to keep the crystal at a constant 90 degrees by strapping it to your wrist! (A warm hand means a timely person.)

The DataBear communicates with the clock via the 6811 SPI port. The SPI system is a high bit-rate, synchronous, Master/Slave, serial data port using two data lines and a clock.

The interface supports full duplex, multiple masters, handles limited data collision detection, and even offers interrupts and status flags.

In most simple single-processor systems, the processor gets to be bus Master. All other SPI devices become Slaves. You might scratch your head trying to work out clock polarity (CPOL) and phase (CPHA), but careful scrutiny of the data sheets usually solves the problem. With only four possibilities, you can always psyche it out with a scope or debug monitor.

Once you've configured the SPI, all you need is software which knows how to talk with each slave. Each slave will probably speak a slightly different protocol.

Our 68HC68T1 clock speaks data in nybbles and (of course) in a reverse

order, but that's about par for the course.

The Master sends data serially out the Master Out Slave In (MOSI) pin, while simultaneously generating a shift clock on the SCK pin. Slaves send data to the Master on the Master In Slave Out (MISO) pin. Tie the 6811's SS line high when it's the Master Device; use the pin as a device select (Slave Select) when it's the Slave.

Make sure that the SPI clock rate is not too fast for any of the Slave devices, especially CMOS parts. Each Slave will probably have its own device select line, so you can program a custom clock rate, polarity and phase for each device.

I've found the SPI is probably the most trouble-free subsystem in the entire DataBear. More SPI parts seem to be available each month. Other serial devices, such as A/D or D/A Converters, often work fine even though they are not marketed as SPI compatible.

A/D Converter Subsystem

The 6811 A/D Converter is a ratiometric converter. The output of the converter is a digital ratio of the input signal to the voltage references. With the

6811, these references are V_{rl} and V_{rh} . Any voltage between V_{rl} and V_{rh} will result in an eight-bit output according to the formula:

$$\text{Output (0-255)} = \frac{\text{Input} - V_{rl}}{V_{rh} - V_{rl}} \times 255$$

It's easy to tie V_{rl} to ground and V_{rh} to V_{cc} , but digital noise can be a problem. You can reduce the noise by adding an RC-filter to the V_{cc} power at V_{rh} . Better still, use a zener diode or adjustable voltage reference to regulate V_{rh} .

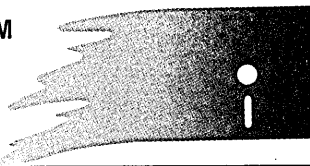
Also be careful about how you route a ground to V_{rl} . If digital currents flow in ground traces used for V_{rl} , you'll have a noise problem again.

I used a National LM235-2.5 as a reference diode for the DataBear. The LM136/236/336 series of references are adjustable, conditioned zener diodes. The nominal tolerance is 5%, and I use a pot to divide the reference's output down to 1.280 Volts. Refer to the "Voltage Reference" section of Figure 4 for a schematic of the DataBear reference circuit.

The A/D Converter in the 6811 has different modes of conversion controlled

Continued on page 16

QEdit™ ADVANCED



THE Quick EDITOR

The Fast, Easy-To-Use, Full-Featured Text Editor. Now Also Available as a TSR Utility!

QEdit has always been the text editor that combined price, performance and value. QEdit TSR makes that value even better by giving you instant access to the power of QEdit—no matter what program you're working in.

- QEdit is packed with features:
- Completely configurable, including keyboard and colors
 - Simple to install, requires less than 50K disk space
 - Simultaneous file editing—limited only by available memory space
 - Open up to eight windows
 - "Pop-Down" menu system and customizable Help screen
 - Easy-to-use macro capability—including keyboard recording

- Column blocks
- Recover deleted text
- Exit to DOS or DOS shell from within QEdit
- Execute command-line compilers from within QEdit
- Wordwrap and paragraph reformat capability
- Import files and export blocks
- Great for use with laptops—QEdit saves battery drain by editing files entirely in memory

System Requirements
QEdit requires an IBM PS/2, PC/AT, PC/XT, PC, PCjr, or compatible. Minimum system requirements are 64 KB of memory, PC-DOS 2.0 or MS-DOS 2.0 or greater, 50 KB of disk space. QEdit runs GREAT on floppy based systems and laptops. QEdit TSR also requires EMS, XMS, or a hard disk.



QEdit is a great piece of software. Highly recommended. **99**
John C. Dvorak,
PC Magazine
September 12, 1989

QEdit is, without question, the smallest, fastest, most versatile text editor in the DOS world. **99**
P. L. Olympia, Ph.D.,
DBMS
July, 1989

Winner, 1989 Data Based Advisor Reader's Choice Award

UNCONDITIONAL MONEY BACK GUARANTEE

INTRODUCING QEDIT TSR
Instant Access to the World's Best Editor
SPECIAL PACKAGE PRICE OF ONLY \$99.00 INCLUDES QEDIT ADVANCED

To order direct call
404-641-9002
extension 12



Add \$3.00 for shipping—\$10.00 for overseas shipping. **UPS 2nd DAY AIR available within the U.S. for ONLY \$5.00**
COD's accepted—please add \$3.00
Georgia residents add 4% sales tax

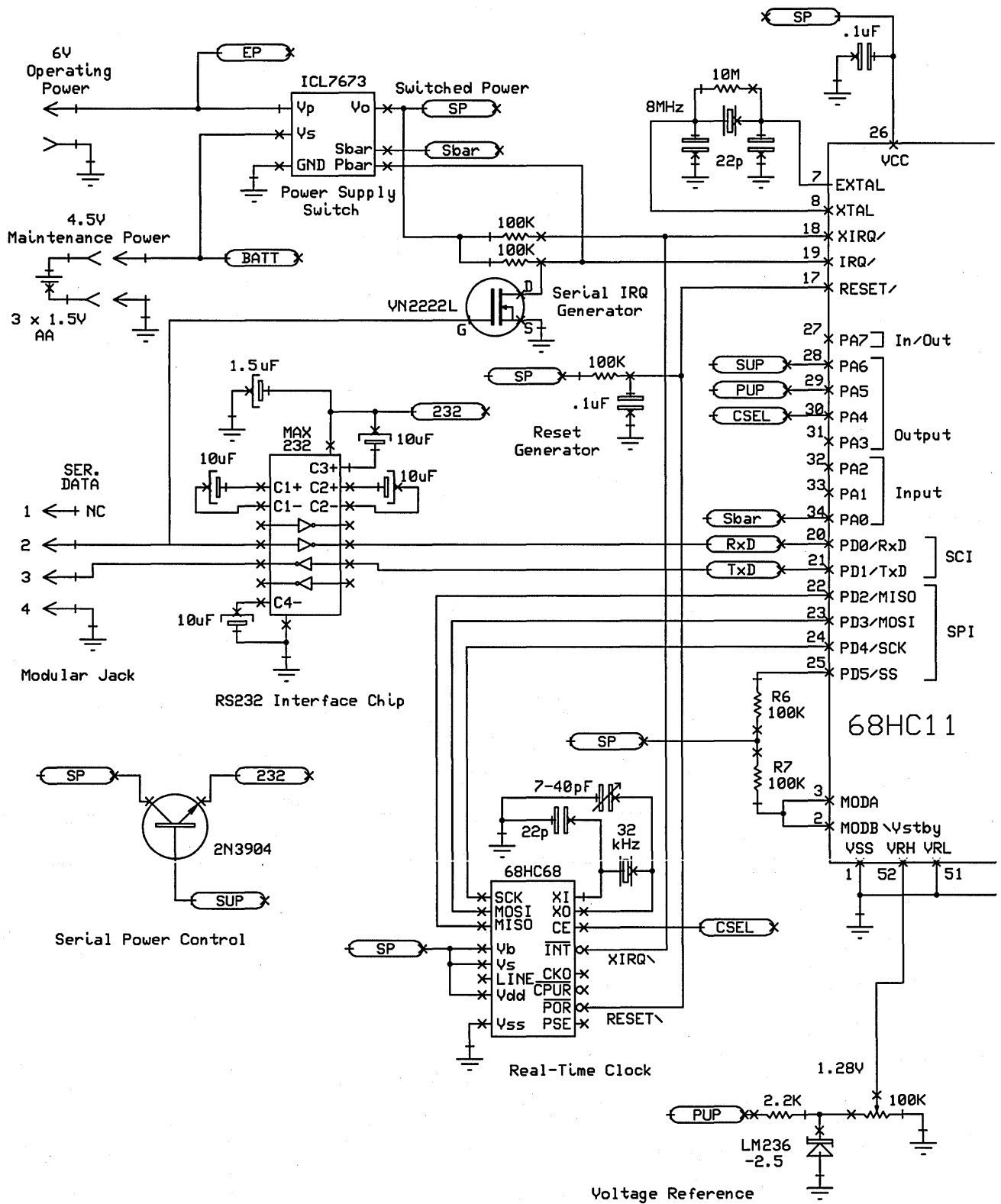
SEMWARE®
4343 Shallowford Rd. • Suite C-3
Marietta, GA 30062-5003

QEdit is a trademark and SemWare is a registered trademark of Applied Systems Technologies, Inc.
© 1989 Applied Systems Technologies, Inc.

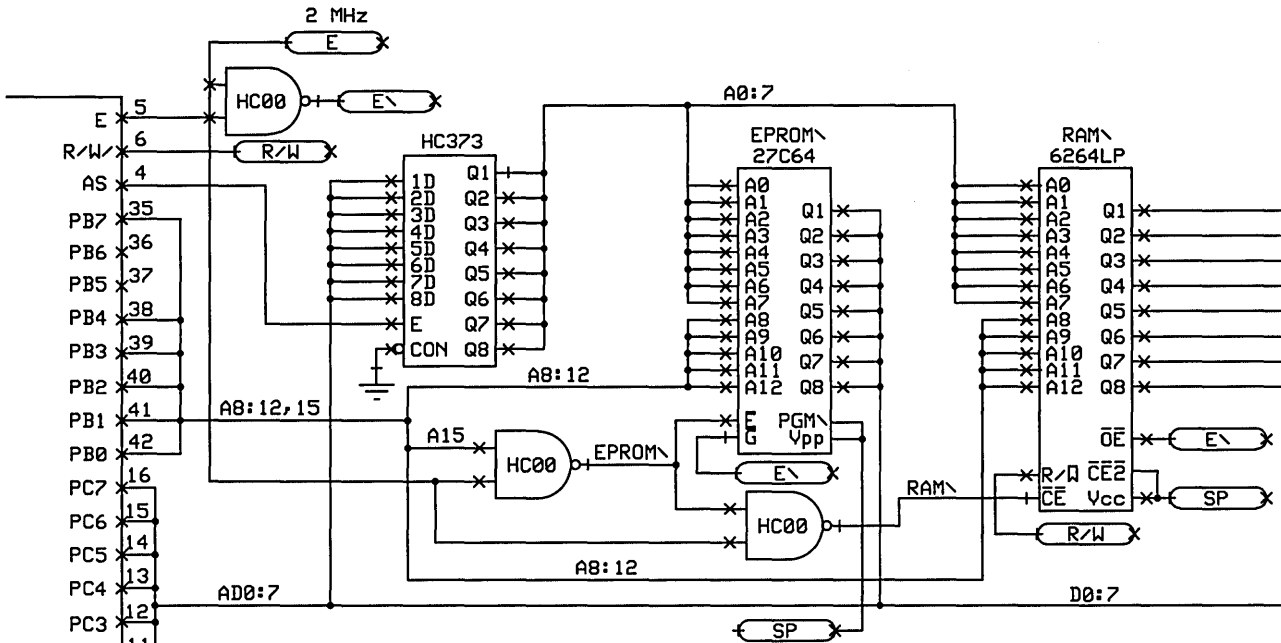
QEdit™ \$54.95 ALSO AVAILABLE **\$79.00**
ADVANCED QEdit™ for OS/2

Reader Service Number 127

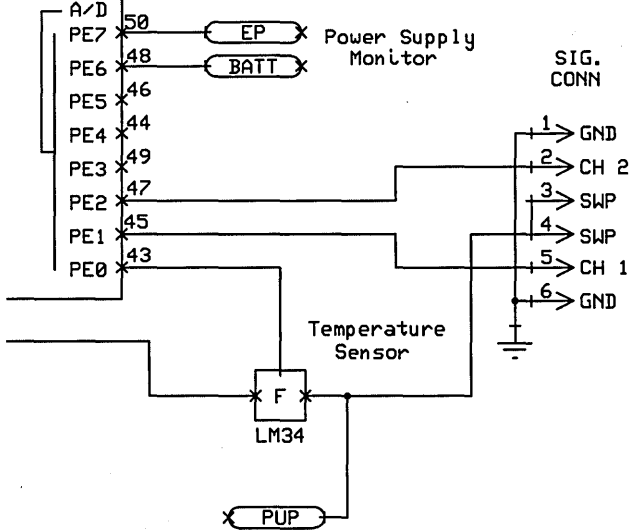
Figure 4—DataBear™ Functional Block Diagram c.1990



EPROM\ = \$8000 - \$FFFF
 RAM\ = \$0000 - \$7FFF



SUP - Serial Power Control
 PUP - Analog Power Control
 CSEL - Clock Select



Langan Products
 2660 California Street
 San Francisco, California 94115
 (415) 567-8089

by setting programmable flags referred to as:

SCAN - Continuous Scan Control, and;

MULT - Multiple/Single Channel Control.

A single channel can be sampled once or continuously; a group of four channels can be sampled in sequence, either once or continuously. CCF (conversion complete flag) indicates there's valid data.

Because the conversion is quick, taking just 32 E-clock cycles, I didn't set up an interrupt for conversion completion. (Exiting and reentering the interrupt driver would take almost as long as the conversion, so why bother?)

Any unused A/D channels can double as high-impedance digital inputs (Port E).

Real Signals

I've installed an LM34 (or LM35) temperature sensor on the printed circuit board. The LM34 is a linear, positive temperature coefficient sensor. It outputs a voltage that's proportional to temperature from -50 to +300 degrees F. With the LM34, the output is 10mV/degree F with 1.000 Volts at 100 degrees F. Easy, huh?

Supply anywhere from 4 to 30 Volts to the sensor and connect it to the 6811 A/D Converter and you've got temperature!

Although there are more and more linear sensors with the conditioning electronics built-in, most aren't as easy to use. Many outputs range between 0 and 0.1/0.5/5.0/10.0 Volts, 0 to 1 mA, 4 to 20 mA, strain gage mV/V-type outputs, frequency, pulse-per-event, etc. You'll have fun monitoring a collection of these outputs with a single system.

Mainframe loggers, such as those from Fluke and Hewlett-Packard, have module connectors for signal conditioning assemblies. These assemblies are quite expensive, and designing a logger to handle all the standard modules can make the logger expensive as well.

I wanted the DataBear to log any variable, any signal, as long as it was between 0 and 1 Volt! Obviously I needed to add some signal conditioning.

For example, there's a standard relative humidity sensor package for the Bear. The sensor is a half-bridge humidity-to-strain transducer. You apply power and it spits out a humidity-proportional voltage.

Figure 5—Interrupt Vector Assignments

Vector Address	Interrupt Source	CC Register Mark	Local Mask
FFCO, C1 • • FFD4, D5 FFD6, D7	Reserved • • Reserved SCI Serial System	— I Bit	— See Table A Below
FFD8, D9 FFDA, DB FFDC, DD FFDE, DF	SPI Serial Transfer Complete Pulse Accumulator Input Edge Pulse Accumulator Overflow Timer Overflow	I Bit I Bit I Bit I Bit	SPIE PAII PAOVI TOI
FFE0, E1 FFE2, E3 FFE4, E5 FFE6, E7	Timer Output Compare 5 Timer Output Compare 4 Timer Output Compare 3 Timer Output Compare 2	I Bit I Bit I Bit I Bit	OC5I OC4I OC3I OC2I
FFE8, E9 FFEA, EB FFEC, ED FFEE, EF	Timer Output Compare 1 Timer Input Capture 3 Timer Input Capture 2 Timer Input Capture 1	I Bit I Bit I Bit I Bit	OC1I OC3I OC2I OC1I
FFF0, F1 FFF2, F3 FFF4, F5 FFF6, F7	Real Time Interrupt IRQ (External Pin or Parallel I/O) XIRQ Pin (Pseudo Non-Maskable Interrupt) SWI	I Bit I Bit X Bit None	RTII See Table B None None
FFF8, F9 FFFA, FB FFFC, FD FFFE, FF	Illegal Opcode Trap COP Failure (Reset) COP Clock Monitor Fail (Reset) RESET	None None None None	None NOCOP CME None

Table A
SCI Serial System Interrupts

Interrupt Cause	Local Mask
Receive Data Register Full	RIE
Receiver Overrun	RIE
Idle Line Detect	ILIE
Transmit Data Register Empty	TIE
Transmit Complete	TCIE

Table B
IRQ Vector Interrupts

Interrupt Cause	Local Mask
External Pin	None
Parallel I/O Handshake	STAI

So I built an op-amp circuit which would power the sensor, amplify its output, and remove the resulting offset so that the final signal lay between 0 and 1 Volts. The Bear controls power to the interface's electronics; it's turned on before each sample is taken and removed afterwards.

Each type of sensor required its own interface electronics. There are many good books available on interfacing sensors to A/D Converters. (Analog Devices has a good series of texts available. National and Motorola both have loads of good data sheets and application notes for little or no charge. The *Op-Amp Cookbook* by Walter Jung is also full of interface ideas.)

The 6811 makes it easy for the designer by accepting different high and low voltage references, extremely high input impedance, built-in sample-and-hold circuitry, and an eight-channel multiplexer. All are accessible and controllable via CPU registers.

Parallel I/O

Depending on the flavor of 6811, you have as many as 38 parallel I/O lines. However, in the expanded, multiplexed address bus mode, with ports B and C working as the address/data bus, you cut I/O to 22 lines.

The SCI port uses 1 or 2 lines and the SPI port uses 4. Because the DataBear uses both the SCI and the SPI, we're left with 16 lines. I reserved the A/D Converter channel inputs for analog, leaving just 8 parallel I/O lines.

Port A's I/O is closely linked to the advanced timer functions built into the 6811 (see the section on Timers). Some of the Port A lines are dedicated inputs (PA0, 1, and 2), others are dedicated outputs (PA3, 4, 5, 6), and the remaining pin (PA7) is bidirectional under program control.

In the single-chip mode, ports B and C become available for parallel I/O. Port B is all output. Port C is bidirectional.

Port D (which has only six lines) is split between bidirectional I/O, SCI port (RD and TD), and the SPI Port (SCLK, MOSI, MISO, SS). All, or part of, the I/O lines can be used as parallel I/O depending only on your uses of the SCI and SPI systems.

I/O pins can sink from 1 to 10 mA, depending on how much current adjacent pins sink, but they source only about 1 mA, max. If you want to control any significant amount of current, you'll

need to add a buffer chip or transistor. I have used the 6811 to drive a sensitive relay, but it's not a good idea.

I used a parallel I/O line to control the current to the MAX232 serial chip. The chip can pull around 30 mA when the charge pump starts up. (It's always extra work to prime a pump.) The Serial Power Control section in Figure 4 shows how a ten cent 2N3904 transistor handles the MAX232's current flow. The ULN2001 series driver chips also do a good job of driving current-gobblers.

Timers

One of the nicest features of the 6811 is its array of timers and counters. The general architecture is based on the MC6840 counter-timer, with improved and enhanced control and interrupt capabilities. The 6811 can act as a frequency counter, a waveform generator, or period timer. If you're clever you can even make it do combinations of these if you pay careful attention to subtleties like interrupt latency and counter rollover.

The Pulse Accumulator Input (PAI) operates through pin 7 of Port A. Transitions on pin 7 clock an 8-bit counter. Counter rollover can generate an interrupt which you can also count, thus extending the counter to 16-bits or beyond. I've used the PAI function up to 50 KHz. It's rated to count at one-half the E-clock frequency.

The Input Capture Functions (ICF) let you time events (transitions) via a 16-bit counter running at the E-clock frequency.

Output Compare Functions (OCF) do the opposite—they cause a level transition at an external pin whenever their control register matches the value of the 16-bit free-running counter.

All the features are linked to the interrupt system, with reassignable priorities. I can hear the wheels turning out there, especially among the servo-motor guys! While the DataBear doesn't make much use of the timer functions, they are a 6811 crown jewel.

Low Power Modes

Most CMOS processors will shift to a low-power mode or modes under program control. The 6811 has two low-power modes: WAIT and STOP.

In WAIT, the E clock oscillator continues for the duration of the WAIT. Plus you can choose which subsystems stay active.

During STOP the E clock oscillator stops, shutting down all internal systems, such as the SCI and SPI.

Current consumption in WAIT can be as low as 6 mA, depending on the processor modes and number of peripherals still running. In STOP mode the current consumption drops to well under 1 mA.

It takes an interrupt or a RESET to wake the processor from either low-power mode.

In WAIT mode, the interrupt can be an incoming serial data character or a timer timeout. The SCI subsystem also has a special wake-up feature which allows an active serial RD line to wake up the processor.

When the processor's in STOP mode, the interrupt must come in from outside. The DataBear's processor spends approximately 99.5% of its time in STOP mode, so I've included three sources of external interrupts: the clock chip (XIRQ), the power source switch (IRQ), and a FET connected to the RD serial line (also IRQ).

Clock interrupts normally occur once a second and take priority over other interrupts. I use the clock interrupt as the Bear's time base, driving the data sampling. In Continuous Mode, the clock interrupt is always every second so sample rates are multiples of one second. In Fixed Mode, the clock interrupt occurs at intervals as short as $1/128^{\text{th}}$ of a second during the programmed sampling bursts.

Serial interrupts will wake the Bear from dead STOP by pulling the IRQ line low. +12 Volts on the RD line turns on the VN2222 FET, which is wire-ORed with the open-drain status output of the ICL7663 power source switch.

The ICL7663 power source switch also pulls the IRQ line low whenever the supply goes higher than the maintenance level. The DataBear is able to determine which interrupt awakened it by the context in which the interrupt occurred and by some status checks in the interrupt handler.

Power

For battery-powered systems, like this, a CMOS microprocessor offers two major advantages: low power consumption, and the ability to operate from a range of voltages. The 6811 consumes approximately 20 mA operating on 3 to 7 Volts.

Using batteries sounds like a piece of cake until you consider the more subtle

issues. First, batteries develop internal resistance as they discharge so the voltage drops. Although open-circuit voltage may be fine, it may drop below minimums as the processor turns on.

I gave the Bear a 4.5 Volt maintenance supply to allow an orderly cessation of sampling as the output of the regular battery drops. An Intersil ICL7663 selects whichever battery has the higher output.

The second and third complicating factors are switchover hysteresis and the level at which regular operation will cease completely. The DataBear preserves RAM data as the batteries run down by ceasing regular operation. The clock interrupts are turned off and the time of power loss is stored. With any luck, the user will retrieve the Bear before the maintenance supply dies.

You'll find there are many types of batteries: zinc-air, lithium, mercury, silver-oxide, ni-cad, and alkaline. In fact, there are so many options that batteries would make an article in themselves.

I finally chose alkaline AA cells because they're cheap, easy to find, and pack a fair number of Amp-hours.

Battery supplies have been the most difficult part of the system. I've planned to use the Bear for long-term, remote or unattended field measurements, which

makes the power supply issue very significant. My best suggestion is that you obtain as much technical information on the batteries you plan to use in your system, then make your selection.

Evaluation Module

Developing a PROM-able program for any microprocessor can be difficult, no matter which processor and which development tools. Unless you use commercial hardware, even the most basic system resources must be tested and debugged before you can start writing software.

Fortunately, Motorola provides low-cost development hardware for their microprocessors. There are two systems available for the 6811 family: the 6811EVM, and the 6811EVb.

The -EVb costs less than \$100 and comes with the BUFFALO monitor in ROM. It also comes with the 68HC24 so you can use all 38 parallel I/O lines, as well as the external PROM and RAM. The memory space available is 64K, with the monitor PROM in the upper 8K at \$E000. The -EVb's manual includes code examples.

The -EVM will run you between \$400 and \$500 but has more complete debugging support. They've thrown in a monitor with this board, too.

I used a 6811EVb to do the initial development for the DataBear. That way I could concentrate on getting the software running without worrying about the hardware.

I had to learn how to use the -EVM, but that's a lot easier than learning developing from scratch. I built all the custom hardware on a small wirewrap board and connected it to the expansion connector on the -EVb.

By downloading it to the -EVb (via the monitor) and carefully setting breakpoints, I was able to debug the code without emulators. I used the Archimedes cross-assembler running on my DOS system.

The initial prototype consisted of the custom wirewrap board and the -EVb sandwiched together and squashed into a plastic box. Though the combination was about four times the size of the final Bear, clients and customers were enthusiastic!

My message is, *it can be done* on a nonexistent budget with simple tools.

Vectors

Once you start developing code for a

standalone 6811, you have to deal with certain facts. First, define the event vectors. The event vectors are addresses of code routines which the processor executes in response to certain events, such as interrupts or resets.

The 6811 supports all the usual vectors, such as the external hardware interrupts and reset. It adds vectors for all the timer functions, SCI and SPI ports, real-time and watchdog timers, illegal instruction, and software interrupts (SWI). For the 6811, the vector table lives at \$FFD0 through \$FFFF. The 6811 reset vector lives at \$FFFE:FF.

You'd better have something in mind for every single vector, even if it's only an RTI (Return from Interrupt). Prototypes often suffer from spurious interrupts which can be a bear (no pun intended) to isolate.

Relocatable Registers And RAM

The 6811 must be initialized within 64 E-clock cycles of a reset. The most important is mapping the processor internal registers and internal RAM to the desired 4K (\$1000) page boundary. Write the locations of the RAM and registers to the INIT register in the first set of instructions following reset.

You can use the init code to determine: whether the IRQ interrupt is level or edge sensitive, the oscillator delays after STOP, and watchdog timer periods. Take a special look at the INIT, CONFIG, and OPTION register descriptions in the Motorola literature.

EEPROM

Some versions of the 6811 include 512 bytes (or more) of EEPROM. This can be particularly useful for storing serial numbers, network or board addresses and IDs, configuration or password data, etc. The EEPROM is located at a fixed memory address. If you have EEPROM, make sure you can instruct your linker to avoid its location.

Miscellaneous Notes

I won't try to cover everything, but I should mention a few tricks I discovered during this project.

First, I used a current probe and an oscilloscope to find out which subsystems were drawing significant power when they weren't being used. Helped cut power use.

Second, I found it's important to let external devices stabilize (after the processor comes out of STOP) before sam-

68000

SK*DOS - A 68000/68020 DOS containing everything you expect in a DOS - on-line help, multiple directories, floppy and hard disk support, RAM disk and/or disk cache, I/O redirection, and more. Supplied with editor, assembler, Basic, powerful utilities. Supported by Users' Group and BBS. Software available from other vendors includes C compiler, Basic, editors, disassemblers, cross-assemblers, text formatter, communications programs, etc. Priced at \$165 with configuration kit, less if already configured for your system.

HARDWARE - 68xxx systems start at \$200. Call or write.

Star-K Software
Systems Corp.

P.O.Box 209

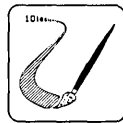
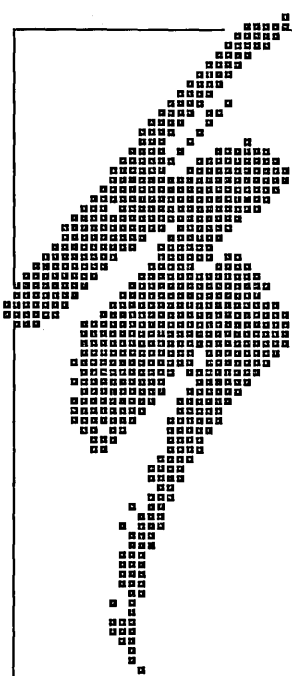
Mt. Kisco NY 10549

(914) 241-0287 / Fax (914) 241-8607

Reader Service Number 40

GRAPHICS!

Add lightning fast graphics to your programs quickly and easily through the popular PCX file format. Why reinvent the wheel? Make your programs immediately compatible with hundreds of packages from Aldus PageMaker to ZSoft's PC Paintbrush with these linkable graphic libraries.



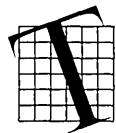
"An exceptional product" - Programmer's Journal, Aug '89 \$195

Version 3.5 of the **PCX Programmer's Toolkit** gives you over 60 powerful functions to manipulate bitmapped graphics. Use Virtual screens, Super VGA modes, LIM 4.0 support, a 300 page manual, 9 utilities including screen capture and display, and the fastest routines on the market.



Need Special Effects, but caught in a GRASP? \$99

Why create a demo when you can create the real thing? Don't be trapped in a slideshow editor or demo program when you can use **PCX Effects** for the PCX Toolkit and your favorite programming language. A Music Language and spectacular effects for exploding your graphics!



Blazing Graphics Text \$149

With **PCX Text** you can display text with graphics as fast as it always should have been. Display characters, strings, fixed and proportional text, background transparency, and more. Includes a font editor, 85 fonts, and text utilities for blazing graphics bitmapped text.

NO ROYALTIES!
30-DAY MONEY-BACK GUARANTEE!
SOURCE CODE AVAILABLE!

All packages support 12 compilers for C, Pascal, Basic, Fortran, Assembly, and Clipper. All modes of the Hercules, CGA, EGA, VGA, and Super VGA adapters are supported, up through 800x600x256 (22 modes in all). Assembly Language source code is optionally available. Trademarks are property of their respective holders.

G.E.N.U.S.
MICROPROGRAMMING
11315 Meadow Lake • Houston, Texas 77077 • (713) 870-0737

FAX: 713-870-0288

Add \$5/package for UPS Ground
Texas Residents add 8% Sales Tax

For orders and information, call:

1-800-227-0918

VISA/MC/AMEX/COD/CHECK/PO

pling data or sending them instructions. This is important because low-power analog systems can take several hundred microseconds to wake up.

Third, I found that the input impedance of the 6811's A/D Converter is so high that open, unused inputs will show significant voltage. Tie unused inputs to ground through a 100K resistor.

Fourth, with any low-power system you'll need to pay attention to input impedance. Pullups should have a pretty high resistance; 100K works well.

To Boldly Go...

So what's the Bear doing?

It's logging blast furnace temperature data for specialty melts, monitoring freezer temperatures, and recording weather data.

It has been stuffed inside a fresh fish during air shipment ("Bear Eaten by Fish!").

It's been taken to the opera (you get drowsy in the balcony when it's 80+ degrees up there), and hung on grape vines in Napa Valley. It's even measured actual earthquake weather in San Francisco!

The Bear Of The Future

Where does it go from here? For starters, like any creative designer, I want to take advantage of more features of the 6811. I'd like to use the frequency counting and pulse counting features of the PAI system so the Bear could log digital data. It would be nice to have the Bear log the time of events, not just at regular intervals. And, of course, there are thousands of sensors that can tie right into our 0 to 1 Volt standard input.

Eventually I expect to enter the world of surface-mount, volume production. I will have to do a new circuit board and find a whole new list of suppliers and packages to make the Bears.

If I Knew Then What I Know Now

No design article would be truly complete without a True Confessions section. Hindsight being what it is, I certainly would not have done the whole package in assembly. As I write this, other programmers are adding software features. It would be much easier now if I'd developed in C from the beginning.

I made it much harder on myself in the battery department by not seeking

expert advice at the beginning. We've learned some things about batteries that cost time and money. (My money.)

I am thankful that I used a processor which I knew well. There are many subtleties in the use of microcontrollers. They are powerful critters, but you can wind up with powerful headaches if you ignore their little gotchas. Take the time to learn their ways and you can do a lot.

References

Motorola 68HC11A8 HCMOS Single-Chip Microcontroller Manual, Motorola Document Number MC68HC11A8/D.

Motorola 68HC11A8 HCMOS Single-Chip Microcontroller Programmer's Reference Manual, Motorola Document Number MC68HC11PM/AD.

◆ ◆ ◆



Waylaid By Snakes:

A Lesson In Object-orientation

When you're up to your knees in a new issue (this one), the last thing you want to deal with is a bunch of snakes. Fortunately, these snakes won't object to GRAS (that's Generally Regarded As Safe). Now, whose system would benefit from a case of slithers?

Understand now, I set out to do what I was supposed to—build a voice-control system using the Motorola VCP-200 (Radio Shack #276-1308). This chip will recognize five phrases independent of the speaker: go, stop, turn left, turn right, and reverse. Its data sheet (from Radio Shack) includes a schematic for the chip's circuit. All I needed to do was connect the chip to the microphone and analog amplifier/filter from *Micro C* #50.

But I didn't do it *yet*. I became distracted trying to imagine how I'd use my apparatus. (I should have listened to Larry, who never gets bogged down in these kinds of details.)

So I digressed—what was I to do with the signals from the chip? How could I use commands like: "left turn" and "reverse?"

I speculated—draw something on a screen: an entity which moved under voice control (leading perhaps to a vocal PAC MAN). But what entity?

Then (as though in a vision on a hot, hazy day in the high desert) it came to me: snakes. A snake (an object) is a bunch of linked segments (also objects). To move the snake forward, I'd create a new segment, call it the head, then tell it (the lead segment) to send a message back to the tail saying "add a new segment" or "drop a segment."

So I built snakes instead of connecting hardware. In the process, I generated

some interesting C++ code and utilized several C++ 2.0 features. SNAKES, my code creation, is (of course) a riot.

When you run SNAKES, tell it how many snakes you want, how long you want each snake to be, and how quickly you want to see action. By increasing the delay factor, you can slow things down as much as you like. Each has its own ASCII character.

I compiled SNAKES using Zortech C++ version 2.06 and their disp package to increase display speed. In the *Micro C* listing, I've #ifdefed disp out so SNAKES will work (albeit slower) with other implementations of C++. (These include UNIX as long as you have an ANSI terminal, or are willing to rewrite the brief `segment::draw()` and `segment::erase()` routines.) You can also easily change these routines to support graphics.

Classes

Figure 1 (SNAKE.HPP), the header file, contains the class definitions for point (which holds x-y information), segment, and snake. I made point a struct (a class with all members public) because I wanted the convenience of having coordinate information packaged together without having to bother with private data.

The constructor simply initializes the data elements x and y using the initializer list. (That is, you can treat built-in types as if they had constructors in the initializer list *only*.)

We overload the == operator to determine whether one point is equivalent to another. This prevents a snake from backing over itself.

`random_point()` is a static member function which returns a random point. `random_point()` creates a temporary object by calling the point constructor.

Note: we don't need an identifier since the object (a point) immediately returns.

A static member function addresses the class as a whole, not a specific instance of the class. This could have been an independent (non-object contained) function, but I made it static so anyone using class point can now see (and access) it. Also the identifier `random_point` is local to class point, and doesn't "pollute the global name space." For this and other reasons, static member functions are an important addition to C++ 2.0.

Segments

The segment class contains a pointer to the previous segment in the list. So any segment can send a message to the segment behind it ("drop the tail segment, add a segment").

The point object lets the segment locate itself on the screen. It draws itself and prevents its own redundancy—the entire snake slithering over itself (although other snakes may slither over it).

segment displays itself on the screen using a pattern and orients itself through a variable called heading of the enumerated type direction. heading determines a snake's direction and tries to keep it going that way.

(Aside: my original design didn't include heading. I added it after I got the system working and saw snakes doubling back on themselves, creating an ugly mess. Fortunately, it was a simple matter to add heading to the existing system. In general, try to get the system up quickly. You can't know some things while you're working on the design, only after you see the design working.)

Local Enumerations & Anti-pollution

Notice that enum direction is inside the class definition. In C++ 2.0, enumera-

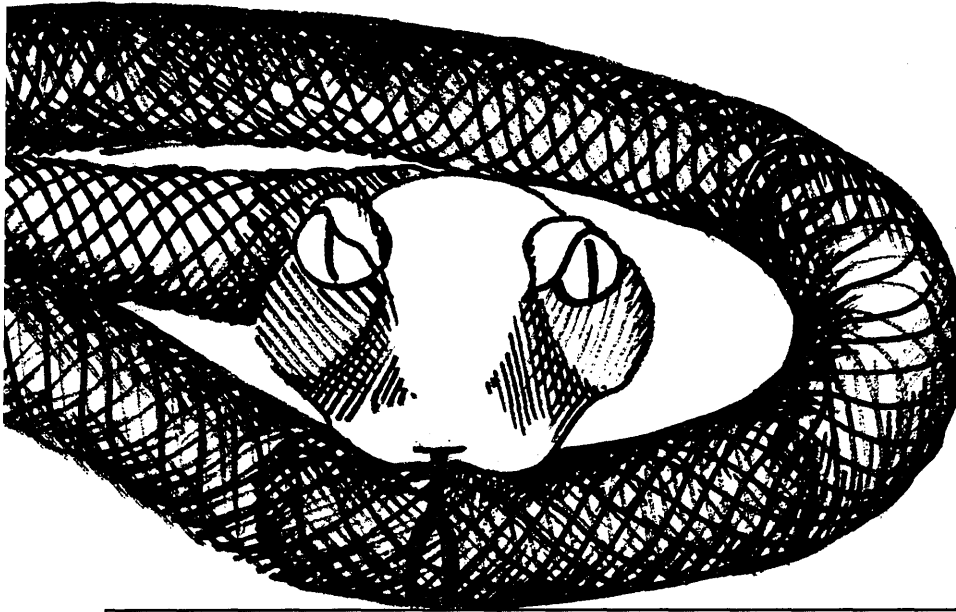


Figure 1— SNAKE.HPP

```
#include <stddef.h> // size_t definition

struct point { // coordinate representation
    int x, y;
    point(int xi = 0, int yi = 0) : x(xi), y(yi) {}
    int operator==(point rval) { // test equivalence
        return x == rval.x && y == rval.y;
    }
    static point random_point(); // generate random point
};

class segment {
    segment * previous;
    point sp; // position on screen
    char pattern; // to display on the screen
    enum direction heading; // "tendency of motion" of the snake
    void erase();
    void checkheap(); // make sure this is on the heap!
public:
    // this enumeration is local to the class:
    enum direction { UP, DOWN, RIGHT, LEFT };
    void draw();
    // re-define operator new for this class ONLY:
    void * operator new(size_t sz);
    segment(point p, char ptrn);
    segment(segment * prv, direction dir);
    ~segment() { erase(); }
    int shed_tail(); // go back to the tail and drop it off
    point seg_point() { return sp; }
    int cross_over(point); // is this point an existing segment?
    direction path() { return heading; }
    void redraw_tail(); // redraw everything from here back
};

class snake {
    segment * head;
    int length, maxlength;
public:
    snake(point p, int size, char ptrn);
    void crawl(direction);
    void slither();
};
```

♦ ♦ ♦

When you run SNAKES, tell it how many snakes you want, how long you want each snake to be, and how quickly you want to see action.

tions are local to classes. The enumeration tag is global (everyone can refer to direction). But the enumeration names are local to the class, so they don't pollute the global namespace—you must refer to them as `segment::UP`, etc.

Preventing namespace pollution is a very important feature of object-oriented languages. Big projects often become unmanageable simply because they create more names than they can keep up with.

You can also use enumerations local to a class to create "local const" values. You can define a const inside a class, but you can't initialize it when you define it (unlike its definition everywhere else, when you *must* initialize it). Do it instead in the constructor initializer list (thus you can end up with different values for the const depending on which constructor you call).

However, you can create the equivalent of a local const *and* initialize it at the same time using an untagged enum, like this:

```
class x {
```

```
enum {red=7, blue=4}; //no tag!
// ...
};
```

Now you can use the identifiers red and blue just like const values. Well, almost—at the time I wrote this, different compilers exhibited different behaviors when I tried to use red as a dimension in an array definition. I'm not sure how it's supposed to work. Note that you can never use even a const as an array dimension in ANSI C.

Class Overloading Of "new"

Another feature of C++ 2.0 lets you overload the operators new and delete (which control dynamic memory allocation) on a class-by-class basis. Normally you do this when you want to create dynamic objects more efficiently. I've done it here because segment objects must never be created on the stack, only on the heap.

To signal the programmer, segment::operator new() (shown in Figure 2) puts a "signature" in the dynamically-allocated memory. The function segment::checkheap() looks for this signature. If it doesn't find the signature, it sends a message to the programmer. It isn't foolproof, but will usually detect misuse. All segments must be created on the stack because they're only destroyed by segment::shed_tail() when it calls delete.

"segment" Constructors

segment has two constructors. The first—for the initial segment of the snake—establishes the starting point and the pattern to be used. The second makes a segment which is linked to an existing segment, in a specified direction. Note that both constructors call checkheap() to insure the object was created on the heap.

The first constructor (Figure 2) is straightforward, but the second must "wrap" the coordinate of the new segment around the display screen if it goes off the edge. Also, the draw() method isn't called in the second constructor, since a new snake may need a new segment which crosses over an old segment of itself. The calling function must check the new segment before actually drawing it.

Recursive Member Functions

The next three functions are pure fun. They recurse back down the snake

Figure 2—SNAKE.CPP

```
#include "snake.hpp"
#include <stdlib.h>
#include <stdio.h> // assuming ANSI terminal or ANSI.SYS on a PC
#include <time.h>
#include <string.h> // memset()
#ifdef __ZTC__ // use disp package w/ Zortech
#include <disp.h>
#include <conio.h>
#endif
const height = 23; // screen height
const width = 78; // screen width

point point::random_point() {
    return point(rand() / (RAND_MAX/width),
                rand() / (RAND_MAX/height));
}

void * segment::operator new(size_t sz) {
    void * tmp = ::new unsigned char[sz]; // allocate with global new()
    memset(tmp, 'x', sizeof(segment)); // fill with x's
    return tmp;
}

void segment::checkheap() {
    if(pattern != 'x') { // check for memset pattern
        fputs("can only create segments on the heap!", stderr);
        exit(1);
    }
}

segment::segment(point p, char ptrn)
    // capriciously choose old heading:
    : sp(p), previous(NULL), heading(LEFT) {
    checkheap();
    pattern = ptrn;
    draw();
}

segment::segment(segment * prv, direction dir)
    : previous(prv), sp(prv->sp), heading(dir) {
    checkheap();
    pattern = previous->pattern;
    // if(pattern++ == 'z' + 1) pattern = 'a'; // pattern test...
    // create new segment in appropriate direction & wrap:
    switch (dir) {
        case UP :    if(--sp.y < 0) sp.y = height;    break;
        case DOWN : if(++sp.y > height) sp.y = 0;    break;
        case LEFT : if(--sp.x < 0) sp.x = width;     break;
        case RIGHT : if(++sp.x > width) sp.x = 0;    break;
    }
}

// Recursive function to go back to the beginning of the
// sequence of segments, to find the tail and remove it:
int segment::shed_tail() {
    if (previous == NULL) {
        delete this; // NULL means we're at the tail
        return 1; // indicates to the next call 'up' that the tail was
        found
    }
    // recursive call until tail is found:
    if(previous->shed_tail()) // 1 means the call removed the tail
        previous = NULL;
    return 0; // means the call didn't remove the tail
}

// Recursive check to see if p crosses an existing segment:
int segment::cross_over(point p) {
    if(previous == NULL) return p == sp;
    if(p == sp) return 1;
    // recursive call to go to the end, or a cross point:
    return(previous->cross_over(p));
}

// Recursively redraw entire snake:
void segment::redraw_tail() {
    draw();
    if(previous == NULL) return;

```

Continued on page 23

Continued from page 22

```
    previous->redraw_tail();
}

void segment::draw() {
#ifdef __ZTC__
    disp_move(sp.y, sp.x);
    disp_putc(pattern);
#else
    printf("\x1b[%d;%dH", sp.y, sp.x); // move cursor
    putchar(pattern);
#endif
}

void segment::erase() {
#ifdef __ZTC__
    disp_move(sp.y, sp.x);
    disp_putc(' ');
#else
    printf("\x1b[%d;%dH", sp.y, sp.x); // move cursor
    putchar(' ');
#endif
}

snake::snake(point p, int size, char ptrn)
    : length(0), maxlength(size) {
    head = new segment(p, ptrn);
}

// Just mindlessly goes where you tell it:
void snake::crawl(direction dir) {
    head = new segment(head, dir);
    if(++length >= maxlength) {
        --length;
        head->shed_tail();
    }
}

// find it's own way, tending to the current path:
void snake::slither() {
    const int chance = 10; // 1 in 10 chance to change directions
    direction old = head->path();
    // Choose a new direction by first throwing the dice to decide
    // whether to actually change direction. If so, choose a new
    // direction randomly:
    if (rand()/(RAND_MAX/chance) != chance/2)
        direction dir = old; // change only if magic number is rolled
    else {
        dir = (direction) (rand()/(RAND_MAX/4));
        head->redraw_tail(); // occasionally redraw everything
    }
    segment * new_seg = new segment(head, dir);
    // check that the new segment doesn't cross over the existing snake:
    if (head->cross_over(new_seg->seg_point())) {
        delete new_seg;
        return; // new direction causes cross-over
    }
    // successfully found new direction
    head = new_seg;
    head->draw();
    if(++length >= maxlength) {
        --length;
        head->shed_tail();
    }
}

static void pause(int rate) {
    for (int i = 0; i < rate; i++)
        ;
}

main(int argc, char * argv[]) {
    if( argc < 4 ) {
        fputs("usage: snake num_of_snakes size_factor pause_rate\n"
            "      CTRL-BREAK to quit\n", stderr);
        exit(1);
    }
    printf("\x1b[=71"); // turn off ANSI line wrap
#ifdef __ZTC__
```

Continued on page 24

towards the tail. It's as if you're traversing a linked list, with each link an object with its own member functions. This is inspired by the much more sophisticated "agent" example in Ravi Sethi's *Programming Languages—Concepts & Constructs* (Addison-Wesley, 1989). If the rest of the book is anything like this example, it's a gem.

The function `segment::shed_tail()` must do three things. If the current segment is the tail (in which case the previous pointer will be NULL), then it deletes the tail (thus the reason for forcing segments to be made on the heap). However, if the tail was just deleted, then the segment just ahead of the tail is the new tail.

Thus, when the tail is deleted, the recursive function returns a 1 to tell the next segment up that it should set its previous pointer to NULL to indicate it's the new tail. Finally, if nothing happens, it returns a 0 to tell the next segment to do nothing but return 0 in turn.

Don't Tread On Me!

`segment::cross_over()` takes its argu-

Unique Programmer's Books & Tools for MS-DOS & OS/2

Professional function libraries and advanced books by Michael J. Young. Create high-performance MS-DOS programs, TSRs, OS/2 programs, and Presentation Manager applications. Items not available anywhere else!

C and assembly language!

*Please call or write for
free catalog:*

Young Software Engineering
20 Sunnyside Avenue, Suite A
Department MC01
Mill Valley, CA 94941
415/383-5354

Reader Service Number 207

"dynamic programming style." In other words, we want to determine the conditions (i.e., the number and type of objects) of a program at *runtime* rather than at *compile* time. By delaying system decisions, we allow new discoveries and adjustments to be made after we've "finished" the system.

Not only does this allow us to modify a program without reprogramming, but it also allows the user to discover things about the system the programmer never even dreamed of (which may work themselves back into a future version of the program).

Whether designers like it or not, all programs go through cycles of design, use, redesign, reuse, redesign, etc. This (we maintain) is one of the reasons existing design techniques don't work so well—they provide good structure and documentation (at least theoretically), but not good design flexibility.

In main() (Figure 2), you can see that the user specifies the number of snakes, the size of the snakes, and the speed of the simulation by entering parameters on the command line.

After the ANSI C random number generator is seeded using the current time, the arguments are picked off the command line and an array of pointers to snakes is created on the heap. (Astute readers will notice I never free this space—sloppy, but fairly safe since the program releases the space when it exits.)

The #ifdef statements determine whether the sizes of the snakes are chosen randomly or are a fixed size (both options use size_factor).

I create a delay function (called pause() defined just before main()) using the ANSI C time functions declared in time.h (see Figure 3).

Get Distracted

I had a lot of fun playing with this program—got mesmerized actually. If nothing else, it's a terrific way to waste CPU cycles.

The source code and SNAKE.EXE are available through the usual Micro C channels. The source is also part of the *Using C++* source-code disk (see the Revolution2 ad this issue).

Farewell

I wrote my first article for *Micro C* four years ago. I expected to be swamped with consulting offers. I wasn't, but I wrote for every issue since.

Eventually, I combined the articles into a self-published book, *Computer Interfacing with Pascal & C*. (Academic Press will publish a very revised and expanded version called *PC Interfacing with C & C++* this fall). Next came *Using C++*, and soon you'll see *The Tao of Objects* co-written with our pal Gary Entsminger.

Writing for *Micro C* opened the doors to other magazines (I'm the C++ editor at *The C Gazette*), and speaking at SOGs gave me the practice and nerve to speak at "real" conferences. Most important, I met Larry, Gary, and the gang at *Micro C*. No question: writing that first article was one of the best moves I ever made.

I started in the "brown wrapper" days ("Hmmm... is this *Micro C* or *Naked Volleyball Quarterly*?"). The magazine has vastly improved in look and style, and it seems like we've just hit our stride. I'm sad to see it go, but perhaps we've finished creating it. Maybe it would have become an Institution rather than a platform for experimentation (*Micro C* never *did* figure out what it was about...).

There are all kinds of reasons that *Micro C* shouldn't have happened: it wasn't commercial enough, not enough market research, blah, blah, blah. It did happen though—a rare and magical fractal-chaotic event. It happened because *we* wanted it to happen, and more important, because *you* wanted it to happen.

Farewell, and look for me in the computer mags....

Bruce Eckel is the author of Using C++ (Osborne/McGraw-Hill, 1989) and a member of the ANSI C++ committee. He's the owner of Revolution2, a firm specializing in C++ training and consulting.

Reference

Sethi, Ravi; *Programming Languages—Concepts & Constructs* (Addison-Wesley, 1989).

◆ ◆ ◆

C++ Source Code

from Bruce Eckel's *Using C++* (Osborne/McGraw-Hill 1989. ISBN 0-07-881522-3). Almost 500K: complete source from the book plus additional projects. Tested with Zortech C++ 2.06 & Glockenspiel C++ (Commonview). All chapters and projects in separate subdirectories, each with a "makefile."

Projects include:

- » Dynamically-sized arrays (DynArrays)
- » Simple Database
- » TAWK: A Database Interpreter
- » A Time-Based Control System
- » Simulation Example
- » Object-Oriented Menu System
- » Text-screen Windows
- » Mathematical matrix class
- » CAD demo: mouse creates and moves graphics objects
- » MS-DOS directory-management class
- » Graphic "shape" objects
- » Code used to discover compiler bugs

And Much More!

To Order, send \$25 check to Revolution2 (address below). Overseas orders please add \$7 for air mail, and send a U.S. check in U.S. funds or an international postal money order.

Revolution2 provides on- and off-site C++ consulting & training, and embedded systems development services including analysis, design, implementation and desktop-published technical documentation.

Revolution2

Bruce Eckel Box 760
Kennett Square, PA 19348
(215) 444-0828

bix:Beckel cis:72070,32256
net: 72070.32256@compuserve.com

Building A Two-chip Terminal

Motorola's 68HC705 Makes It Simple

Karl builds the world's simplest terminal around one of Motorola's newest controller chips.

This battery-powered RS-232 terminal has it all; it's cheap, easy to build, uses readily-available parts, and boasts the newest microcontroller technology. With this project, you can get started in 68HC705 design while you build a versatile piece of computer gear.

The terminal's features make it ideal for remote and low-power use. The LCD readout and CMOS microcontroller (MCU) keep the current drain to below 40 mA. Use of a special-purpose level-shifter IC allows RS-232 operation from a single 5 Volt supply.

The 68HC705's on-chip serial port and timer system help the terminal run at 9600 baud while controlling the display and monitoring the keyboard. The design is flexible enough to allow parts substitution, if you can't find exactly what I used in my project.

The Display

My screen is a two-line liquid crystal display (LCD). You can find many formats of LCDs in most of the large mail-order ads. Although this project uses a 2-line by 16-character display, feel free to use anything else you like. Just be sure your display is compatible with the one shown.

How do you determine compatibility? Look for an LCD that uses the Hitachi HC44780 LCD controller chip. This is a surface-mount technology (SMT) chip, soldered onto the back of the display. It usually appears with one or more Hitachi HC44100 driver chips (also SMT).

The HC44780 chip does more than handle character generation. It contains an 80-byte buffer. LCDs built with this

chip can therefore display only a maximum of 80 characters.

But for many applications, a 2-line by 40-character (or 4-line by 20-character) display will do. By shopping around, you can find several different formats of these LCDs. Examples include the Hitachi H2750 (1 row of 16 characters), Hitachi LM032L or Optrex DMC20215 (2 by 20), or Hitachi LM044L (4 by 20).

Mail-order suppliers known to have carried LCDs recently include Alltronics, TimeLine, and Digi-Key. Also check your favorite surplus house; these displays are becoming common surplus items.

Most LCDs include some form of backlighting. The most common style uses a strip of electroluminescent material (ELM), mounted behind the panel. When you apply the proper AC voltage to the two pins on the ELM, it emits a soft blue or green light. This makes your terminal easy to use at night or in low light.

I added a special, three-terminal voltage converter designed for use with ELM. This module changes 5 Volts DC to 200 Volts AC at about 500 Hz, giving my display a soft blue backlight.

You might have to scrounge pretty hard to come up with such a converter; mine came off a discarded laser-tag arcade game. Half the size of an ice cube, the converter's case carries the part number NEL-D32-45, but no manufacturer's name.

I have also seen a similar part offered in the Digi-Key catalog. The terminal works fine without the converter, of course, but backlighting adds a nice touch.

The Keyboard

I chose a surplus Cherry keyboard (very old, it uses TTL chips). This unit provides a 15-pin connector for hooking up to a computer. When you press a key,

the ASCII code associated with that character appears on seven pins of the connector, while a strobe pin goes from ground to +5 Volts, signaling there's valid data.

This means I only need ten wires to hook up my keyboard; seven data bits, one strobe line, power, and ground. I can use a single eight-bit input port to handle all my keyboard data. (Of course, the TTL chips make this keyboard a real power hog; oh well, it only cost \$4.)

With this arrangement, I get all the control codes, all upper-case letters, special characters, and numbers; no lower-case and no PC function-key sequences. Still, it'll serve my purpose.

There are always other options. For example, you could use a surplus, unencoded switch matrix. This type of keyboard consists of a grid of keyswitches, with the wires tied to each row and column brought out to a connector. Pressing a key shorts a specific row to a specific column; your software just scans each row (or column) until it detects a closed circuit. You can then use the row and column where the short occurred to determine which key was pressed.

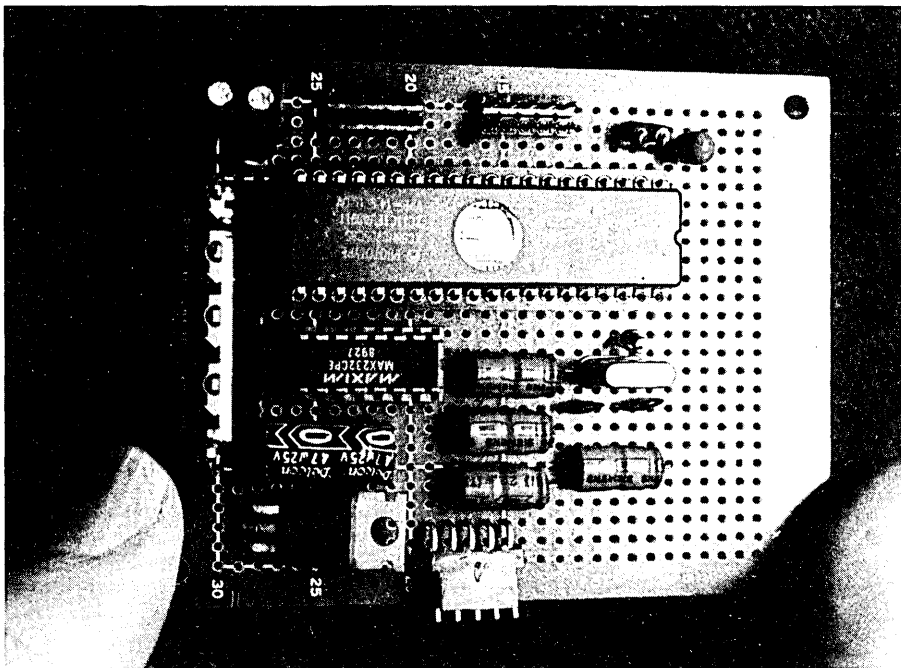
Though this sounds like extra software (it is), the scanned matrix keyboard offers some benefits. Since the matrix consumes no power, it makes a perfect choice for low-power applications. The surplus market seems glutted with a variety of unencoded keyboards. If you shop around, you will surely find just the right size, shape and style. And the price will be right; I picked up a brand-new unencoded keyboard from United Products for just a few dollars.

You could also opt for a PC-style keyboard. The Keytronics 101-style keyboards are usually available; I recently saw an ad from TimeLine that offered a new XT keyboard for \$15.

Unlike the other two keyboard systems, the PC units provide data for a

By Karl Lunt

2133 186th Pl., SE
Bothell, WA 98012
(206) 483-0447



With this project, you can get started in 68HC705 design while you build a versatile piece of computer gear.

pressed key via a serial line. A corresponding clock line tells the host computer exactly when each bit of the data packet is valid. (Check out Bill Curlew's article, "Building An IBM PC Keyboard Translator," in the February/March 1990 issue of *Circuit Cellar INK*, for details on PC keyboards.)

This synchronous transmission system means the keyboard can send (and, in some cases, receive) data over just three wires. At first glance, none of IBM's three available formats will work without additional logic. They just aren't compatible with the 68HC705's synchronous serial line (Motorola calls this port the serial peripheral interface, or SPI).

The problem lies with the number of bits transmitted per keypress. Mode 1 (the oldest and simplest format) uses eight data bits and one start bit for each transmission. The other two modes use

additional bits and support bidirectional transfers.

The SPI, however, can only accept a transfer of exactly eight data bits; it does not need nor expect a start bit. To hook a PC keyboard to the SPI, you could add a serial-to-parallel converter to the 68HC705. Or, you could put extra circuitry inside the keyboard to turn the nine bits into two eight-bit packets. Neither way was acceptable (I wanted this project to take only two chips), so the PC keyboard was out.

As I look that last paragraph over, I realize there is (as always) another way. Hooking the keyboard's clock line to an input port line and the keyboard's data line to the MCU's TCAP (timer capture) line would do the trick nicely.

Following a keypress, the data line's start bit going high-to-low would cause an interrupt and transfer control to the TCAP server routine. This routine

samples the clock line on the input port, looking for a low-to-high transition. Every time such a change occurs, the routine reads the level on the TCAP line to get the value for that bit-time. The server could then construct the keycode, wait for the stop bit to appear, and exit.

Okay, so you *can* add a PC keyboard to this project and still use only two chips.

The Maxim MAX232 Chip

I've built a couple of projects with this chip and love using it. Being able to run RS-232 from a single +5 Volt source reduces circuit complexity, chip count, and board size.

The Maxim chip works by generating +8 and -8 Volts on-chip from the system's +5 Volt supply, using a charge-pump technique. You hook up four 22 mfd capacitors for the charge-pump; the chip gives you two output and two input buffers that are RS-232 compatible.

You might run into some problems, however. The chip does not supply a lot of current to the RS-232 buffers. If you try to push 9600 baud over a long cable, you could get errors.

A new version of this chip, the

MAX233, puts the capacitors on the chip. This will further reduce the parts count for your RS-232 interface. But this chip suffers from the same low drive current as its MAX232 brother. At least one engineer I know has cautioned me about using either chip to talk distances.

Having said this, I'll add that I have yet to see any data errors directly due to the Maxim devices. Plus, both enjoy a wide following in the industry. I certainly intend to keep using them.

The MCU

Consider what it takes to make an RS-232 terminal. You need a bi-directional serial port good for 9600 baud and plenty of I/O lines to hook up to your LCD and keyboard.

You also need a timer system for properly clocking the control signals into the LCD, an interrupt system for catching the high-speed data flow on the serial line, and enough RAM and EPROM to hold your program.

Motorola's 68HC705 provides all this, at a current drain of less than 10 mA. Check out *Micro C* Issue #49 for an in-depth look at the 68HC705. For now, I will only discuss those aspects of the MCU needed by the two-chip terminal project.

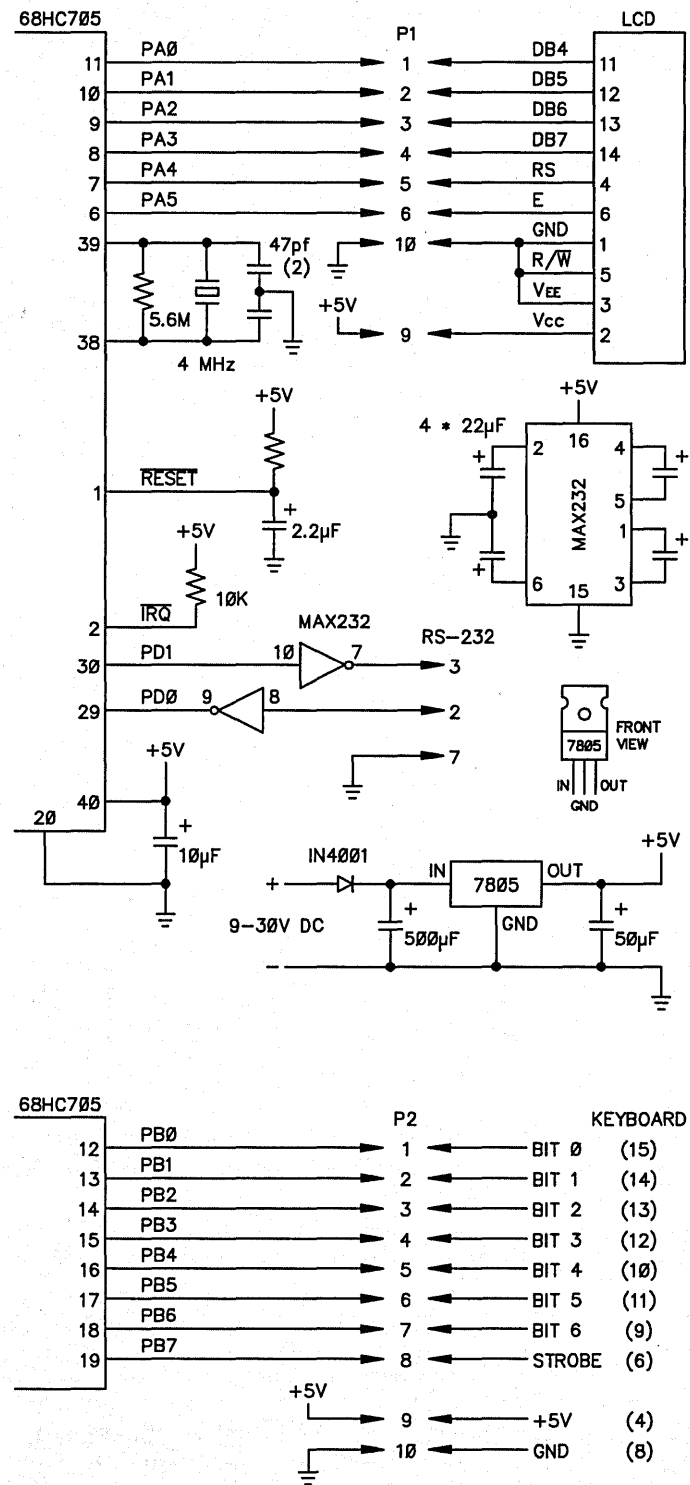
You control the extensive array of I/O and timing functions on the MCU by writing 8-bit values into specially assigned memory locations, called registers. For example, you can use all eight bits of I/O port A as outputs by storing the value \$FF into location \$04, known as DDRA (data direction register A).

By using a 4.0 MHz crystal, the MCU can hit 9600 baud on the serial port. Although the project currently does not support it, you could easily change baud rates from the keyboard or a row of dip switches. To alter baud rates, simply write different timing values to two baud-rate selector registers.

You must wait a prescribed amount of time after each change of the LCD. I use the MCU's 16-bit timer system to generate 200 μ sec time slices. The LCD routines simply wait until the correct number of slices (called tics) elapse before sending the next command.

Since the terminal could receive a stream of 9600 baud data, the MCU must handle the incoming data immediately. The MCU's serial communication interface (SCI) can generate an interrupt upon receiving a character from the serial port.

Figure 1—LCD and Keyboard Interface to '705



End diskette compatibility problems. Call Emerald Microware.

CompatiCard I by Micro Solutions

This four drive universal floppy controller will let you run up to 16 disk drives (4 per CompatiCard), including standard 360K, 96 TPI, high density 1.2M, 8" (SSSD or DSDD), and 720k/1.44M 3 1/2" drives. The CompatiCard I comes with its own MS-DOS driver, utility programs, and will let you boot on an XT (must be used as a secondary controller on an AT or 386). Use it with UniForm-PC for maximum versatility.

CompatiCard I Board \$ 119.95
CompatiCard I with UniFORM-PC \$ 179.95
8" drive adaptor board \$ 15.00
External drive cable set \$ 15.00

CompatiCard II by Micro Solutions

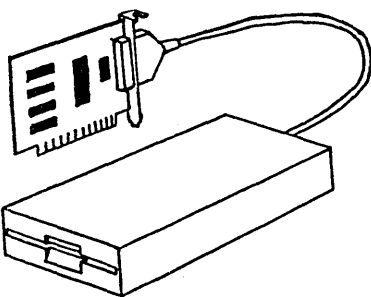
Two drive version of the CompatiCard, for the XT or AT. Same drive support as the CompatiCard I except no 8" or single density.

CompatiCard II \$ 89.95
*** Special *** CompatiCard II with
internal 1.2M or 1.44M drive \$ 199.95

CompatiCard IV by Micro Solutions

Meet the newest four drive controller in the CompatiCard family. This CompatiCard may be used as a primary or secondary controller in almost any PC, AT, or 386 System. Boot or use 360k, 720k, 1.2M, 1.44M, or 2.88M, at any location in your system. The CompatiCard IV has a BIOS ROM on board so no external driver software is required.

CompatiCard IV \$ 139.95



MegaMate by Micro Solutions

You don't have to be a computer expert to install this attractive 3 1/2" external drive on your PC or AT. Just plug the MegaMate controller board into any empty slot, attach the drive cable, run the installation software, and you're ready to run 720k or 1.44M diskettes.

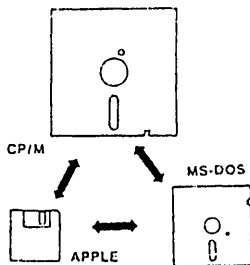
MegaMate \$ 329.95

Apple ⇄ MS-DOS

MatchPoint-PC by Micro Solutions

Apple II or NorthStar diskettes in your IBM? 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 UniForm-PC program, as well as utilities to format disks, copy, delete, and view files from Apple DOS, PRODOS, and Apple CP/M disks.

MatchPoint-PC Board \$ 179.95



UniForm-PC by Micro Solutions

Have you ever needed to use your CP/M diskettes on your PC? Now you can access your CP/M files and programs on your MS-DOS computer just as you would a standard MS-DOS diskette. UniForm allows you to 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, AT, OR 386. With UniForm-PC and the CompatiCard, you can use 5 1/4" high density, 96TPI, 3 1/2" (720k/1.44M), and even 8" drives.

UniForm-PC by Micro Solutions \$ 64.95
Also available for Kaypro, & other CP/M computers

CP/M ⇄ MS-DOS

UniDOS Z80 Coprocessor Board by Micro Solutions

Don't throw out all of those old, reliable CP/M programs, run them at LIGHTNING speed on your PC or AT with the UniDOS 8MHz. Z80 coprocessor board. And the UniDOS Z80 runs so smoothly and transparently that you won't even be able to tell whether you're running DOS or CP/M. UniDOS emulates most common computers and terminals such as Kaypro, Xerox 820, Morrow, Osborne, VT100, and many others. Supports all standard CP/M system calls, and now works with MS-DOS version 4. Includes UniForm-PC.

UniDOS Z80 Coprocessor Card \$ 169.95

UniDOS by Micro Solutions

If you have a fast machine or have a V20 chip installed, you may not need to use a card slot to run your CP/M programs. Run 8080 code directly on the V20, or use emulation mode for Z80 programs.

UniDOS by Micro Solutions \$ 64.95
UniDOS w/UniForm & V20-8 chip \$ 135.00

MatchMaker by Micro Solutions

Now you can copy your Macintosh diskettes right on your PC/XT/AT with the MatchMaker. Just plug your external 3 1/2" Macintosh drive into the MatchMaker board and experience EASY access to your Mac diskettes. Includes programs to read, write, initialize, and delete files on your single or double sided Mac disks.

MatchMaker Board \$ 139.95
MatchMaker w/External Mac Drive ... \$ 325.00

(503) 641-8088

Call or write for our complete catalog of software, parts, accessories and complete repair services for the Kaypro, Xerox 820, and IBM PC/AT.

Copy II PC by Central Point Software

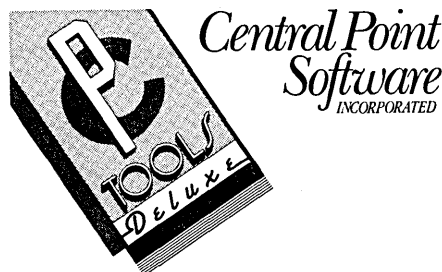
Don't let a damaged copy protected diskette stop you cold. Copy II PC lets you back up your master disks so you can keep going even when your key disk can't.

Copy II PC \$ 24.95

Copy II PC Deluxe Option Board by Central Point Software

Have a copy protected diskette with a particularly stubborn protection scheme? Would you like to be able to read your Macintosh disks in your 3 1/2" internal drive in your PC or AT? Repair a disk that's damaged, even between sectors? How about speeding up your hard disk backups (if you are using PC Tools Deluxe)? The Copy II Deluxe Option Board can help you do all of this, and more. A must for the sophisticated user.

Copy II Deluxe Option Board \$ 139.95



PC Tools Deluxe V5.5 by Central Point Software

This is one of the great bargains in MS-DOS utility software. But with so many features built in, we think that many people are overlooking the REAL value in PC Tools: DISASTER RECOVERY! Sure, the shell is great for copying, viewing, editing, and deleting files, and the desk top environment is nice for its appointment calendar, note pad, phone dialer, calculators, and ASCII table. The fast hard drive backups can't be beaten by any other program, and the file unfragmenter speeds up hard drive accesses. But where else can you get all of that along with UNDELETE, REBUILD, and UNFORMAT? Have you ever entered "ERASE *.*" and realized when all was said and done that you were in the wrong directory? How about those dreaded messages from CHKDSK, like "File allocation error..."? The very first time you recover your missing files will make you a believer. Don't wait until it happens.

PC Tools Deluxe V5.5 \$ 99.95



P.O. Box 1726
Beaverton, OR 97075

VISA and Mastercard accepted. Please include \$6.00 shipping and handling, \$8.50 for COD, UPS-Blue or RED Label additional according to weight. Prices subject to change without notice. Please include your phone number with all correspondence.

I use the SCI's interrupt to grab each character as it comes in, then stash the data in a circular queue. Eventually the mainline code gets around to pulling the character from the queue and processing it.

Data from the keyboard arrives much more slowly. (In my somewhat advanced years, I can no longer type much above 4800 baud.) The MCU simply glances from time to time at the keyboard port. If the line tied to the keyboard's strobe signal shows that data is available, the MCU grabs it.

Data from the terminal to the host computer also uses the SCI, but the program has no need to send anything in a steady stream. It (currently) sends a single character to the computer each time a key is pressed, so this function hardly requires interrupts.

On To The Software

I've divided the assembly language code into several major functions. The LCD routines initialize and update the display. The output compare (OCMP) server handles the 200 μ sec tic interrupts. The SCI server takes care of incoming data from the computer. The mainline code does everything else, such as processing keyboard data and keeping the display looking pretty.

I based my LCD library on information from Ed Nisley's article, "The True Secrets of Working with LCDs," from the April/May 1989 issue of *Circuit Cellar INK*. Ed's explanation of the LCD's inner workings really helped.

Because he did all his software in C, Ed didn't concern himself with the required delays between LCD commands; the C code ran slow enough all by itself.

Adding Delays

Writing in assembly language, however, means adding delays. These waits can be significant. The initialization ritual, for example, calls for delays as long as 5 msec; the shortest delay (between sending two visible characters to the display) still takes 120 μ sec. (Note: All LCD commands have a required minimum delay before a subsequent command will be correctly processed. However, there is no maximum delay.)

I solved this delay problem with the OCMP server and a global variable called WAIT\$.

The MCU's 16-bit timer continually counts down from \$FFFF, through \$0000, and starts over again at \$FFFF.

The output compare register (OCR) can hold an arbitrary 16-bit value. When the value of the free-running timer matches the value stored in the OCR, the MCU generates a timer interrupt.

To generate a fixed-length time slice, or tic, simply figure out how many timing counts correspond to the delay you need, add that value to the current value of the timer, and store the sum in the OCR. After the proper number of counts, the timer will reach the value you stored in the OCR and trigger an interrupt. Your interrupt service routine then calculates the sum for the next tic, updates the OCR....

In my case, I needed to count off the correct number of 200 μ sec tics. So my interrupt routine does more than just set up the next interrupt count. It also checks the value in WAIT\$; if that value is not yet zero, the routine decrements it.

Therefore, any routine that must wait (for example) 400 μ sec simply loads a value of 2 (to wait for two tics) into WAIT\$, then hangs around until WAIT\$ becomes 0.

You will notice my software only uses the A and X registers. That's all you get! The tiny programming model on this chip really makes you work to get the most out of your code.

More About LCDs

I connected my LCD to the terminal board using a 4-bit interface. Since the LCD interface also requires a register select line (RS) and an enable line (E), I could run the whole display with only six I/O lines. Refer to the schematic for details. Pay particular attention to the data lines. Note that the low four bits of the MCU's I/O port connect to the *high* four data bits on the LCD.

Unfortunately, the LCD comes out of reset in an 8-bit interface mode. You must follow a carefully defined ritual in setting the LCD to read the 4-bit interface. The code in LCDINIT does the job.

First off, I reset all the lines to the LCD and load a 15 msec power-up delay into WAIT\$. I then write a sequence of three commands, each resetting the LCD to an 8-bit interface. (Yes, I know we want a 4-bit interface, but you gotta do this just like the book says, or it isn't going to work.) Note the different delays used following each command.

Next, I set up the display for a 4-bit interface. The remaining commands fix the type of display format, cursor control, font, and cursor address. If you

want to try other display characteristics, check the Hitachi manual or Ed's article in *Circuit Cellar INK* for details.

The LCD accepts two types of data. Commands, such as those discussed above, must be written to the display with the RS line low. Data, such as displayed text, must be sent while RS is high. To make these operations easier, I included LCDCMD and LCDCHAR.

These routines use a common section of code, found at LCDCHAR1. Each sets up the RS line as needed, loads an appropriate delay value into the AR, and drops into the code at LCDCHAR1. From here, the byte gets divided into two 4-bit nybbles. Calls to LCDOUT then transfer the nybbles (MSB first) to the display.

LCDSTRING simply sends each character of a null-terminated string to the display. The only tricky bit involves the load indexed, indirect addressing mode of the 68HC705; the chip doesn't have such a mode.

To provide this essential addressing mode, I resorted to self-modifying code (actually, a RAM-based subroutine, which is just as bad).

The initialization code for the terminal program builds up a two-instruction subroutine called LDAIND by writing two opcodes into low RAM. Address LDAIND holds a \$D6 (load accumulator, indexed, with a 16-bit offset), while address LDAIND+3 holds a \$81 (return from subroutine). The two bytes at LDAIND+1 and LDAIND+2 get changed by another routine before calling LDAIND.

At run-time, a section of code stores the 16-bit address of a null-terminated string into the two bytes at LDAIND+1. It clears the XR (X index register), then calls LCDSTRING. Since the LDAIND subroutine is all set up, LCDSTRING can simply call the routine to get the character in the string pointed to by the XR. It then increments the XR, gets the next character, etc., until it gets the null marking the string's end.

And Now The Hardware

I built the prototype terminal on a Radio Shack experimenter's card (276-158A). You can find nearly all the parts in a Radio Shack or Active Electronics catalog; write these companies for a catalog if you don't have one already.

The only part you might have trouble with is the 68HC705 MCU. You will need to contact a Motorola distributor

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)	\$750
CQL Query System (SQL retrievals on B-trees plus windows)	\$325
GraphiC 5.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	\$325
PC Courses (Aspen, Software, System V compatible, extensive documentation)	\$290
C-Data Manager (object-oriented data management, persistent objects from runtime definitions, network and entity models)	\$250
MEWEL (extensible window and event library by Magma Software; message-passing & object-oriented; SAA-compatible; dialog editor)	\$250
TurboTeX (Release 2.0; HP, PS, dot drivers; CM fonts; LaTeX; MetaFont)	\$250
db.File & db.Retrieve by Raima (B-tree and network database with SQL query and report writer; multi-user \$475)	\$245
Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF; specify compiler)	\$225
CDirect (multi-user hashed file manager; variable length fields, binary or ASCII data, alternate keys)	\$210
SilverComm (complete asynchronous communications library)	\$210
Wendin-DOS Plus (self-bootable, multitasking, multiuser MS-DOS replacement; includes XTC editor)	\$180
QuickGeometry Library (large collection of mathematics, graphics, display & DXF subroutines for CAD/CAM/CAE/CNC)	\$170
CBTree (B+tree ISAM driver, multiple variable-length keys)	\$165
TurboGeometry (library of routines for computational geometry, Version 3.0)	\$160
AT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for ATs)	\$160
WKS Library Version 2.01 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	\$155
C Generator (generates C code to read & write file records defined with C structure syntax)	\$150
OS/88 (industrial-strength U*xx-like operating system, many tools, cross-development from MS-DOS)	\$150
Cephes Mathematical Library (over 100 high-quality, double-precision scientific functions)	\$150
ME Version 2.1 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still \$75)	\$140
Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	\$140
Turbo G Graphics Library (all popular adapters, hidden line removal)	\$135
Rogue Wave Vector & Matrix Classes (inc. C++ overloads for standard operators, matrix inversion & FFT; Zortech or GNU C++)	\$125
Power Search by Blaise Computing (regular-expression compiler; generates machine code on the fly)	\$120
Install 2.3 (automatic installation program; user-selected partial installation; CRC checking)	\$120
TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	\$115
NEW! Hold Everything (spawn new programs; swap parent to EMS or disk; handles video, interrupts, & environment; returns error level)	\$105
B-Strings (dynamic string handling; cut, copy, paste, search, user input, etc.; non-fragmenting memory management)	\$105
Minix Operating System (Version 1.3; U*xx-like operating system, includes manual)	\$105
PC/IP (CMU/MIT TCP/IP for PCs; Ethernet, Appletalk & NETBIOS drivers, RVD, gateways)	\$100
B-Tree Library & ISAM Driver (file system utilities by Sofifocus)	\$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
Booter Toolkit (floppy disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	\$85
Otter 1.0 (beautiful theorem-prover by Bill McCune; includes manual & two books by Wos; complete starter kit)	\$80
JATE Async Terminal Emulator (includes file transfer and menu subsystem)	\$80
PowerSTOR (upto a gigabyte of heap space on extended memory, expanded memory, and/or hard disk)	\$80
MultIDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	\$80
HY-PHEN-EX (a hyphenator for American English with over 4,800 rules)	\$75
Make (macros, all languages, built-in rules)	\$75
evalO (C function to evaluate ASCII infix expression string; 17 built-in functions)	\$75
XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XT's)	\$75
Professional C Windows (lean & mean window and keyboard handler)	\$70
Heap Expander (virtual memory manager using expanded memory, extended memory, and disk space)	\$65
Quincy (interactive C interpreter)	\$60
Symtab/Ptree (general-purpose symbol table/parse tree construction and management package; specify Symtab or Ptree)	\$60
Coder's Prolog (Version 3.0; inference engine for use with C programs)	\$60
Async-Termio (Unix V compatible serial interface for MS-DOS; stty, ioctl, SIGINT, etc.)	\$55
Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	\$50
Floppy TAR (TAR backup and restore on MS-DOS devices; direct access to non-standard devices)	\$50
SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	\$50
OBJASM (convert .obj files to .asm files; output is MASM compatible)	\$50
Multi-User BBS (chat, mail, menus, sysop displays; does not include Hayes modem driver)	\$50
LaplaceB (LaPlace polynomials, real and complex)	\$50
CLIPS (rule-based expert system generator, Version 4.3; advanced manuals available)	\$50
Upgrade! NEW! Pascal P-Code Compiler & Interpreter (full ISO standard Pascal)	\$50
PCHRT (40 functions to manage multiple microsecond timers; generate precision delays; insert timers on any interrupt)	\$45
Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	\$45
Fortran-to-C Translator by Polyplot (Fortran-IV-like Fortran to ugly C; plan to adapt to your own flavor of Fortran)	\$40
DES Encryption & Decryption (2500 bits/second on 4.77 MHz PC for on-the-fly encryption at 2400 baud)	\$40
FlexList (doubly-linked lists of arbitrary data with multiple access methods)	\$40
Virtual Memory Manager by Blake McBride (LRU pager, dynamic swap file, image save/restore)	\$40
Heap I/O (treat all or part of a disk file as heap storage)	\$40
Bison & BYACC (YACC workalike parser generators; documentation; no restrictions on use of BYACC output)	\$35
PC-XINU (Comer's XINU operating system for PC)	\$35
RXC & EGREP (Regular Expression Compiler and Pattern Matching; RXC makes finite state machine from regular expression)	\$35
Cheaper! REGX Plus (search and replace string manipulation routines based on regular expressions)	\$30
CCALC (handy extended-precision calculator; real and complex models; many built-in functions)	\$30
GNU Awk & Diff for PC (both programs in one package)	\$30
6-Pack of Editors (baker's half-dozen public domain editors for use, study & hacking; includes microEmacs 3.10 & Stevie, a vi clone)	\$30
Crunch Pack (14 file compression & expansion programs)	\$30
NEW! PC-MAIL (UUCP mailer by Wietse Z. Venema; send, receive, and manage UUCP mail)	\$25
FLEX (fast lexical analyzer generator; new, improved LEX; official BSD Version 2.1 with docs)	\$25
List-Pac (C functions for lists, stacks, and queues)	\$25
Using C++ Library (the code from the book by Bruce Eckel and then some; Zortech 2.0 compatible)	\$25
A68 (68000 cross-assembler)	\$20
XLT Macro Processor (general purpose text translator)	\$20
Data	
Moby Pronunciator (150,000 words & phrases encoded with full IPA pronunciation & emphasis points; 900 distinguished by part-of-speech)	\$160
Moby Part-of-Speech (200,000 words and phrases described by prioritized part(s)-of-speech)	\$120
Moby Hyphenator (150,000 words fully hyphenated/syllabified)	\$105
Moby Words (500,000 words & phrases, 9,000 stars, 15,000 names)	\$65
Smithsonian Astronomical Observatory Subset (right ascension, declination, & magnitude of 258,997 stars)	\$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
Interactive Computer Ephemeris (high-precision moon, sun, planet & star positions; USNO (no source) & Downey 4.8 (C source))	\$30
<i>The Austin Code Works</i>	<i>Voice: (512) 258-0785</i>
11100 Leafwood Lane	<i>info@acw.com</i>
Austin, Texas 78750-3409 USA	<i>BBS: (512) 258-8831</i>
	<i>FAX: (512) 258-1342</i>
Free surface shipping for cash in advance	For delivery in Texas add 7%
	MasterCard/VISA

directly for this chip, as I don't think it is in the mail-order houses yet. Almac Electronics (206-643-9992) in Bellevue, Washington, will handle the parts mail-order in singles; price for the -S version of this part (with the EPROM window) is about \$18.

Be sure to get a copy of Motorola's HC705 C8 Technical Data book, literature number MC68HC705C8/D. Besides all the technical info on using the 'HC705, this book includes a full schematic for building a programming board (piece of cake). It also includes instructions on using the programming board.

You can use just about any wiring technique you want for this project. I used point-to-point soldering with 30 AWG wire; wirewrapping would work also.

By the time you read this, I will have a general-purpose experimenter's printed wiring board (PWB) for the 68HC705. The board will contain all the basic circuitry needed to build a working MCU project, and could easily be the only PWB you might need to develop an application.

For example, this two-chip terminal project would consist of a working ex-

perimenter's PWB connected to a keyboard and LCD.

After you get your terminal wired up, you need to assemble the source code. I use a PC/XT clone running Motorola's AS5 Freeware cross-assembler. Borland's Sidekick handles my editing, and I burn the finished code into the MCU using Motorola's M68HC05PGMR programmer board and PROG7 support software.

You can download both PROG7 and AS5 from the Motorola Freeware BBS (512-891-3733; 8 data, no parity, 1 stop). Motorola uses the BBS to distribute free cross-assemblers (available for both the PC and Mac), working source code for MCU designs, technical updates, and marketing announcements. Dial up and take a look.

Even if you don't yet have a keyboard, you can still hook up the display and apply power (use a DC source from +7 to +25 Volts). Your LCD should show the power-up announcement "Working..."

If you don't get the proper display, use an oscilloscope or logic probe to check pin 35 of the MCU. My software toggles this line each tic; you should see a 50% duty-cycle square wave that changes state every 200 usecs. If you see this signal, you likely have an error in your LCD connection.

If you don't see this signal, check your board's power supply and oscillator circuitry; you may not be getting +5 Volts to the chip, or your crystal may not be oscillating.

After you get the display working, wire up your keyboard using the schematic as a guide. The software expects you to supply seven data bits (on port B, bits 0-6) and a positive-going strobe (port B, bit 7); you can use any keyboard that will supply these signals. Note that the circuitry does not supply +12 Volts; keep that in mind when you hit the surplus stores.

With your keyboard installed, temporarily short pins 2 and 3 of the RS-232 connector. When you apply power and press a key, the display should echo that key. If so, you have a working terminal!

Time To Tinker

Since everything about this terminal can be found in the software, you can really customize this project. Consult the Hitachi manual for the low-level LCD commands if you want to change the shape/blink of the cursor. The Maxim

chip has a pair of unused RS-232 buffers; you could hook these up to I/O pins on the MCU and provide hardware handshaking on the serial line.

Since a keyboard character goes through the MCU before it gets shipped out the serial port, you can easily imbed function codes or signature keys into the software.

This design leaves an entire 8-bit port vacant; how about hooking up relays, lights, a piezo-beeper (for a control-G bell code), or other control devices? You could even hook up an external A/D converter and have a stand-alone environmental control station. The station could then download data over the serial port to a host computer, while the keyboard lets you run programs on the host for evaluating data.

Anyone doing field work would probably find a use for this terminal. How about hooking up some rechargeable batteries (or a solar panel, if you use a switch matrix keyboard) and taking the terminal outdoors?

If you run road rallies, you could easily connect the terminal (complete with backlighting) to a small rally computer stashed in the backseat, and do some real digital navigation.

That's About It

This project was great fun. Everything I develop with the '705 makes me want to try another design or two. If you customize this project (as I am sure you will), I'd like to hear what you have done.

Active Electronics

P.O. Box 9100
Westborough, MA 01581
(800) 888-9939 (outside New England)
(800) ACTIVE6 (New England)

TimeLine Inc.

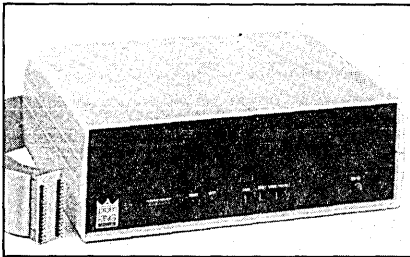
1490 W. Artesia Blvd.
Gardena, CA 90247
(800) 872-8878 (outside California)
(800) 223-9977 (California)

United Products

1123 Valley Street
Seattle, WA 98109-4425
(206) 682-5025



STOMP OUT EPROM MADNESS



The PROM KING emulates EPROMS, saving both time and money during your development cycle. Programmable in seconds via your PC printer port or any computer RS232 port, it can emulate most 27xxx devices.

- 8K-8M bit devices
- 8-256 bit downloads
- High speed download:
- Easily expandable:
- Universal RS232
- 4 EPROMS per unit
- PC printer port
- Up to 8 units
- Menu driven software
- Also programs like
- Battery backup
- a real EPROM

\$599 for 150nS units with 256K bits
Ask for pricing of other options

Made in USA by:

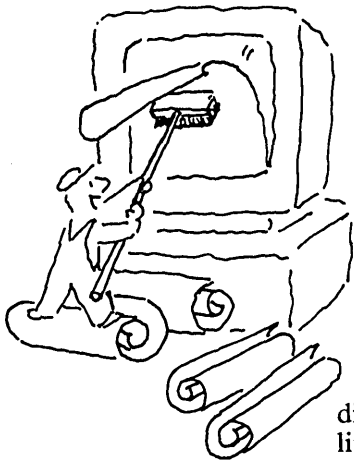
TRAXEL LABS INC.
BOX 239 • RONKONKOMA, NY • 11779
516-737-5147

Reader Service Number 178

Two great tools.

NEW VERSION 2.8
CLOCK & EDIT UNITS, DIALOG
BOXES, AND PROGRESS BARS!

SAYWHAT?! The lightning-fast screen generator.



Whether you're a novice programmer longing for simplicity, or a seasoned pro searching for higher productivity, you owe it to yourself to check out Saywhat. You see, with Saywhat, you can build beautiful, elaborate, color-coded screens in minutes! That's right. Truly *fantastic* screens for menus, data entry, data display, and help-panels that can be displayed with as little as one line of code in *any* language.

Here's what you get:

- Design screens, windows, and *moving bar menus!*
- Easy-to-use, powerful editor lets you create screens in a jiffy.
- Pop up your screens and menus with one line of code in dBASE, all the dBASE compilers, your favorite BASIC, Pascal, or *any other language!*
- Screen Library Manager.
- Generates runtime code.
- No runtime license or royalty fees.
- Comes with a 100 page manual, plus dozens of sample programs and free utilities.

\$49⁹⁵



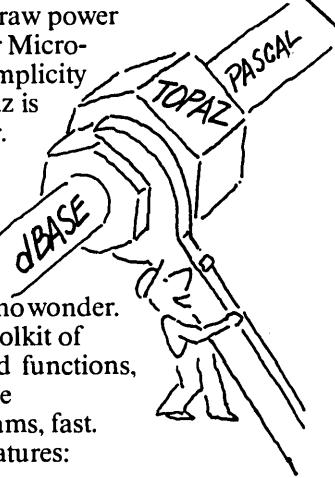
MONEY BACK GUARANTEE.

If you aren't completely delighted with Saywhat or Topaz, for any reason, return them within 30 days for a prompt, friendly refund.

Dealers: SAYWHAT?! and TOPAZ are available from Kenfil Distribution, and in Europe from ComFood Software, W. Germany 49-2534-7093

TOPAZ The breakthrough DBMS toolkit for Pascal

If you'd like to combine the raw power and speed of Turbo Pascal or Microsoft's QuickPascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. That's because Topaz was specially created to let you enjoy the best of *both* worlds. The result? You create complete, truly dazzling applications in a very short time. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you produce outstanding, polished programs, fast. Check out these powerful features:



- Over 220 routines all with easy-to-use, dBASE-like syntax.
- Data entry routines like SAY, GET, PICTURE, RANGE, color selection, unlimited data validation.
- Open up to 10 DBF files, with up to 15 indexes with USE, SELECT, SKIP, APPEND, PACK, INDEX ON, SET INDEX TO, and FIND.
- No need to buy dBASE. CREATE, BROWSE and REPORT utilities included.
- Easily implement Saywhat and Lotus-style moving bar menus.
- BROWSE or EDIT any DBF file with just one line of code! Programmable and windowed too.
- Pick from windowed data or file-names with one line of code.
- Comprehensive Time & Date math in 7 international formats.
- Powerful code and report generators included!
- Comes with a complete 320 page manual, plus sample programs to get you started.

\$74⁹⁵

Guaranteed!

S O F T W A R E S C I E N C E I N C .

Reader Service Number 129

MICRO CORNUCOPIA, #53, May, 1990 33

A Roundoff Roundup

When Rational Numbers Aren't Really Rational

Are rational numbers really rational? Perhaps not. Does it matter? Most likely.

Gary Entsminger, in his Tidbits column "Faith in Numbers: Chaos in Chaos" (*Micro C* #48, July-August 1989), discusses the problem of roundoff error in recursive iterations of non-linear equations. Mr. Entsminger points out a problem facing computer mathematics which, at least for some applications, has a simple solution. This article makes several references to Mr. Entsminger's column, so you may wish to reread it.

Mr. Entsminger defines a rational number as a real number which is either reducible to an integer, expressible as a terminating decimal fraction, or expressible as a decimal fraction with a repeating sequence of digits. This is a common and workable definition, but not an exact one.

Take an ordered pair (x,y) of integers and define an equivalence relation $=$ such that for any two ordered pairs (x,y) and (a,b) with $x < a$ and $y < b$, $(x,y) = (a,b)$ if and only if there exists an integer z where $az = x$ and $bz = y$. Think of multiplying both the numerator and denominator of a fraction by the same number. Then define two binary operations $*$ and $+$ by:

$$(x, y) * (a, b) = (xa, yb)$$

$$(x, y) + (a, b) = (xb + ay, yb)$$

This set of all ordered pairs of integers, in conjunction with the relation $=$ and these two operations, is the number-theoretic definition of the rational numbers. Think of x as the numerator and y as the denominator of a fraction.

This means I can represent a rational number as an array of integers. Compare this to floating point. Longint in Turbo Pascal is 32 bits long and can represent

2^{33} unique integers. If I use two of these to represent a rational number, I can represent 2^{33} integer values with $(2^{33} - 1)/2$ divisions between each integer. (I cannot have zero for a denominator, and negative values in the denominator cancel out negatives in the numerator.)

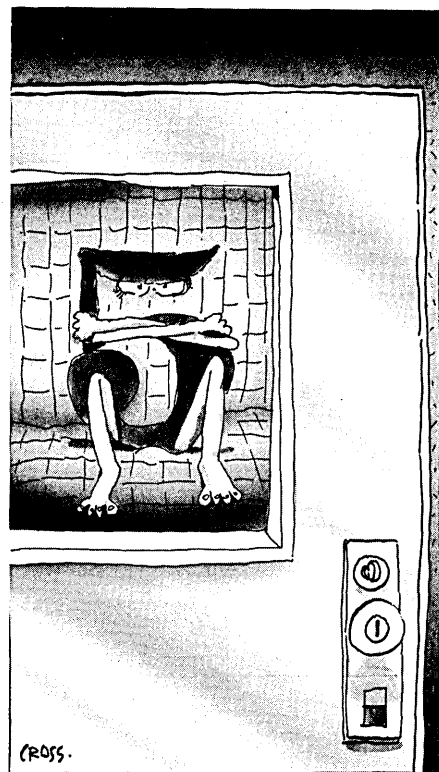
I can then represent $2^{65} - 2^{32}$ numbers, with some repetitions. Using the same number of bits in floating point notation, I can represent 2^{66} different values with about the same number of repetitions, so floating point notation can represent a much larger set of numbers.

There is a tradeoff. No matter how many bits I use, floating point can never represent some simple rational numbers accurately. Take the rational number $1/3$ for example; there are no integers a and b for which $1/3 = a + 2^b$. The entire range of numbers represented by floating point notation is riddled with numbers which can't be represented exactly, namely any simplified rational number m/n where n cannot be represented as a sum of non-zero powers of two.

Mr. Entsminger states that recursive iterations of non-linear equations like $\text{Nextx} = \text{RX}(1 - \text{X})$ must "...surely encounter at least one irrational number." Since the rationals are closed under multiplication and addition, this statement, as written, is false. Even if we consider roundoff error, this error is due to a finite number of decimal places, and any terminating decimal fraction is a rational. But I think Mr. Entsminger is attempting to express something more complex than fundamental concepts of basic algebra.

We call a real number z "rational" if it can be "rationalized": if we can find integers x and y such that $x/y = z$. There is a similar process which seems natural for floating point notation. We call a real number z "rational," as a floating point number, if we can find integers a and b for which $z = a + 2^b$. Then, in floating

No matter how many bits I use, floating point notation can never represent some simple rational numbers accurately.



By Lance Dannon Bresee

University of California Observatory
Lick Observatory Electronics Lab
University of California at Santa Cruz
1156 High Street
Santa Cruz, CA 95064

Figure 1—RATIONAL.PAS

```
Unit Rationals;
Interface

Type
  Rational = Array[1..2] of longint;

Function Prime( N : longint ) : boolean;
  {Returns TRUE if integer N is Prime and False otherwise}
Function GCD( A,B : longint): longint;
  {Returns the greatest common divisor of A and B}
Function LCM( A,B : longint): longint;
  {Returns the least common multiple of A and B}
Procedure Reduce( Var X : Rational);
  {Reduces rational X to simple form}
Procedure Invert( Var X : Rational);
  {Inverts rational X}
Procedure Radd( X1,X2 : rational; Var Y : Rational);
  {Returns the reduced sum of X1 and X2 in Y}
Procedure Rsub(X1,X2 : Rational; Var Y : Rational);
  {Returns the difference of X1 - X2 in Y}
Procedure Rmult(X1,X2 : Rational; Var Y : Rational);
  {Returns the product of X1 and X2 in Y}
Procedure Rdiv(X1,X2 : Rational; Var Y : Rational);
  {Returns the result of X1/X2 in Y }

Implementation
Function Prime( N : longint ) : boolean;
Var
  Pr, Done : boolean;
  Divisor : longint;
  Count : byte;
Begin {Prime}
  Pr := False;
  Done := False;
  Divisor := 3;
  Count := 0;
  If N <= 10 then
    begin {if}
      Done := true;
      If N in [1,2,3,5,7] then Pr := true;
    end {if}
  end
```

Continued on page 36

point notation, $\frac{1}{3}$ is "irrational." Because of the large number of these "irrational" numbers, computer science has, by adopting floating point notation for all non-integral numbers, wandered up something of a blind alley.

Using floating point notation, if I increase the number of bits I use in my representation, I increase the range of rational numbers I can represent. But I still cannot accurately represent any of these "irrational" rationals. In fact, I will encounter more of these "irrationals" in my broader set. If, however, I represent rational numbers as ordered pairs of integers, increasing the number of bits I use increases both the range and accuracy.

The unit RATIONAL.PAS (see Figure 1) defines a rational number as an array of integers and defines several operations on rationals. I have not included error checking for overflow problems because I rarely use numbers large or small enough to cause problems that frequent simplification cannot remedy.

There are also no functions for testing for a zero or negative in the denominator. Since multiplication is the only operation performed on denominators, these are not needed. The only exception is in the INVERT procedure.

The program GAUSS.PAS (see Figure 2) uses both reals and rationals to do a Gauss elimination on a system of two equations in two unknowns. Run this program yourself with different coefficients and notice the difference in results. Try coefficients like 3, 6, and 21, as well as primes.

Mr. Entsminger points out that no system is capable of representing rational numbers to an unlimited significance. No matter which representation I use, I can only represent a finite, discrete set of numbers. Since one of the most signifi-

Continued on page 39

```

else If (N mod 10) in [0,2,4,5,6,8] then
  done := true;
While not Done do
  If Divisor >= N then begin
    Pr := true;
    Done := true;
  end
  else if ((N mod Divisor) = 0) then
    Done := true
  else begin
    Count := (Count + 1) mod 4;
    If count = 1 then
      divisor := divisor + 4
    else divisor := divisor + 2;
    end;
    Prime := Pr;
End; {Prime}

```

```

Function GCD ( A,B : longint): longint;
Var
  x,M,N : longint;
  Done : boolean;
Begin
  M := abs(A);
  N := abs(B);
  x := M;
  If N <= x then x := N;
  Done := false;
  While not done do
    If ((N mod X)=0) and ((M mod X)=0) then
      done:=true
    else x := x - 1;
  GCD := x;
End;

```

```

Function LCM ( A,B : longint ): longint;
Var
  X,Y,N,M : longint;
  Done : boolean;
Begin
  M := abs(A);
  N := abs(B);
  Y := M;
  If N < Y then Y := N;
  X := Y;
  Done := false;
  While not done do
    If ((X mod M) = 0) and ((X mod N) = 0) then
      done := true
    else x := x + Y;
  LCM := X;
End;

```

```

Procedure Reduce( Var X : rational);
Var
  A : longint;
Begin

```

```

If X[1] = 0 then begin
  X[1] := 0;
  X[2] := 1;
end
else begin
  A := GCD(X[1],X[2]);
  X[1] := X[1] div A;
  X[2] := X[2] div A;
end;
End;

Procedure Invert( Var X : Rational);
Var
  A : longint;
Begin
  If X[1] = 0 then
    Writeln('DIVIDE BY ZERO ERROR!')
  else if X[1] < 0 then begin
    A := X[2];
    X[2] := Abs(X[1]);
    X[1] := 0 - A;
  end
  else begin
    A := X[1];
    X[1] := X[2];
    X[2] := A;
  end;
End;

```

```

Procedure Radd( X1,X2: rational; Var Y: rational);
Var
  Z,A : longint;
Begin
  Z := LCM(X1[2],X2[2]);
  Y[2] := Z;
  A := Z div X1[2];
  Y[1] := X1[1] * A;
  A := Z div X2[2];
  Y[1] := Y[1] + ( X2[1] * A);
  Reduce(Y);
End;

```

```

Procedure Rsub( X1,X2: rational; Var Y: rational);
Var
  Z,A : longint;
Begin
  Z := LCM(X1[2],X2[2]);
  Y[2] := Z;
  A := Z div X1[2];
  Y[1] := X1[1] * A;
  A := Z div X2[2];
  Y[1] := Y[1] - (X2[1] * A);
  Reduce(Y);
End;

```

```

Procedure Rmult(X1,X2: Rational; Var Y: Rational);
Var
  A,B : Rational;

```

KNOWLEDGE=POWER

Dis•Doc v3.xx- REALMODE.EXE

Help	Load	Format	Edit	Search	Output	Patch	Xoption	Dos	Keep	Quit
Format: All				P	<16>	no patches				
	call	sl		-string-						
	mov	ax		Text						
				Binary						
	mov	ds		-module-						
	mov	al		Subroutine						
	xor	si		Main						
				Origin						
	mov	cx								
	mov	bx								
:;>>>> Conversion Section				Xref address						
	les	di		Word	J					
				Address						
	lfs	bx			b					
	movzx	cx		Help						
	repz	st								
	sti									
	mov	ax,4c00h								
	int	21h								

Unlimited file size. . . Fully automatic

Batch mode and interactive

Locates Data/Code boundaries

Built-in BIOS Preprocessor

All Data formats including DB, DW, DD & DUP

EXE Unpacker included!

No Source Code? No Problem.

New **Dis•Doc Professional** is your dual-mode key to any DOS source code. It works in batch and interactive modes simultaneously, allowing you to generate the core information of even the most complex programs **fast**. . . and modify them even faster. Most programs will come apart in just two minutes. Imagine what you can do with a tool this powerful! **Dis•Doc** sifts through programs **eight times** for guaranteed accuracy. When code gets mixed up with data, our toolbox comes to the rescue with **smart** search and **easy** edit utilities. **Dis•Doc** can handle any instruction set up to and including 486 and offers a variety of other great features that you can sample on our **Free Demo Disk**.

Warning: Dis•Doc Professional may change the way you work forever.

Programmers who used to shy away from fixing outmoded programs with no source code are going to discover a **valuable** new talent: the ability to modify and revise codes that would cost way

too much to start over (it's a programming manager's dream). Save your employer huge new-programming fees and enhance your marketability. **Dis•Doc** is so easy to learn, you'll be a high-dollar hero in no time!

Knowledge really *is* power.

Dis•Doc Professional is an amazing new teaching tool. Learn how programs work. . . take them apart and see the writing techniques that top pros use. Use it to assist in debugging. Hunt down viruses and write killers. **Dis•Doc** can save you **years** of frustration, and it only costs **\$149.95** including the EXE Unpacker (until January 1). To order your **Dis•Doc Professional** kit or our free demo disk, simply call:

1-800-446-4656

WITHIN CT & OUTSIDE THE U.S., CALL (203) 953-0236

MasterCard & VISA • Shipped Immediately Via UPS
RJSwantek, Inc., 178 Brookside Road • Newington CT 06111

```

Begin
  A[1] := X1[1];
  A[2] := X2[2];
  B[1] := X2[1];
  B[2] := X1[2];
  Reduce(A);
  Reduce(B);
  Y[1] := A[1] * B[1];
  Y[2] := A[2] * B[2];
  Reduce(Y);
End;

Procedure Rdiv(X1,X2: Rational; Var Y: Rational);
Var
  X : Rational;
Begin
  X := X2;
  Invert(X);
  Rmult(X1,X,Y);
End;
End.

```

◆◆◆

Figure 2—GAUSS.PAS

```

Program GAUSS;
Uses RATIONALS;
Var
  A,B,C,D,E,F,X,Y,Z : Rational;
  Ar,Br,Cr,Dr,Er,Fr,Xr,Yr,Zr : Real;
Begin
  A[1] := 131;
  A[2] := 1;
  Ar := 131;
  B[1] := 23;
  B[2] := 1;
  Br := 23;
  C[1] := 31;
  C[2] := 1;
  Cr := 31;
  D[1] := 7;
  D[2] := 1;
  Dr := 7;
  E[1] := 27;
  E[2] := 1;
  Er := 21;
  F[1] := 67;
  F[2] := 1;
  Fr := 67;
  {Using Rationals}{First divide 1 by a}
  X := A;
  Rdiv(a,x,y);
  a := y;
  Rdiv(b,x,y);
  b := y;
  Rdiv(c,x,y);
  c := y;

```

```

{Subtract D*1 from 2}
x := d;
Rmult(x,a,y);
Rsub(d,y,z);
d := z;
Rmult(x,b,y);
Rsub(e,y,z);
e := z;
Rmult(x,c,y);
Rsub(f,y,z);
f := z;
{Divide 2 by e}
x := e;
Rdiv(e,x,y);
e := y;
Rdiv(f,x,y);
f := y;
{Subtract b*2 from 1}
x := b;
Rmult(e,x,y);
Rsub(b,y,z);
b := z;
Rmult(f,x,y);
Rsub(c,y,z);
c := z;
Writeln('Using rationals we get...');
Writeln(a[1],'/',a[2],'X = ',c[1],'/',c[2]);
Writeln(e[1],'/',e[2],'Y = ',f[1],'/',f[2]);
writeln;
{Using reals}{Divide 1 by a}
xr := ar;
ar := ar / xr;
br := br / xr;
cr := cr / xr;
{Subtract D*1 from 2}
xr := ar * dr;
yr := br * dr;
zr := cr * dr;
dr := dr - xr;
er := er - yr;
fr := fr - zr;
{Divide 2 by e}
xr := er;
dr := dr / xr;
er := er / xr;
fr := fr / xr;
{Subtract B*2 from 1}
xr := br * dx;
yr := br * ex;
zr := br * fr;
ar := ar - xr;
br := br - yr;
cr := cr - zr;
Writeln(' Using REALS we get');
Writeln;Writeln(ar:5:9,' *X = ',cr:5:9);
Writeln(er:5:9,' *Y = ',fr:5:9);
End.

```

◆◆◆

cant properties of the rational numbers is that given any two rationals a and b, there must be a rational z such that $a < z < b$, I can never represent a complete range of the rationals. In modeling natural systems, I do not need to.

The set of all numbers which exist in nature is discrete. In making such a bold claim, I feel compelled to defend it.

Using the most accurate measuring instruments available, and the best techniques of extrapolation, you can measure a quantity to only finite tolerance. You will then be able to get two measures for which no intermediate measure can be found.

Consider Zeno's paradox. Zeno asserts that if a turtle and a rabbit were to run a race, and the turtle were given a head start, the rabbit could never pass the turtle. His reasoning is that when the rabbit achieved the turtle's starting position, the turtle would have moved some distance ahead. The distance between the two, though less, would still be positive. Continuing in this manner, Zeno says that the turtle must always be some positive distance ahead.

This is true only if distance is infinitely divisible. Try this experiment yourself with two objects moving at constant velocity, one faster than the other. Give the slow one a head start and you will see that the fast one does pass the slow one. The distance between them reduces to a point where no smaller distance exists.

Mr. Entsminger's example of the equation $Nextx + RX(1 - X)$ yields values between 0 and 1, representing some portion of the maximum population of wolves. If the maximum number of wolves is some number z and the population after any iteration is given by $z * Nextx$, we see that values of $Nextx$ of minute enough significance are not useful. Suppose $Nextx = (K(z + 1) + 1)/(z + 1)$ for some rational K. We then have a fraction of a wolf, which is not likely to reproduce.

For many applications, all discrete representations of the rational numbers are equally flawed. Trigonometric equations, for example, rely on the irrational number pi. The results obtained using ordered pairs will be at least as inaccurate as those obtained using floating point notation. But for many applications, representing rational numbers as ordered pairs of integers will greatly increase accuracy.



CITIZEN PRINTERS

Citizen 120D	120/25 CPS	80 COL	\$165.00
Citizen 180D	180/35 CPS	80 COL	\$180.00
Citizen HSP500	300/66 CPS	80	\$350.00
Citizen HSP550	300/66 CPS	120	\$475.00
Citizen GSX140	200 CPS	24 PIN	\$345.00

GOLDSTAR MONITORS

MBM-1210A	12" TTL	AMBER	\$ 90.00
MBM-1401A	14" Flar	Screen TTL	\$135.00
MCH-1420	14" EGA	640x350	\$345.00
MCH-1430	14" VGA	640x480	\$375.00

VIDEO CARDS

ATI VIP Card	VGA,CGA,EGA,MGA,		
	ANALOG AND TTL	800x560	\$175.00
ATI VGA Wonder	256	1024x768	\$225.00
ATI VGA Wonder	256	16 bit	\$375.00
Paradise Basic	EGA	640x350	\$ 99.00
Paradise Basic	VGA	640x480	\$175.00
Paradise VGA Plus		800x600	\$199.00
Paridise VGA Plus		16 Bit	\$275.00

HARD DRIVE CONTROLLERS DOR XT & AT

WD WX1 Auto Config	PC/XT	\$ 65.00
WD WX2 Auto Config	PC/XT	\$ 55.00
LONGSHINE LCS-6210D	PC/XT	\$ 45.00
WD WAH AT 16 Bit 2 Hard Drive		\$ 75.00
WD 1003SM1 AT 2:1 Int.		\$ 99.00
WD 1003SM2 Hard/Floppy 2:1		\$125.00
WD 1006SM1 AT 1:1 8K Cashe		\$125.00
WD 1006MS2 Hard/Floppy 1:1		\$135.00
WD 1003SR1 RLL 2:1 Int		\$125.00
WD 1003SR2 RRL Hard/Floppy		\$135.00
WD 1006SR1 RLL 1:1 8K Cashe		\$135.00
WD 1003SR2 RLL Hard/Floppy		\$145.00

SEAGATE DRIVES AND CONTROLLER IN STOCK
Please call for current price.

***** 2400 BAUD MODEM *****
by Computer Peripherals
2400 Baud Internal and Software \$85.00
2400 Baud External and Software \$140.00

CASCADE ELECTRONICS, INC.
ROUTE 1 BOX 8
RANDOLPH, MN 55065
507-645-7997

Please ADD shipping on all Orders
COD Add \$3.00 Credit Card ADD 5%
MN Add 6% Sales Tax Subject to change

Initializing Variables In Turbo Pascal

Handling Initialization During The Link Rather Than At Run Time

This is hacking at its best. Dave obviously understands the limitations of Turbo and he's sent along some fixes. Fun stuff.

The most recent versions of Turbo Pascal (5.0 and 5.5), have a great deal to offer in terms of flexibility, code readability, and development speed. However, because of the way the compiler and its internal linker work, and because of Borland's support for only one memory model, it has some real limitations.

Procedure Tables To Go

One of Turbo Pascal's annoying limitations is the compiler's inability to initialize arrays of procedures (procedure tables) at compile time.

Procedure tables are easy in C. The external linker automatically initializes the address values in an initialized array of procedures (at link time after they have been resolved). See Figure 1 for a simple example. (I'm *not* a C programmer by trade, so the example may not be elegant.)

If you were to translate this directly into Pascal (see Figure 2) and then try to compile it, Turbo complains about the line that initializes the array of procedures.

So what do you do if you want to create a procedure table in Turbo Pascal? Normally you have to waste some run-time code to assign each of the procedures to the array members (see Figure 3).

For an array of four procedures, this doesn't add much to the run-time. But how about 500 procedures? Then it adds up. The .EXE file for the small interpreter you just knocked together using procedure tables would be fatter than it should be. Plus, I think it's kludgy to do all those

Figure 1—PROCTABL.C

```
void proc0(void)
{
    puts("procedure 0");
}

void proc1(void)
{
    puts("procedure 1");
}

void proc2(void)
{
    puts("procedure 2");
}

void proc3(void)
{
    puts("procedure 3");
}

void static (*proc[]) (void) = {proc0,proc1,proc2,proc3};

main()
{
    int n;

    for(n=0;n<4;(proc[n])().n++);
}

♦ ♦ ♦
```

Figure 2—Pascal Version of Figure 1

```
PROCTBL0.PAS(26): Error 99: File and procedure types not allowed here.
Proc : array[0..3] of procedure = (Proc0,Proc1,Proc2,Proc3);
      ^

{$F+}
procedure Proc0;
begin
    writeln('Procedure 0');
end;

procedure Proc1;
begin
    writeln('Procedure 1');
end;

procedure Proc2;
begin
    writeln('Procedure 2');
end;

procedure Proc3;
```

```

begin
  writeln('Procedure 3');
end;
{$F-}

const
  Proc : array[0..3] of procedure = (Proc0, Proc1, Proc2, Proc3);

var
  n : word;

begin
  for n := 0 to 3 do
    Proc[n];
  end.

```

♦ ♦ ♦

Figure 3—PROCTABL1.PAS

```

var
  Proc : array[0..3] of procedure;
  n : word;

{$F+}
procedure Proc0;
begin
  writeln('Procedure 0');
end;

procedure Proc1;
begin
  writeln('Procedure 1');
end;

procedure Proc2;
begin
  writeln('Procedure 2');
end;

procedure Proc3;
begin
  writeln('Procedure 3');
end;
{$F-}

begin
  Proc[0] := Proc0;
  Proc[1] := Proc1;
  Proc[2] := Proc2;
  Proc[3] := Proc3;
  for n := 0 to 3 do
    Proc[n];
  end.

```

♦ ♦ ♦

One of Turbo Pascal's annoying limitations is the compiler's inability to initialize arrays of procedures (procedure tables) at compile time.

assignments, especially when the linker knows what the addresses are before it creates the .EXE file.

Is there a way to make the compiler handle this at compile time? Using only Turbo Pascal, no. The compiler handles the initialization of typed constants before linking. At that time it isn't sure where the procedures will be located.

However, by using a dash of assembly language, you can force the compiler to handle most of the initialization during the link, leaving only the initialization of a single pointer variable for the run-time code. Figures 4 and 5 show one way you can do this.

The fake procedure Procs is never called (and never should be, unless you feel like reaching for the red button). It's simply used to export a pointer from an external assembly language program. Since you cannot declare any data PUBLIC in an external procedure and still have Turbo Pascal see it, you have to cheat.

Fortunately, cheating will make your .EXE file smaller, especially when you

have lots of procedures in your table. The trade off is that you have to maintain a PROCS.ASM file in addition to the Pascal source code.

Typed Constants Without Limit, Anyone?

We all know how valuable that 64K of data segment for variables in Turbo Pascal is, right? Every typed constant (initialized variable) gets its bite (or two). Sometimes 64K seems downright tiny!

Of course, you can allocate all your variables on the heap. But then you have to initialize them with assignments to constants embedded in your code or load in a separate data file. Both approaches add considerably to the size of your .EXE file.

You don't have this problem in any implementation of C that supports multiple data segments. So wouldn't it be nice if Turbo Pascal could manage it? If only you had some way to assign initialized variables to their own segment. Perhaps you wouldn't have to switch to C!

Using a little assembly language, you can do this in Turbo Pascal. Using only 4 bytes of that valuable Turbo Pascal variable space you get access to as much as 64K of initialized data. This trick turns code space into initialized data space. See Figures 6 and 7 for a way to do this.

Note that to get 64K of initialized data space using this technique, you would have to make the external procedure containing the data the sole member of a separately compiled unit. And do it so that virtually none of the code space would be taken up by executable code.

When unitizing, you should also include the pointer variable in the unit and use the run-time initialization capability (placing an assignment statement between a BEGIN and END in the implementation part) to set up the pointer variable. You only use the unit to have all the initialized data available. Using the unit approach is also a good way to hide the fake procedure `_DataSeg1`.

See Figures 8 and 9 for a sample using the unitized approach.

Putting the initialized data into the .ASM file also means you can use the power of TASM or MASM operators and macros to generate complex patterns of initialized data that Turbo Pascal can't match. Just try initializing a typed constant with the equivalent of 1000 `dup('TEXT')` in Turbo!

Figure 4—PROCS.ASM

```
public Procs

; import the addresses for the Pascal procedures
extrn Proc0:far,Proc1:far,Proc2:far,Proc3:far

code segment
assume cs:code

Procs proc far

; use the addresses to build a procedure table
dd Proc0
dd Proc1
dd Proc2
dd Proc3

Procs endp

code ends
end
```

◆◆◆

Figure 5—PROCTBL2.PAS

```
($F+)

($L PROCS.OBJ)
procedure Procs; external;

type
  ProcArray = array[0..3] of procedure;

var
  Proc : ^ProcArray;
  n : word;

procedure Proc0;
begin
  writeln('Procedure 0');
end;

procedure Proc1;
begin
  writeln('Procedure 1');
end;

procedure Proc2;
begin
  writeln('Procedure 2');
end;

procedure Proc3;
begin
  writeln('Procedure 3');
end;

begin
  Proc := @Procs; { point Proc to the procedure table in PROCS.OBJ }
  for n := 0 to 3 do
    Proc^[n];
end.
```

◆◆◆

Figure 6—DATASEG1.ASM

```
public _DataSeg1

code segment
assume cs:code

_DataSeg1 proc far

DefaultInputFile db 11,'DATA_IN.DTA', 68 dup(0) ; string[79]
```



```

DefaultOutputFile db 12,'DATA_OUT.DTA', 67 dup(0) ; string[79]
InitFileOffset   dd 1623447 ; longint
BigArray         db 1000 dup ('TEXT ' ) ; array[1..5000] of char;

_DataSeg1 endp

code ends
end

```

♦ ♦ ♦

Figure 7—INITDATA.PAS

```

{$F+}
{$L DATASEG1.OBJ}
procedure _DataSeg1; external;
{$F-}

type
  DataSeg1Type = record
    DefaultInputFile : string[79]; { 'DATA_IN.DTA' }
    DefaultOutputFile : string[79]; { 'DATA_OUT.DTA' }
    InitFileOffset : longint; { 1623447 }
    BigArray : array[1..5000] of char; { 1000 dup('TEXT ' ) }
  end;

var
  DataSeg1 : ^DataSeg1Type;
  n : word;

begin
  DataSeg1 := @_DataSeg1; {point to initialized data in DATASEG1.OBJ}
  with DataSeg1^ do
  begin
    writeln(DefaultInputFile);
    writeln(DefaultOutputFile);
    writeln(InitFileOffset);
    for n := 1 to 30 do
      write(BigArray[n]);
    writeln;
  end;
end.

```

♦ ♦ ♦

Figure 8—DATA1.PAS

```

unit Data1;

(***** interface *****)

type
  DataSeg1Type = record
    DefaultInputFile : string[79]; { 'DATA_IN.DTA' }
    DefaultOutputFile : string[79]; { 'DATA_OUT.DTA' }
    InitFileOffset : longint; { 1623447 }
    BigArray : array[1..5000] of char; { 1000 dup('TEXT ' ) }
  end;

var
  DataSeg1 : ^DataSeg1Type;

(***** implementation *****)

{$F+}
{$L DATASEG1.OBJ}
procedure _DataSeg1; external;
{$F-}

begin
  DataSeg1 := @_DataSeg1; {point to initialized data in DATASEG1.OBJ}
end.

```

♦ ♦ ♦

Confusing Code?

```

#include <stdio.h>
text_count(){int c,nlines,mwords,nchars,inword;
inword=NO;nlines=mwords=nchars=0;while((c
=getchar())!=EOF){++nchars;if (c=='\n')
++nlines;if (c==' ');{c=='\n')}inword=NO;else
if (inword=NO){inword=YES;+mwords;}printf(
"%d %d %d\n",nlines,mwords,nchars);}

```

C It Your Way!

```

#include <stdio.h>
text_count ()
{
  int c, nlines, mwords, nchars, inword;

  inword = NO;
  nlines = mwords = nchars = 0;
  while ((c = getchar()) != EOF) {
    ++nchars;
    if (c == '\n')
      ++nlines;
    if ((c == ' ') || (c == '\n'))
      inword = NO;
    else if (inword == NO) {
      inword = YES;
      ++mwords;
    }
  }
  printf ("%d %d %d\n", nlines, mwords, nchars);
}

```

NOW with C-Clearly™, format C source code exactly the way you want it. C-Clearly's context sensitive analysis will format any C program in your own personal or corporate style.

EASY to use, C-Clearly's style templates are a snap to modify, since they resemble C source code you edit into your preferred format. Templates are included for several common styles as well as standard K & R.

LISTINGS can also be created with function names and comments highlighted for improved readability. Listing options include line numbers, headers and/or footers and flow lines.

IDEAL for making obtuse code clear. Allows all of your source code to be presented in a consistent format of your choosing. Also great for code walkthroughs and final documentation listings.

WORKS with all IBM PC, XT, AT, PS/2 and compatibles, with 512K RAM. Automatically processes all include files and pre-processor statements. ANSI-C compatible. Not copy protected.

C-Clearly \$129.⁹⁵

Shipping & Handling USA \$5; Canada/Mexico \$10; Other Countries \$15; CA Residents add sales tax. Visa/MasterCard/COD accepted.

**For orders or information call:
1-800-662-8266**

V COMMUNICATIONS, INC.
4320 Stevens Creek Blvd., Ste 275, Dept. MC6
San Jose, CA 95129 (408) 296-4224

Reader Service Number 62

MICRO CORNUCOPIA, #53, May, 1990 43

Initializing Large Amounts Of Data

For large amounts of initialized data, a program that reads the data structure definition and then creates the appropriate db, dw, dd directives, etc., would make creating the .ASM file easier. You might write an include file that has the initialized data appear after each of the members in the record structure definition. Then feed this include file to your program as a data file and have it create the .ASM file.

See Figure 10 for an example.

Your program could parse each line to determine the equivalent assembly language data type and number of elements and then initialize it with the data in the comments. As the famous cop-out goes, "Such a program is left as an exercise for the student."

Editor's note: Yes, yes. I enjoy writing in Turbo Pascal but I, too, am running up against limitations in the language, especially with the 64K limit on the data segment.

◆ ◆ ◆

Figure 9—USEDATA.PAS

```
uses
  Data1;

var
  n : word;

begin
  with DataSeg1^ do
  begin
    writeln(DefaultInputFile);
    writeln(DefaultOutputFile);
    writeln(InitFileOffset);
    for n := 1 to 30 do
      write(BigArray[n]);
    writeln;
  end;
end.
```

◆ ◆ ◆

Figure 10—Example Include File Layout

```
DataSeg1Type = record
  DefaultInputFile : string[79]; (= "DATA_IN.DTA" )
  DefaultOutputFile : string[79]; (= "DATA_OUT.DTA" )
  InitFileOffset : longint; (= 1623447 )
end;
```

◆ ◆ ◆

MultiPurpose Lab Interface for Your IBM®

Turn your IBM-compatible computer into a powerful laboratory instrument with our MultiPurpose Lab Interface (MPLI) hardware and software. MPLI allows you to use a wide variety of sensors to make measurements and to analyze and graph the results. Data can be collected at a rate of thousands of samples per second or slowly over periods as long as several months. Sensors available include assembled, ready-to-use units and parts kits. The MPLI hardware-software combination consists of:

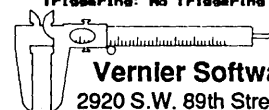
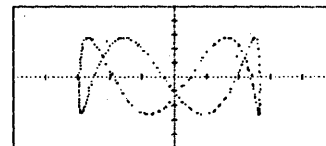
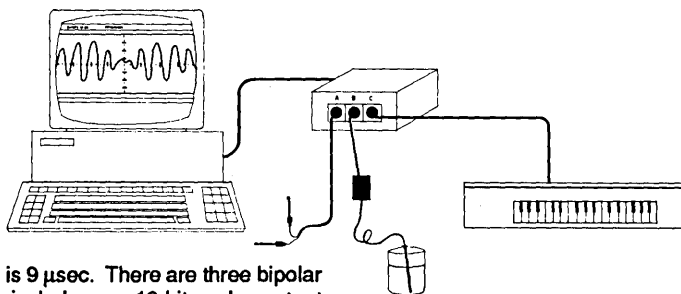
- **A 12-bit A/D Interface Board:** A 12-bit, 8-input analog-to-digital converter with built-in sample and hold. Conversion time of the A-to-D is 9 μsec. There are three bipolar and three unipolar software-selectable voltage ranges. The board also includes one 12-bit analog output and access to 16 digital I/O lines. Includes manual and sample programs disk. (Order Code AIB-PC, \$240)
- **MultiPurpose Lab Interface Box:** Includes three 8-pin DIN sockets which allow quick connection to 3 of the analog inputs, voltage out and power leads. A prototyping area for building your own circuits is included on the circuit board inside the box. A variety of sensors and probes are available which plug into the MPLI box: pH, thermocouple, temperature (AD590), force, microphone, etc. (Order Code BOX, \$45)
- **MultiPurpose Lab Interface Software:** The MultiPurpose Lab Interface Program allows three input channels to be calibrated to display any type of input signal; for example, channel A can read temperature, channel B can read pH, and channel C can read pressure. Data from each of the channels can be collected, graphed, and saved on disk. The output voltage can be controlled by the program. The Oscilloscope mode lets the computer act as a triple-trace, storage oscilloscope with a sampling rate of at least 40,000 samples/sec (on an 4.77 MHz IBM PC), much faster on other computers. The program even does X vs. Y plotting, so you can display Lissajous figures on the monitor. (Order Code MPP-IBM, \$49.95)

The IBM MPLI Package price of \$310 includes:

- 12-bit Interface Board with manual and sample programs
- MPLI Interface Box, cable and test leads
- MPLI Program with extensive manual

A similar package is available for the Apple® II computer.

Call, write or fax for more information.



Vernier Software
2920 S.W. 89th Street #C
Portland, Oregon 97225
(503) 297-5317 FAX (503) 297-1760

ERAC CO.

P.O. Box 1108, 14179 Halper Road
Poway, California 92064 • (619) 679-8360

Baby 386-20/24 Motherboard

One 32 bit-slot, Five 16-bit slot, Two 8-bit slot. Maximum 8 meg on board (1 meg SIMMS). 0K on board. 80287 or 80387 or Weitec Co-Processor slot. Shadow RAM.

-20 \$630 -24 \$650

1 meg SIMMS-80ns \$115

16 BIT VGA Card

RESOLUTION	COLOR
640x480	256 of 256*
800x600	16 of 256*
1024x768	16 of 256*

*with 512K RAM

Runs all standard VGA modes, 256K RAM on board. Drivers for AutoCad, Lotus, Framework, GEM, VP, WP, WS & MS Windows. 1 year warranty.

For 512K \$30 extra \$185

MOTHERBOARDS 386DX/SX/286

DX5000 386/25MHz-32K Cache-16 Meg-Simms-AMI Bios-Full	\$998
DX4000 386/20/24MHz Shadow-8 Meg-Simms-AMI-Baby	630/650
P9400 386SX/16MHz 8 Meg-Simms-Phoenix-Bios-Baby	375
P9200 386 SX/16MHz 2 Meg-Dip-AMI-Bios-Baby	365
SUPABOARD 286/12MHz 4 Meg-Dip-Dallas Clk-AMI-Baby	197

I/O, KB, CONTROLLERS, Etc.

XT Enhanced Hard Disk Drive Controller with Cables	48
DC-11M AT Hard/Floppy Contr. 1:1 438Kb/Sec, To 2048 Cyl.	93
AT I/O PLUS 1 Par(Lpt 1-3), 2 Ser(Com 1-4), Game, Cables, Ser 2(opt)	38
VGA-16 16-bit VGA Board 800x600 with Driver Software	118
2400 Baud External Modem, with Software and Manual	99
KB5161 AT/XT 101 Keyboard, Cherry Keyswitches (Click)	47
COLORMOUSE Black-Red-Blue-Beige-Green-Yellow, Software & Manual	39
VOICE MASTER KEY, Add voice commands to software XT/AT	147

MONITORS

VGA 1489 1024x768, .28 Dot, 14", Swivel, Hor. 31/35KHz, Vert. 50, 60, 70, 87 Hz	430
--	-----

CASES

MINI-TOWER, 230 W, Reset, Turbo, Keylock, Speaker	143
TOWER, 230W, Reset, Turbo (2 Dgt), Keylock, Speaker	229

KAYPRO Equipment Bargains

9" Green Monitor-83, 84, K16	\$60
Host Interface Board	15
Keyboard	50
Replacement Power Supply	70
Drivetek 2.6M FDD (Robie or K4X)	75

We Repair CPM Kaypros

CPM COMPUTERS

K4-84	425
K10	495
K4X	425

IC's

81-189 Video Pal	\$15
81-194 RAM Pal	15
81-Series Char. Gen. ROMs	10
81-Series Monitor ROMs	10

We carry all IC's for Kaypro repair.

Test Equipment

OSCILLOSCOPES

TEK 7403N/7A18N/7B50A 60 MHz ...	\$650
Leader LB0520 30 meg Dual Trace ...	300
TEK 475 Dual Trace 200 MHz	1250
Scope Probe x1, x10 100 MHz	25

ANALYZERS

TEK 491 10MHz-40GHz	\$3500
HP851B/8551B 10HMz-40GHz	1500
Biomation 805 Waveform Rcrdr	195
Biomation 8100 2-Channel Waveform Recorder	295
HP1600A Logic Analyzer 16ch	295
HP1600A/1607A Logic Analyzer 32ch ..	495
Gould K40 32ch Logic Analyzer	750

MISCELLANEOUS

Optronics 550 MHz Freq Cntr	\$95
Heatgun 120Vac 7 A	35

TERMINALS

Televideo 925	\$99.00
---------------------	---------

NiCds

AA Cells .6ah	\$1.00
12V Pack AA Cells .7ah	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
7.2V RC-Pack 1.2ah	18.00

GEL CELLS

6V 5ah	5.00
6V 8ah	6.00
12V 15ah	15.00
12V 2.5ah	8.50
D Cell 2.5ah	2.00

ROBOTICS

5V DC Gear Motor w/Tach 1"x2"	\$7.50
Z80 Controller with 8-bit A/D	15.00
12V Gear Motor 30 RPM	7.50
Cable: DB9M-DB9F 1' length	2.00
High Voltage Power Supply Input: 15-30V DC Output: 100V 400V 16KV	6.50

SWITCHERS

AT 200W Pulls, tested	\$35.00
5V/75, 12V/6, -12V/3, -5V/5	85.00
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/1 1/4A, 12V/6A, -12V/6A ..	29.00
5V/30A	39.00
5V/100A	100.00
5V/120A	110.00
HP DC/DC 12Vin, 5V/8A, 12V/5A, -5V/3A ..	45.00

VERSATEC 8222F 22"

Electrostatic Printer Plotter
200 dots per inch. Up to D size.
1" per second

AT 80286-6 CPU BOARD

with reset and mono/color switch. Connector for KB, Battery & SPKR. Phoenix Bios (tested with Award 3.03), 6MHz, can be upgraded to 8 or 10MHz. Use with backplane, add memory board, I/O board, etc.

ONLY

HOURS: Mon-Fri 9-6, Sat 10-4
For more information, please call
(619) 679-8360

MINIMUM ORDER \$25.00

TERMS: VISA OR MasterCard (Add 3%), Certified Checks, Money Order, NO COD. Personal Checks must clear BEFORE we ship. Include shipping charges. California residents add 7 1/4% Sales Tax.

Fast Neural Networks

Transputers & Code Optimization Make Them Almost Cheap

You can run neural networks on 286s and 386s; but as the networks grow, the time required for training and running gets significant. Fortunately neural nets lend themselves to parallel processing, and fortunately there are transputer boards out there that don't cost megabucks.

Although not the ultimate solution, a coprocessor board (CPB) plugged into a PC (or Macintosh) can significantly increase the speed of a neural network for a relatively low cost.

The CPBs I'll discuss and program in this article are characterized by one or more high speed digital signal processors and a large amount of memory.

A few companies market CPBs (the SAIC Delta II, the HNC Anza Plus, etc.) specifically for neural nets. These CPBs come with software for implementing several different neural networks. One of these systems costs \$10,000 plus.

Other CPBs (Mercury cards, transputers, etc.) are marketed for general numerical and digital signal processing. They're a little less jazzy, but cheaper.

Transputers, in particular, are useful for optimizing neural networks because they're designed for parallel processing. The transputer has the advantage of low start up cost (\$3,000 for a single transputer with 1M memory, board, and compiler). However, its performance isn't as good as some of the more expensive systems. You can overcome this deficit, however, by ganging transputers.

In this article I'll discuss the transputer in general, then show you how to program individual and parallel transputers. In particular, I'll use every trick in the book to optimize the matrix-vector multiply operation (the key neural network process). Finally, I'll discuss a group of programs which create a neural

network—a neural net that can run on as many transputers as you want.

The Transputer

Inmos Ltd. makes several variations on their transputer. The T800 model, the one I'll discuss here, has both an internal 32 bit integer processor and an internal 32 bit floating point unit (see Figure 1).

The 20 MHz version of the T800 can compute at rates up to 1.5 million single precision (32 bit) floating point operations per second (faster than either the 68020/68881 or the 80386/80387).

This transputer has 4K of internal RAM, an external 32 bit address space, and 32 bit wide data. It contains special communication hardware (serial DMA channels called links) that allows many

transputers to work in parallel.

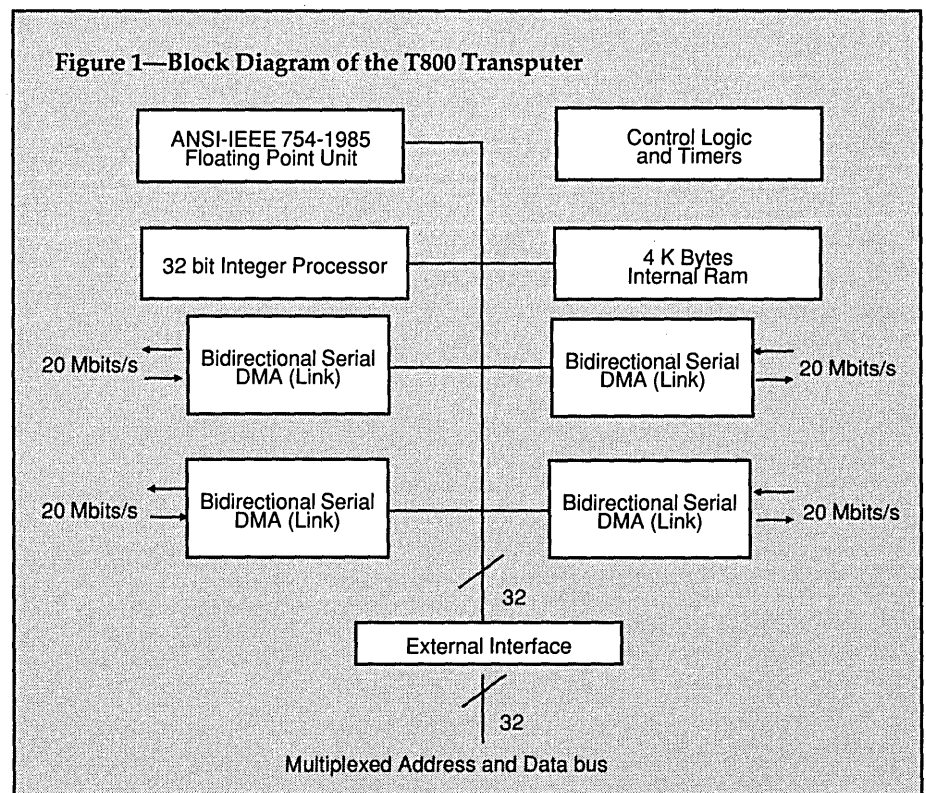
This transputer also has a couple of timers as well as hardware supported multitasking. The transputer was definitely designed to run numerically intensive applications.

Inmos-SGS Thomson and several other vendors have developed products that incorporate transputers into PCs. Each transputer comes on a small daughter board called a TRAM.

So far TRAMs exist for the PC, Mac II, VME, and Eurocard buses. Depending on memory configuration, up to ten TRAMs will fit onto a PC motherboard.

Interprocess Communications

You don't have to decide if two processes will reside in the same proces-



sor when you write the code for the processes; the code is the same.

Processes communicate through one-way channels. In theory, the only way for a process to access data that originates from another process is through a channel (but you can get around this if you want to).

For a channel between processes on adjacent transputers, we implement the channel using a point-to-point hardware link. Each link can either transmit or receive a serial stream of data at 20 Mbits/sec. There are four pairs of links (half of each pair for transmitting, the other for receiving) on the T800. We can therefore attach a processor to a maximum of four other processors.

While individual links are capable of only 1.7 Mbytes/sec throughput (2.4 Mbytes/sec when you use both links simultaneously), the aggregate communications bandwidth of an array of transputers is phenomenal. For example, if you connect all the links of 16 transputers, you can move over 4.8 gigabytes of data per second.

You can connect transputers in various arrangements (e.g., pipes, rings, meshes, trees, hypercubes, etc.). This differs significantly from multiprocessors with shared communications buses that can handle only a few additional processors before bus contention bogs things down.

To create a channel between two processes within the same processor, use a memory location to hold the flag for the microcode that implements the communications instructions.

Communications are always synchronized. When one process wants to communicate with another, it must wait until the other is ready. If there are two processes (whether local or across adjacent transputers) and one is waiting to communicate, it will be suspended.

Transputers, in particular, are useful for optimizing neural networks because they're designed for parallel processing.

When the second process is ready to communicate, data transmission begins. The first process can then continue. It doesn't matter whether the first process was waiting to transmit or receive.

Multitasking

A microcoded scheduler manages multitasking. The transputer supports two levels of process priority: high and low. Processes are either active or inactive. The scheduler maintains two lists of the active processes: one for high priority and one for low.

The scheduler runs all active high priority processes one at a time until they're all waiting for communications or a time delay. Then the process at the top of the low priority list executes for one time slice (approximately one millisecond), or until it becomes inactive as a result of waiting for communication or waiting for a programmed time delay.

Inactive processes don't require attention (i.e., time) from the scheduler. After a communication has completed or the programmed time delay has transpired, the scheduler will add the

process back to the list of those waiting to execute.

In sum, the use of the internal channels and the multitasking scheduler allows a single transputer to support several concurrently running processes. The use of the external channels extends the concurrency across multiple transputers.

Programming Languages

Several languages support the transputer either individually or in parallel configurations: Occam, the first language developed for the transputer (it supports strong data typing, multitasking, multiprocessing, and intertask I/O); FORTRAN; C; Pascal; and Ada.

Each of these compilers extends its language or provides libraries of routines to support the transputer. The transputer programming environment is similar to any PC programming environment.

The C compiler I use, developed by Logical Systems Inc., supports communications and multitasking. It generates assembly language output for the assembler. This means that I can read the code the compiler generates. I can then use the inline assembly language support of the C compiler to optimize the parts that need it. (Occam also supports "human readable" assembly language.)

Optimizing Matrix-vector Multiply

You can improve neural network efficiency by optimizing the matrix-vector multiplies.

Most programmers use indices for their matrix-vector operations to get around the 80x86's 64K segments (see Figure 2). The transputer can access data anywhere within its 32 bit address space and can have very large arrays. This sets the stage for dispensing with indices and

accessing the data via pointer arithmetic.

To use pointer arithmetic, you must organize the data for the weight matrix (in memory) in a very specific manner.

In practical terms this means you arrange all the weights for each next-layer node consecutively. Then instead of calculating an offset into the weight matrix using indices for each weight, you increment a pointer to access each weight sequentially (see Figure 3).

Using this approach with very small arrays on an 8 MHz IBM PC/AT with 80287, I achieved 9800 multiply-accumulates/sec (MAC/sec). On a 20 MHz Compaq 80386 with 80387, I got 65,000 MAC/sec. This is about an 8% improvement over an indexed system.

Next I experimented with the 20 MHz T800 transputer. With indices—184,800 MAC/sec. With pointers—276,900 MAC/sec. A significant improvement, but mediocre considering the advertised performance of 1.5 Mflops. Since a single multiply-accumulate operation executes two floating point operations, the transputer should be getting around 750,000 MAC/sec. What's going on?

Because I perform the matrix-vector multiply with floating point numbers, it's crucial to keep the floating point processor running continuously. The transputer's architecture is designed such that the integer processor can calculate the addresses for the next set of operands while the floating point processor executes a multiply or an add.

To do this you interleave the integer processor instructions with the floating point processor instructions. Since my C compiler generates an assembly language source file, I can see how efficiently it uses the two processors.

Using inline assembly language, I can replace the original C instructions with assembly language instructions. In practice this turns out to be only six or seven assembly language statements for a multiply-accumulate instruction.

You can also speed things up by using two of the transputer's features: how it creates constants, and its high speed internal RAM.³

Loops

Since so few operations occur per iteration during the matrix-vector multiply loop, a lot of time goes into incrementing the index and testing when to exit the loop. To save time you can "open the loop" by enlarging the

Figure 2—C code fragment showing straightforward implementation of Matrix-Vector Multiply.

```

IN[n] values for input nodes
W[m][n] values of weights
OUT[m] values for next layer nodes

for(i = 0; i < m; i++)
{
    OUT[i] = 0; /* Initialize sum */
    for(j = 0; j < n; j++)
    {
        OUT[i] = OUT[i] + (IN[j] * W[i][j]); /* Multiply and Accumulate */
    }
}

♦ ♦ ♦

```

Figure 3—C code fragment showing implementation of Matrix-Vector Multiply using pointers only.

```

IN pointer to the n values for the input nodes
W pointer to the m x n values of weights
OUT pointer to the m values for next layer nodes

for(i = 0; i < m; i++)
{
    *OUT = 0; /* Initialize sum */
    IN = address of first input node
    for(j = 0; j < n; j++)
    {
        *OUT += *IN++ * *W++; /* Multiply and Accumulate */
    }
    OUT++; /* Move to next node */
}

♦ ♦ ♦

```

Figure 4—C Code Fragment showing how the Multiply Accumulate loop can be opened up (in this example by 4). The assembly language code increments I and W on the fly instead of adding 4 at the end of the loop.

```

IN pointer to the n values for the input nodes
W pointer to the m x n values of weights
OUT pointer to the m values for next layer nodes

nr = n - (n%4); /* Determine remainder */

for(i = 0; i < m; i++)
{
    *OUT = 0; /* Initialize Sum */
    IN = address of first input node
    for(j = 0; j < n%4; j++)
    {
        *OUT += (*IN * *W) /* Multiply and Accumulate */
                + *(IN+1) * *(W+1) /* Multiply and Accumulate */
                + *(IN+2) * *(W+2) /* Multiply and Accumulate */
                + *(IN+3) * *(W+3); /* Multiply and Accumulate */
        IN += 4;
        W += 4;
    }
    for(j=0; j < nr; j++) /* Use normal approach */
    { /* for the last few */
        *OUT += *IN++ * *W++; /* Multiply and Accumulates */
    }
}

♦ ♦ ♦

```

code within the loop to perform several multiply/add calculations (see Figure 4).

This approach will improve performance (to varying degrees) no matter which processor you use. In fact, the fastest code (assuming we're using a non-caching processor) would require no looping. Of course this is unrealistic for all but the smallest neural network because of the memory required for the instructions.

How much should you open the loop? You can determine this by taking advantage of how the transputer works with constants. The fastest executing instructions are a byte long and use the first nybble to indicate the instruction and the second nybble for a constant. Thus, constants between 0 and 15 are easily handled in one-byte instructions.

If it needs larger numbers, the transputer builds them a nybble at a time (also using a byte-long instruction). The instruction associated with the last nybble necessary to form a constant will also operate on the constant (e.g., load, store, etc.). For example, if you need a number between 256 and 4095, you need three nybbles (with an additional nybble instruction for each, i.e., a total of 3 bytes). Of course each additional nybble requires an additional clock cycle.

Therefore a good amount to open the loop up by is 16 (for constants between 0-15). This requires only one nybble (and therefore one instruction) to access each operand. To make the code usable for a neural network of any size, add regular loop code after the opened loop code to handle the situation when the number of nodes is not divisible by 16.

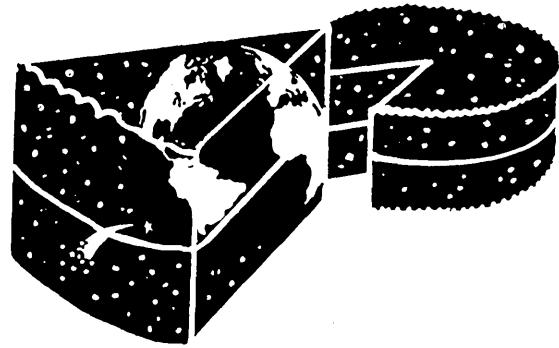
RAM

The second key feature of the transputer is its internal RAM. Most compilers use this RAM for stacks and workspace pointers, but we can tell the compiler to set aside space within internal RAM for a small routine or two. Instructions and data found in this area (4K) can be retrieved in a single clock cycle.

Since the less expensive TRAMs use external memory requiring four clock cycles for access, we can speed up the matrix vector multiply by using the internal RAM. It's too bad we can't get the arrays in there also.

Now what are we up to? Using the hand-optimized assembly language to maximize the floating point operation and opening the loop by 16 to minimize

So you want Tele™ to do it ALL...



OK, piece a cake!

Berry Computers presents The Tele Operating System — A MULTITASKING MS-DOS COMPATIBLE

Computer software exists in layers. Tele contains the layers between your application and the hardware. Tele executes binary programs intended for MS-DOS in a multitasking, windowing environment.

You can use Tele to run several programs at once thereby eliminating boring and unproductive delays. For instance, in a software development environment, you can set a compiler running and simultaneously edit another source file.

Tele uses a preemptive task scheduler. Its file system is fully compatible with MS-DOS. It runs on any processor compatible with the Intel 8086 family.

Most programs intended for MS-DOS will multi-task under Tele. Microsoft could find no way to make MS-DOS multi-tasking and still support all its existing applications.

Tele simply does the best possible. Tele services the documented interface to MS-DOS, fits in less than 100K of memory, and provides the important features of OS/2., including installable file systems.

Tele improves display performance by a factor of 2, whether you use multiple windows or not! On processors slower than 20 MHz the improvement is even greater.

Source code in C and assembly is included for the interface to MS-DOS. If you have to have MS-DOS idiosyncracies, you can easily put them in yourself.

Treat yourself to an operating system where the marketing and financial folks had no say at all in the design. The result is an efficient, structured program that either provides or specifically anticipates all features of modern mainframe and desktop operating systems.

Tele Operating System	\$100
Tele Tool Kit	
Multitasking (MT)*	\$ 50
Windows (WI)*	\$ 40
File System (FS)	\$ 40
Tele Operating System and Tool Kit	\$200
Demonstration Disk	\$ 5

*MT formally called SK, WI formally called CD.

Telephone support is freely available.

Tele is available from:

Crosby Associates
P.O. Box 248
Sutter Creek, California
95642

CALL NOW TO ORDER:
(209) 267-0362
FAX (209) 267-9246*

*Note new Fax #

Visa, Mastercard, American Express & Discover Card accepted.

MS-DOS and OS/2 are trademarks of Microsoft Corporation.

Reader Service Number 147

the loop instruction overhead, I got around 675,000 MAC/sec. When I put the code in internal RAM, the speed improved by 50,000 MAC/sec to over 725,000 MAC/sec. Now we're talking!

Note: You can download the code for the matrix vector multiply and the code for performing the transposed vector matrix multiply (needed during the error correction phase of the neural network simulation) from the Micro C BBS. The transposed code isn't quite as efficient as the first but still does over 700,000 MAC/sec.

Whew! I think we've eked every bit of speed out of the transputer. You can use this code with both online and epoch training. Take your PC code and modify

sors) increase linearly in performance with an increase in processors.

One such approach is a "processor farm."⁴ In a processor farm one processor serves as the "farmer" and the remaining processors serve as "workers." The worker processors execute identical code. The farmer processor sends packets of data to each worker. When a worker finishes with its work (its packet), the farmer retrieves the results and sends a new packet. Farming continues until the work is done.

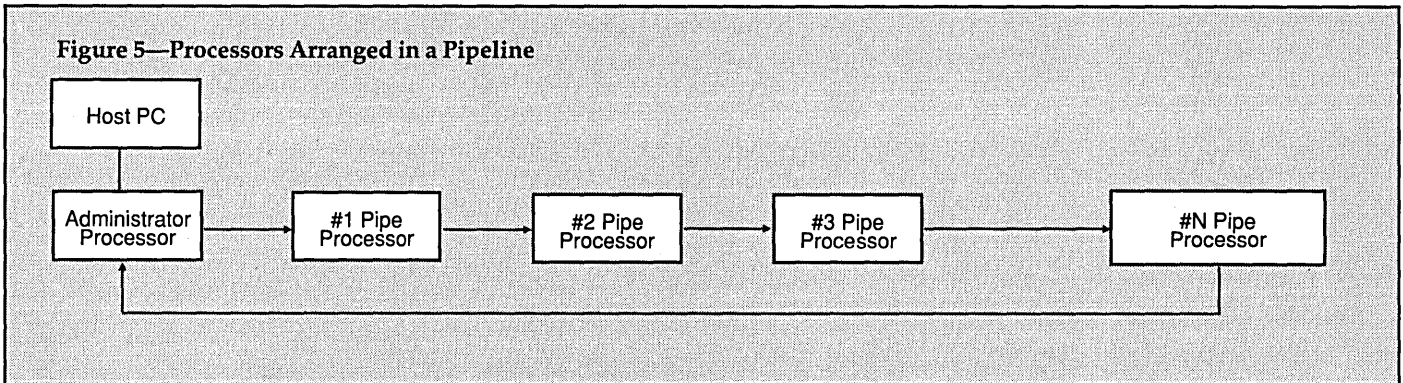
This approach is probably the most appropriate for using multiple transputers to simulate neural networks. Each worker runs its own neural network.

weights and the weight corrections through the pipeline. Each pipe processor has a complete copy of the neural network's nodes as well as the input and output values for each exemplar vector that it's responsible for.

The weights are stored and modified in the administrator processor at one end of the pipeline (typically another transputer). The administrator processor is connected to the host PC and to the first and last processors in the pipeline. It gets the topology and exemplar and test vectors from the PC (typically in files) and then uses this information to configure the pipe processors.

When the pipe processors have their

Figure 5—Processors Arranged in a Pipeline



it to insure that the data for the weight matrix is arranged correctly. Otherwise, code written for the PC using Turbo C (from Borland International) or Microsoft C compiles under the Logical Systems' C compiler with little or no modification.

Using Transputers In Parallel

So far, I've used a single transputer to simulate neural networks. If you want to process more data and/or process data as quickly as possible, then a parallel approach to transputing is the ticket.

The underlying architecture of the transputer, with its multitasking and synchronized communications capabilities, almost eliminates the programming hassles associated with controlling several tasks concurrently.

Processor Farms

To implement a problem on several transputers, you need to distribute the workload as evenly as possible.

Some algorithms fit more naturally into a parallel implementation than others. So, you look for algorithms which (when applied to parallel proces-

Pipelining

While a processor farm makes sense once you know the neural network's weights, it's not good for training. An algorithm called pipelining is a better approach for training.

Pomerleau et. al.⁵ describes the algorithm for use with the Carnegie Mellon University Warp machine (a very expensive systolic array processor). Chong and Fallside⁶ describe it for the transputer.

This algorithm uses a group of processors arranged in a pipeline (see Figure 5). In a pipeline, each processor receives intermediate results and/or data from its upstream neighbor, processes the data, and then sends its results/data to its downstream neighbor.

For best results, try to communicate (send/receive data) and compute (process data) in parallel. Also, each processor's work load should be relatively equal. So no processor waits for another.

For neural network simulations on an array of transputers, you typically send a subset of the exemplar vectors to each pipe processor. Then you pipe the

exemplar vectors and a copy of the neural network nodes, the administrator processor sends a copy of all the weights down the pipeline. As a pipe processor receives the weights, it immediately sends them to the next processor downstream. Then (in parallel) it uses the weights to make a pass through the neural network for each of its exemplar vectors.

The pipe processor compares the calculated output values with the exemplar output values, then calculates an error value which it sends back to the administrator processor. The administrator node receives the error values from all the pipe processors and determines if the total error for all the exemplar vectors meets the desired criteria.

If the error is low enough, the administrator processor transfers the weights back to the PC and runs through any test vectors that have been supplied.

If the error is too high, the administrator sends a message to each pipe processor instructing it to calculate a change of weights based on its exemplar vectors. The administrator processor then sends an empty array down the pipeline.

When a pipe processor receives the array, it adds the change in weights that it's calculated to the array, then sends it downstream to the next pipe processor.

When the administrator processor gets the array back, it will have the sum of all the changes for each weight based on all the exemplar vectors. The array is then multiplied by the learning rate constant and added to the previous changed weight vector (which has been multiplied by the desired momentum constant) to produce a new weight vector.

The administrator creates a new set of weights by adding the present weight vector to the new weight vector. Then the whole process begins again with the administrator node sending the new weights down the pipeline.

Programming Transputers

To implement the pipelined neural network simulation on the transputers requires only three programs, no matter how many processors are involved. The first program (ADMIN) is the administrator processor program.

This program interfaces to the host

PC and to the pipeline. It's responsible for:

- (1) Retrieving the topology of the neural network;
- (2) Transmitting the information down the pipeline so that each pipe processor can configure itself;
- (3) Retrieving all the exemplar vectors from the host;
- (4) Transmitting evenly divided subsets of the exemplar vectors to each pipe processor;
- (5) Transmitting the weights into the pipeline;
- (6) Getting error measurements from each processor in the pipe;
- (7) Determining whether the total system error is below a user specified value;
- (8) If necessary, instructing the pipe processors to calculate the weight changes;
- (9) Sending an empty weight change array downstream;
- (10) Collecting the filled weight change array from the pipe processors;
- (11) Using the host (i.e., user) supplied momentum and learning rate con-

stants along with the weight change array to calculate a new weight vector;

- (12) Looping back to step 5;
- (13) When training is complete, saving the weights;

(14) And, if necessary, running test vectors on the trained neural network and collecting statistics.

The program (PIPE) is run on all the pipe processors. It's responsible for:

- (1) Receiving and retransmitting the topology of the neural network;
- (2) Setting aside enough space for the weights and its own exemplar vectors;
- (3) Reading and storing its own exemplar vectors;
- (4) Receiving and retransmitting any exemplar vectors destined for processors downstream;
- (5) Receiving and retransmitting the weights;
- (6) Using the weights to calculate the output for the neural network for each of its exemplar input vectors;
- (7) Comparing the neural network calculated outputs with the exemplar output vectors and calculating an error value;

Save "Man-Years of Effort" with Turbo 5.5

Don't Start from Scratch with Object-Oriented Pascal

Object Professional is a huge library of over 200 object types and 2000 methods that will multiply your productivity. Window object types let you use overlapping and resizable windows. The windows include ■ scrolling data entry screens ■ pick lists ■ menus ■ file selection ■ printed forms ■ help capability and more.

Build your programs using proven data object types like stacks, linked lists, virtual arrays, and more. System-oriented routines provide swappable TSRs in only 6K of RAM, EMS management, and much more.

Satisfaction guaranteed or your money back within 30 days. Add \$5 per order for shipping in U.S. and Canada. Inquire about other shipping charges. OPro requires Turbo 5.5. BTF requires Turbo 4.0, 5.0, 5.5, or QuickPascal.

Object Professional includes clear, comprehensive documentation, on-line help, full source code, technical support, and hot demo programs. Pay NO royalties. You'll get up to speed fast with OOP!

"The range of objects is fantastic. Object Professional could literally save you man-years of effort."

Jeff Duntemann

Object Professional 1.0, only \$150.

A Multi-User B-Tree Toolkit

Write powerful network compatible databases faster and easier using B-Tree Filer 5.0. You'll have the fastest, safest, most flexible databases - no rigid structure, no TSR hassles, no running out

of files. And they're compatible with Novell, 3Com, MS-NET, and others.

You get ■ Fixed and variable length records ■ Two billion records per database ■ Up to 100 indexes per index file ■ Fail-safe mode with journaling ■ Units for sorting, browsing, reindexing, and network control.

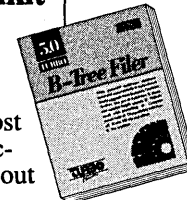
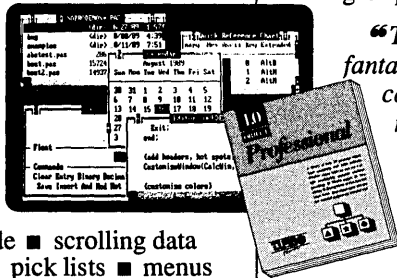
B-Tree Filer includes full source code, documentation, technical support, and you pay NO royalties.

"B-Tree Filer... a well rounded, feature-rich approach to B-Tree databases."

Computer Language, 1/90

B-Tree Filer 5.0, only \$125. (single user)
With network support, \$175.

**Call toll-free to order.
1-800-333-4160**



8AM - 5PM PST Monday through Friday, USA & Canada.
For more information call (408) 438-8608. Fax: (408) 438-8610.
TurboPower Software PO Box 66747 Scotts Valley, CA 95067-0747

(8) Transmitting its error value downstream;

(9) Receiving and retransmitting any error values produced by upstream processors;

(10) Receiving and retransmitting any instructions regarding a backward pass through the network;

(11) If necessary, using the output errors from all the exemplar vectors to calculate the changes in the weights;

(12) Receiving the weight change array from upstream, adding the locally calculated changes, and transmitting the array downstream;

(13) And looping back to 5.

The third program (TAIL) runs on the last transputer in the pipeline in parallel with a copy of the PIPE program. Unlike the other processors that used the hardware DMA channels (links) to transfer data from one processor to the next, the PIPE and TAIL programs on the last processor communicate with each other over a memory channel.

From a programmer's perspective, memory channels operate identically to hardware channels. This means that the last PIPE program is identical to the PIPE programs on the other processors.

The TAIL program primarily transmits information coming out of the pipeline back to the administrator processor and collects garbage. When the neural network topology information reaches the last pipe processor, there's nowhere to send the data.

Since all the PIPE programs are identical, the last PIPE program will try to send the data on anyway. TAIL receives the topology data and discards it. The weights transmitted down the pipeline are likewise garbage collected. The error values and the weight change array, however, are sent back to the administrator.

Each of the processors has a harness program. The harness programs specify whether specific channels are hardware or memory channels and which programs (ADMIN, PIPE, or TAIL) are to be run together. The TAIL program can run on the same processor as the ADMIN program instead of with the last PIPE program. Or, if space allows, a copy of the PIPE program can run on the administrator processor. To maintain load balance, it might be necessary to reduce the number of exemplar vectors that this first PIPE program processes.

Finally we must write a short configuration file (used by the network

loader program) to indicate which program is to run on which transputer. The network loader is responsible for resetting all the transputers and loading each one with its specific program. From the file, the network loader can tell which transputers and which links a program must be transmitted through before it arrives at the transputer on which it's to (ultimately) run.

Unfortunately, programming transputers isn't always as simple as I've made it sound. Fortunately, vendors are improving their operating environments.

For example, some vendors support message handling protocols. With some of the protocols, a communications process runs continuously in parallel with the application program. Any time data must be sent to another transputer, the application program uses a vendor supplied function or a memory channel to pass the data to the communications process. This process adds a header to the data and sends the message out the appropriate link.

Another transputer receiving the message uses the header to determine whether the message is intended for another transputer or for itself. If necessary, the process forwards the message off to another processor. Otherwise the process strips the header off the message and sends the data through a memory channel to the appropriate application process. While this scheme slows down applications, it can make communications simpler.

More important, these communications processes give you a means for debugging remote transputers. Although the application program may have stopped on a remote processor because needed data hasn't communicated to it, the communications process can still send and receive messages. One of those messages can be an instruction to read a particular memory location or register and send the data back over a link.

Discussion

The topology of the neural network, the number of test vectors, and transputer memory resources will determine how you manage the transmission of weights. In most situations the administrator transputer transmits the weights, storing them in the pipeline processors.

During the error correction phase of the training cycle, each processor generates changes for the weights. These

changes are then sent back to the administrator where momentum and learning rate adjustments get made. This approach requires few (if any) parallel processes.

If, however, the network is very large and there are memory limitations at the nodes, you might have to break the weights into packets (say all the inbound weights for a hidden unit) and send them sequentially down the pipeline.

The PIPE program would then calculate all the partial sums for each exemplar vector. With all the weights for one layer received, the activation function would be calculated for each unit.

There are, of course, other approaches to dividing a neural network simulation among transputers. For example, reference 7 describes how each transputer can process a subset of the nodes. Other researchers have used transputers with non-back-propagation neural networks.^{8,9,10}

In Sum

The back-propagation simulation of neural networks maps very nicely into a parallel processing environment. While other parallel processing platforms are available, transputers provide a cheap way to optimize neural network simulations. Plus, we can use these transputers for other applications as well.

References

- 1 Pelczarski, M., "System Review: Microsoft Softcard," *BYTE*, vol. 6, no. 11, November, 1981, pp 152-162.
- 2 Inmos Limited, *The Transputer Reference Manual*, Prentice Hall, London, 1988.
- 3 Atkin, Phil, "Performance Maximisation," Technical Note 17, Inmos-SGS Thomson, Bristol, UK, March 1987.
- 4 "Some Issues in Scientific Language Application Porting and Farming Using Transputers," *The Transputer Development and iq Systems Databook*, Inmos Ltd., 1989.
- 5 Pomerleau, D. A., Gusciora, G. L., Touretzky, D. S., Kung, H. T., "Neural Network Simulation at Warp Speed: How We Got 17 Million Connections per Second," Proceedings of IEEE International Conference on Neural Networks, San Diego, USA, July 1988, pp II-143-150.
- 6 Chang, M. W. H., and Fallside, F., "Implementation of Neural Networks for Speech Recognition on a Transputer Array," Technical Report CUED/F-IN-

FENG/TR8, Cambridge University, Department of Engineering, Cambridge, UK, March, 1988.

7 Beynon, T., "A Parallel Implementation of the Back-Propagation Algorithm on a Network of Transputers," Research Initiative in Pattern Recognition, Royal Signals and Radar Establishment, Malvern, UK, Poster Paper, 1988 IEEE International Conference on Neural Networks.

8 Di Zitti, E., Caviglia, D. D., Bisio, G. M., and Parodi, G., "Neural Networks on a Transputer Array," Proceedings of the 1989 International Conference on Acoustics, Speech, and Signal Processing, pp 2513-2516.

9 Abbruzzese, F., "A Transputer Implementation of a McCulloch & Pitts Network," Parallel Processing and Applications, E. Chiricozzi and A. D'Amico eds., North-Holland, 1988, pp 135-140.

10 Board, J. A., Jr., and Lu, J. S. J., "Performance of Parallel Neural Network Simulations," Proceedings of the Second Conference of the North American Transputer Users Group, J. A. Board, Jr., ed., Durham, North Carolina, USA, North American Transputer Users Group, 1989, pp 185-200.

11 Lee, D. G., Jr., "Preliminary Results of Applying Neural Networks to Ship Image Recognition," Proceedings of the 1989 International Joint Conference on Neural Networks, June 18-22, 1989, Washington DC, IEEE, San Diego, II-576.

12 Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing Volume 1: Foundations*, Rumelhart and McClelland ed., Cambridge, Massachusetts, USA: MIT Press, 1986, Ch. 8, pp 318-362.

13 Leung, H. C., and Zue, V. W., "Applications of Error Back-Propagation to Phonetic Classification," Proceedings of the Neural Information Processing Systems - Natural and Synthetic Conference, November 28 - December 1, 1988, D. S. Touretzky, ed., Morgan-Kaufman, 1989.

Vendors

Companies advertising Transputer TRAMs and/or language compilers for use with the Transputer:

Inmos-SGS Thomson
1000 Aztec West
Almondsbury Bristol BS12 4SQ
UK

Levco
6181 Cornerstone Ct. East, Ste. 101
San Diego, CA 92121

Multis Corporation
99 Willie Street
Lowell, MA 01854

MicroWay, Inc.
P.O. Box 79
Kingston, MA 02364

Computer Systems Architects
950 North University Avenue
Provo, UT 84604

Logical Systems
P.O. Box 1702
Corvallis, OR 97339

3L Limited
Peel House
Ladywell Livingston EH54 6AG
Scotland

Sension
Denton Drive
Northwich Cheshire CW9 7LU
UK

Transtech Devices Ltd.
Unit 17, Wye Industrial Estate
London Road
High Wycombe
Buckinghamshire HP11 1LH
UK

Companies advertising neural network software using coprocessor cards and IBM PC compatible computers:

SAIC
10260 Campus Point Dr.
San Diego, CA 92121

HNC
5501 Oberlin Drive
San Diego, CA 92121

Company advertising neural network software for Transputers and Apple Macintosh II computers:

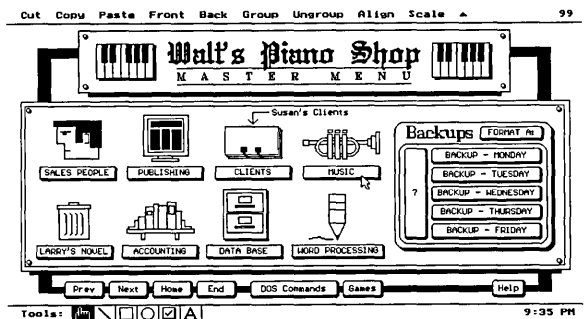
Neurix
One Kendall Square, Suite 2200
Cambridge, MA 02139

◆ ◆ ◆

You're in charge

Don't be limited by someone else's idea of the perfect menu. SuperMenu's unique graphical interface adds a level of creativity, individuality, and ease-of-use not found anywhere else. Why shoehorn your software into a stuffy, preconceived mold when SuperMenu gives you complete control over the structure, format, and appearance of your menu? Most menu systems give you a modest choice of screen colors and then claim to be customizable. SuperMenu gives you color plus a palette of drawing tools, multiple fonts, fill patterns, line styles, pop-up notes, and more. You can even edit the background pattern. In fact, SuperMenu has many features you'd only expect to see in an expensive drawing program, much less a menu generator.

- Graphical mouse-driven interface
- Instant access to your software and DOS options
- Works w/all popular programs
- Easy-to-use drawing tools, fonts, fill patterns, line styles, and colors
- Multiple menus, each with up to 99 pages
- Hypertext-like links between pages
- Pop-up notes
- Passwords prevent unauthorized modifications to your menus
- Date and time display
- NOT memory resident
- Enter DOS commands or invoke a DOS Shell
- Supports CGA, EGA, VGA, and Hercules graphics



SuperMenu's easy-to-use mouse driven interface makes running your favorite software a snap. Other computer tasks, such as daily backups, can be easily automated and accessed with a simple mouse click.

Because SuperMenu has a wide range of powerful tools, menus can be as complex or as simple as you desire. Like we said, you're in charge.

SuperMenu

49⁹⁵ Add \$5.00 S & H
Please specify 5 1/4" or 3 1/2" when ordering

OSCS 
1293 NW Wall Street, Suite 71
Bend, Oregon 97701
(503) 389-5489

Requires: IBM PC, XT, 286, 386, PS/2 or compatible. DOS 2.0 or greater. 256K RAM. Microsoft, Logitech or compatible mouse. CGA, EGA, VGA, Hercules or compatible display.

Reader Service Number 216

Save The Floppies!

R_x For Slipped Disks

This is probably the easiest to use disk recovery program I've ever seen. You just insert the flaky disk and fire up the program. When it finishes, every sector on the disk will be readable, probably.

Data error reading drive B Abort, Retry, Ignore? Grrrrr...

Back in the old days, when CP/M had trouble reading a floppy, I could break out the dependable and user-hostile program DU (Disk Utility) for a little rescue work. DU didn't care about bad sectors; it would read with reckless abandon. Often, the "bad sector" had just a single bad byte. An easy repair job would restore the file's readability.

I thought it would be useful to have this ability in an MS-DOS system. Forget for the moment that MS-DOS already lets you repair disks with its RECOVER program. At the beginning of this project, I hadn't made the acquaintance of RECOVER.

A Little Background

Each sector on a disk consists of sync bytes, ID address marks, ID fields (cylinder, head, sector, bytes per sector), a CRC value for the ID fields, a gap, more sync bytes, data address marks, data (whew, at last), a CRC for the data, and yet another gap. That's at least 105 bytes of overhead for each 512 bytes of data. Add another 146 (or so) bytes of header info for each track and you'll get an idea of what the Floppy Disk Controller (FDC) has to wade through.

At the lowest level, the FDC, acting under orders from the CPU, selects a drive, fires up the drive motor, seeks to the track, performs the requested function, and reports the results to the CPU.

During a read, the FDC looks at the

Assuming that a CRC error means a data error, and barring physical damage to the disk, simply rewriting the questionable data to the same sector should (and has, on numerous occasions) restore the file.

track until it either finds ID fields matching those requested, or sees the disk's index hole pass twice without finding the matching fields (sector not found). If it finds the sector, the FDC sends the 512 bytes of data to the host—in MS-DOS systems, to the Disk Transfer Address (DTA).

As the FDC scans sector ID fields, it generates a CRC value for the IDs and compares this with the CRC recorded on the disk. If the two don't match, the controller returns a CRC error.

A second type of CRC error comes when the CRC calculated during the data read doesn't match the data CRC recorded on disk. This is the error we're after. It means that one or more bytes of

data have changed (or that the CRC has been corrupted). Why'd they change? Beats me: a stray magnetic field, a glob of peanut butter, cosmic rays....

It would be nice to isolate the data CRC error from the ID field CRC error. But after a wade through the XT BIOS listing, I can't see that it differentiates between the two. So we can either make the rash assumption that any CRC error will come from the data, or we can dive into the FDC to get a closer look at the error.

Leave your trunks at home (leave your rash, too). It doesn't make sense to try to fix the sector ID since it gets written only during formatting; reformatting a track would wipe out the information we want to recover. We probably couldn't read the sector anyway, since the corrupted ID fields wouldn't match those requested by the CPU.

Assuming that a CRC error means a data error, and barring physical damage to the disk, simply rewriting the questionable data to the same sector should (and has, on numerous occasions) restore the file.

Levels Of Disk Access

Most programs use DOS interrupt 21h to read a floppy. DOS deals with files, so application programmers like it. No fuss, no muss. But we'd like to get into some real trouble here. We can get at the whole disk by dropping one level closer to the hardware and making use of the BIOS.

Nestled in ROM, the BIOS doesn't know or care about directories, files, or any of the other high level nonsense that DOS requires. With the BIOS you can access any sector on any disk, whether it's allocated to a file or not (including the directory and FATs). And, like DOS INT 21h, you can choose to ignore error conditions.

Figure 1—FLOPFIX.C

```

/* FLOPFIX.C - Reads all sectors of a floppy. Any sectors with bad
   CRCs are rewritten. */

#include <stdio.h>
#include <dos.h>

int crc_failures;
int sectors_recovered;
int non_crc_failures;

void reset_FDC ()                /* reset the floppy disk controller */
{
    union REGS in, out;

    in.h.ah = 0;                  /* service 0 = reset */
    int86 (0x13, &in, &out);     /* INT 13h */
} /* reset_FDC () */

unsigned char read_sector (unsigned char track, unsigned char side,
                          unsigned char sector)
{
    union REGS in, out;

    in.h.ah = 0x2f;              /* get DTA */
    intdos (&in, &out); /* do the INT, ES:BX holds address on return */

    in.h.ah = 2;                /* read diskette */
    in.h.al = 1;                /* # sectors to read */
    in.h.ch = track;
    in.h.cl = sector;
    in.h.dh = side;
    in.h.dl = 1;
    int86 (0x13, &in, &out);     /* drive B: */
    return (out.h.ah);          /* do the read */
} /* read_sector () */

unsigned char write_sector (unsigned char track, unsigned char side,
                          unsigned char sector)
{
    union REGS in, out;

    in.h.ah = 0x2f;              /* get DTA */
    intdos (&in, &out); /* do the INT, ES:BX holds address on return */

    in.h.ah = 3;                /* write diskette */
    in.h.al = 1;                /* # sectors to write */
    in.h.ch = track;
    in.h.cl = sector;
    in.h.dh = side;
    in.h.dl = 1;
    int86 (0x13, &in, &out);     /* drive B: */
    return (out.h.ah);          /* do the write */
} /* write_sector () */

void fix_disk ()

```

continued on page 56

The key here is that, even though an error condition exists, the data *has* been read. It's alive and well at the DTA, and all you have to do is rewrite it to disk.

Code Notes

You won't find anything very startling in FLOPFIX.C (see Figure 1). `fix_disk()` makes three attempts to read each sector. It takes a while for the drive motor to spin up, so a well behaved program should do these retries to ensure that any errors are real. (Reads from a half-fast drive don't have much chance of success.) A call to `reset_FDC()` follows any read failure—again, a required programming practice.

Once we're sure an error has occurred, `fix_disk()` checks the status returned by `read_sector()`. If the status indicates a CRC error, `write_sector()` attempts to rewrite the sector. Global variables for the number of bad sectors and recovered sectors get updated and we move onto the next sector.

You'll notice that each sector access (in `write_sector()` and `read_sector()`) starts with a call to determine the DTA. This could have been done only once during program initialization, since the DTA doesn't change during execution (unless the program changes it explicitly). But the call to DOS function 2Fh provides a convenient way to load ES:BX with the DTA.

But Why Bother?

Sure, sure. Once again, I've reinvented the wheel. But in some situations it's a better wheel, a bit more round. RECOVER has not been as successful as FLOPFIX in restoring damaged sectors to readability (at least, in my humble opinion). On one occasion a RECOVERed file was not TYPEable. Besides, diving into the guts of the machine keeps me out of trouble. It's al-

ways instructional to try to understand the inner workings of a system.

If I had more gumption, I'd disassemble RECOVER to see what makes it tick. I could rip off the good stuff and add it to FLOPFIX. But wisps of cloud over the nearby Cascades promise a fine day of early spring back country skiing. Some other time for the disassembly.

Improvements

As usual, I've kept things very simple. Any number of obvious additions would make FLOPFIX better.

Adapt FLOPFIX to hard drives. No real problem, but hard drives are a little scary.

Use DOS to make the program file oriented rather than sector oriented. This would let you recover a specific file.

Write the recovered sector to a different disk (or, at least, to a different sector). This would allow for recovery of physically damaged sectors.

I see nothing wrong with keeping it simple and easy to use. Dave's always so complimentary about my code. I believe his words were something to the effect

Continued from page 55

```

unsigned char track, side, sector, status;
char tries;

reset_FDC ();
for (track=0; track<40; track++)          /* do 40 tracks */
  for (side=0; side<2; side++)            /* 2 sides */
    for (sector=1; sector<10; sector++)    /* 9 sectors */
      {
        tries = 1;                          /* first attempt */
        status = read_sector (track, side, sector);
        if (status!=0)                       /* error on first read */
          {
            do                                /* try 2 more times */
              {
                reset_FDC ();                /* always reset after an error */
                status = read_sector (track, side, sector);
                tries++;
              } while ((tries<4) && (status!=0));
            if (status!=0)                     /* bonafide error */
              {
                reset_FDC ();                /* reset after failed read */
                if (status==0x10)             /* is it a CRC error? */
                  {
                    crc_failures++;
                    if (!write_sector (track, side, sector)) /* write OK */
                      {
                        sectors_recovered++;
                        printf ("CRC error tr=%d side=%d sector=%d OK\n",
                                track, side, sector);
                      }
                    else                       /* couldn't rewrite sector */
                      {
                        reset_FDC ();        /* reset after failed write */
                        printf ("CRC error tr=%d side=%d sector=%d BAD\n",
                                track, side, sector);
                      }
                  }
                else                           /* not a CRC error */
                  {
                    non_crc_failures++;
                    printf ("NonCRC error=%xh tr=%d side=%d sector=%d\n",
                            status, track, side, sector);
                  }
              }
            }
          }
      } /* fix_disk () */

main ()
{
  crc_failures = non_crc_failures = sectors_recovered = 0;
  puts ("\nFloppy repair program");
  puts ("Put disk in drive B: and hit a key...\n");
  getch ();                                /* wait for key press */
  fix_disk ();
  printf ("\nrc failures detected      = %d\n", crc_failures);
  printf ("sectors recovered           = %d\n", sectors_recovered);
  printf ("non-crc failures detected = %d\n", non_crc_failures);
}

```

◆◆◆

that, "Any idiot could use this program." True. I leave it to the intrepid reader to add features and move FLOPFIX beyond the "idiot phase."

Fini

So much for my contribution to the final issue of *Micro C*. It has been a very great pleasure to meet you all in these pages over the last four years. Perhaps we'll bump into each other again in

another magazine, or pawing through some bin hidden in the bowels of an out-of-the-way surplus parts house. 'Til then....

semper ubi sub ubi

◆◆◆

'C' and 'C++'

DOCUMENTATION TOOLS

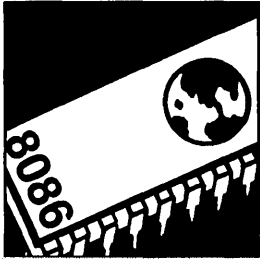
Save TIME and MONEY

Automate ACCURATE documentation

- **C-CALL™ (\$59)** Creates a graphic-tree of the caller/called structure, generates a files-vs-functions table of contents, and a cross-reference of function usage.
- **C-CMT™ (\$59)** Generates and inserts function "Coment-Blocks" showing caller/called functions and showing global/define/local identifiers used.
- **C-LIST™ (\$39)** Lists programs with optional line numbers and page titles. Optional graphic "Action-Diagrams" of the logic/control structure. Optional reformat into standardized formats.
- **C-REF™ (\$49)** Creates cross-reference of global/define/local identifiers. Creates Class-Hierarchy-Diagram of C++ Classes (incl. multiple inheritance).
- **SPECIAL OFFER !! (\$149)** All 4 programs plus FREE integrated **C-DOC™** program.
- **30-DAY** Money-back guarantee !

SOFTWARE BLACKSMITHS INC
6064 St Ives Way,
Mississauga, ONT, Canada. L5N-4M1
(416)-858-4466

Reader Service Number 219



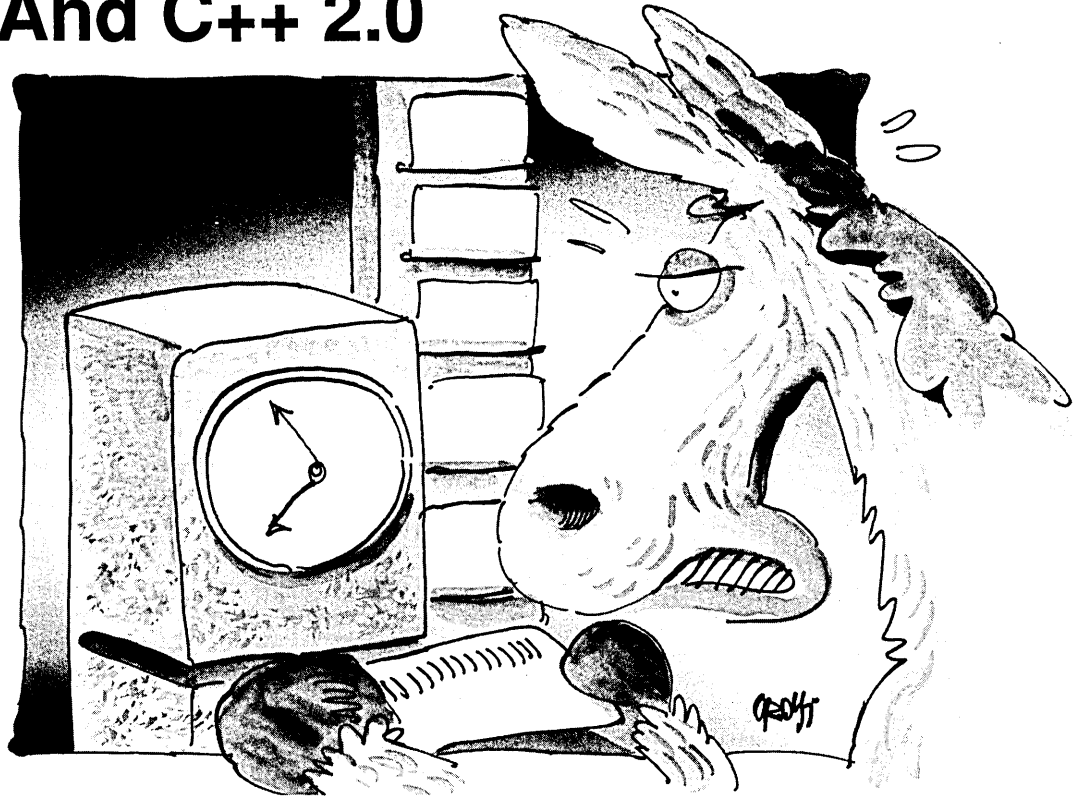
86

WORLD

By Laine Stump

% Redhouse Press
Merkez PK 142
34432 Sirkeci
Istanbul, Turkey

Batch File Prophylactics And C++ 2.0



Laine tries out Zortech's new C++ compiler and debugger at 2 a.m. Here's his red-eye special on this new package.

Just a note here at the beginning for those of you who think you're underpaid: An article I read last week in the Turkish daily newspaper *Sabah* (Morning) reported that, according to a new pricing schedule issued by the Turkish government, the wages of an unskilled worker are now lower than the wages of a donkey. A donkey working as a beast of burden is now paid 480,000 liras (\$202) per month, while a farm or factory worker is paid 225,000 (\$95).

Editor's note: Wonder how much supervisor donkeys get.

This is especially comic in a country where the blank in the common phrase "You son of a &*&#@!" is replaced with "Donkey." (The way things are going, that may soon become a compliment). On the other hand, it is frightening when you consider that food for an average family of four is twice what the worker receives. And that's not counting rent, electricity, water, transportation....

News like this really makes me appreciate the (relative) security of the U.S. or Western Europe.

Not that I think you *aren't* underpaid. Just thought it would make you feel better if you knew it could be worse.

C++ 2.0

My newest toy these days (it just arrived last Friday) is Zortech's C++ 2.0 Developer's Edition. I've been up until 3 a.m. the last couple nights experimenting and reading. I've used Zortech 1.07 for the last year and wanted to find out what they had and had not fixed. So far most everything looks good.

The C++ Developer's Edition includes not just a C++ (native code) compiler, but also a similar C compiler, a Source level C++ (!) debugger, and a library of C++ classes to use in your own programs.

Also included is a TSR library that allows you to make your own programs into popup TSRs with (apparently) not too much work. I will refrain from saying anything good or bad about this until I've tried it. Two days just isn't enough time to sort through all these goodies.

As with just about every compiler on the

market, they've included an editor (ZED). Since I've become hopelessly hooked on Brief, though, I haven't done much except accidentally call it up once or twice from within the debugger. Because of this, I can't say anything to recommend or condemn it.

And, of course, there is the plethora of utility programs. A Linker, MAKE, TOUCH, OBJ2ASM, ARCHIVE, UPDATE. Even a MAKE file dependency generator. Everybody has to have a try at making a better wheel.

C++ Compiler

The Zortech C++ 2.0 release adds all the new features of the C++ 2.0 defacto standard (defined as "what is contained in AT&T's cfront 2.0") except for a few additions to the stream I/O library (which are "currently proprietary to AT&T"). The two most important additions to the language in version 2.0 are multiple inheritance and type safe linkage.

Multiple inheritance means that a derived class can inherit characteristics of more than a single base class. You might find this useful if, for example, you wanted to make a derived class Car that uses the facilities of both classes Body and DriveTrain. (Why you would want a class called Car is beyond me. Maybe you really need the class called JunkHeap which uses Smoke and Rust base classes.) You can see from this excellent example why multiple inheritance is useful.

Type safe linkage means that the type information of classes, variable, and functions is carried not just in the *.H files, but in the object file as well. This way, you avoid the following scenario:

- (1) compile srca.cpp which uses lib.hpp for type information;
- (2) modify lib.hpp and lib.cpp to change a function prototype;
- (3) recompile lib.cpp;
- (4) link srca.obj with lib.obj.

The problem is that if all type checking is done at compile time (as happens in C and C++ 1.0), it is possible to link two modules which have different ideas of what type certain pieces of data and functions are. This could lead to disastrous, unexplainable results.

In C++ 2.0, type checking is done not only at compile time, but also at link time. This is implemented in Zortech C++ (and AT&T cfront 2.0) by adding type information to the name of each variable (the compiler). That way if the

type information is different, you will receive an "unresolved reference" error from the linker.

C Compiler

In terms of its use as a C compiler, Zortech 2.0 has quite a few changes as well. Of course all these changes apply, by definition, to the C++ compiler, too.

MS C (MS Windows) Compatibility

Possibly the most important change is all the work done to make Zortech compatible with MS C (and Turbo C). You can now use Zortech C++ to develop MS Windows, OS/2, and OS/2 Presentation Manager programs. (You will also need to have the Windows Developer's Kit and/or the OS/2 Softset. The OS/2 version of the compiler is a separate upgrade, not included with the Developer's Package.)

Achieving this compatibility required, among other things, adding support for the "pascal," "near," and "far" keywords, and updating the code generator to generate the special procedure prologs and epilogs required by Windows.

There are a few incompatibilities, though. In MS C, normally SS==DS (although this must be changeable with an option to compile OS/2 and Windows DLL libraries, among other things). This is not the case in large data memory models of Zortech C, which put SS in a different segment to allow more room for global and static data.

This becomes a problem only when a near pointer is declared in a program where data is normally far. The near pointer is always assumed to point to DS; in MS C it can be used to point to an automatic or parameter variable (since SS==DS). It cannot in Zortech C, according to the C++ *Compiler Reference* chapter called "Converting Microsoft C Programs to Zortech C/C++."

Another problem is that Zortech does not support the HUGE data type; you can't have any single data object larger than 64K. I have always avoided HUGE arrays anyway (because of their inherent inefficiency), so that isn't a problem for me. You may not like making linked lists of arrays, or arrays of pointers to arrays, however.

Although Zortech has done a lot of work to ensure compatibility between their library and the Microsoft library, there's still a problem with open(). It "takes different parameters and behaves

differently than the Microsoft C version," according to Zortech.

I should point out, though, that just because Zortech's library is compatible with MS C's, it doesn't mean you can link Zortech OBJs with the MS library, or use third party libraries compiled for MS C! This is almost always *not* the case. Compatibility between Zortech and MS C is at the source (function prototype) level.

With some work, it is possible to compile a library that can be linked either with MS C or Zortech OBJs, but not many companies do it that way. Most have a separate version of their library, which is simply the same source code recompiled with the other compiler.

Finally, the "interrupt" function type available in MS and Turbo C is not available in Zortech. Instead they supply an Interrupt Package with their library which handles the installation of interrupt service routines.

Having used both methods, I prefer Zortech's; it doesn't force you to play around with saving the old vectors or worrying about whether to chain to the old interrupt. It will even set up a local stack for the interrupt if you like. Much handier.

There are a few other incompatibilities, all detailed in the manual. The ones I have listed are the most important, however. Most of the rest are just due to the fact that MS C 5.1 is not fully ANSI compatible. So programmers are forced into writing non-ANSI compatible code, which Zortech C doesn't like.

The big advantage of switching from MS C to Zortech is that now you can write your Windows and PM programs in C++ instead of mere C.

Compiler Performance

There's no reason for me to write about this. You can find all the numbers in the advertisements. I will say that the speed of everything is more than acceptable (I never was much for numbers) and the object modules Zortech creates are maybe 2% smaller than previous versions.

Whatever the numbers, they are definitely within Developer class. This is especially true for running the compiler, since it generates OBJs directly, rather than requiring the extra step of translating C++ to C, as most other C++ packages.

Of course, the optimizer takes some time. It can be lived with, though. Just

use it once after all debugging is done.

With Zortech 1.07, I once mistakenly reset my machine, thinking the optimizer had crashed. The program was one I had semiautomatically converted from Pascal, and it was filled with array subscripts and a 500 line case statement. Being used to non-optimizing compilers, after 20 seconds of no disk activity (I'm impatient), I assumed the machine had crashed. Later I figured out my mistake. I also rewrote the program.

C++ Debugger (ZDB)

Several issues ago, I did a mini-review of the original Zortech C Debugger. It was a promising product, but lacked many features I needed to make it my only debugger. The biggest problem was that I couldn't use it to debug assembly language programs.

In the ensuing months, rather than keep CodeView on my disk and switch constantly from one debugger to the other, I simply began to write more programs in C and C++, and fewer in assembly. When I did write an assembly language program that I needed to debug, I used good old SYMDEB (the debugger included with MASM until version 5.0).

Finally, with the release of ZDB, I have a single debugger for everything. As well as testing ZDB with several of my C programs, I also tested it with two programs written in 8086 assembly language and assembled with MASM 5.1. It even worked on a program for which I had no source.

Debugging In General

The original version of ZDB (called ZTCDB) could only handle C programs. While many of the improvements to the new version involve support for C++ and assembly language debugging, there have also been changes to many other parts of the program.

ZDB as a C Debugger is very reminiscent of ZTCDB, but with some nice additions. For starters, although ZDB still needs its debugging (symbol, line number, variable type) information in a different format than the standard CodeView information produced by most linkers, it now automatically does the translation rather than requiring you to run an extra program (ZTCMAP). It also now saves this information at the end of the EXE file rather than in a separate file.

Another nice change: when debug-

ging C programs in assembly mode (where source file lines are displayed interspersed with the assembly language instructions generated by the compiler), you can single step by assembly language statements. In the old version, you could only step from one source statement to the next, skipping over many assembly instructions in the process.

Also, you no longer need to worry about confusion between multiple static variables with the same name. These are displayed in the data window as "variable#file," where variable is the name of the static variable and file is the source file which contains it.

Another very convenient feature (if you can stand learning a new editor), is a command which puts you in the editor (ZED), editing the file currently in the source window with the cursor on the currently selected line. That makes fixing bugs much easier. Unfortunately, there is no facility for connecting this to any editor other than ZED.

There were cosmetic changes as well. For example, previously the menus were Lotus style (on two lines at the top); now they are pull down menus, making it much easier to see the available options.

But, as with most large, complex programs, there are still Cs & Bs (Complaints and Bitches).

First, ZDB assembly mode still displays locals and parameters (e.g., word ptr [BP-6]). On one hand, it is nice to see just how the variable is being accessed. (I can remember complaining about CodeView doing just the opposite because it concealed what was really happening.)

On the other hand, it confuses the issue. All the information is there (in the CV debug information) to display this properly. Why not at least make it an option? For example, it could be displayed in raw form when in the display mode where instruction bytes are listed to the left of the disassembly, and in symbolic mode otherwise.

Also related to assembly mode display is the habit of ZDB to believe that the last few assembly statements of the last function in a source file are actually in "No File." There is a kind of No Program's Land between each source file.

Another problem, which only occurs if you use SEC (Someone Else's Compiler) or SEA (Someone Else's Assembler) is that ZDB has no idea what to do with 186 opcodes. This is unfortunate, since MASM, among other programs,

generates some 186 opcodes automatically (if you tell it to do so). These opcodes are displayed as "?????" in the source window.

Yet another bug: after accidentally going into ZED (the editor) a few times when I didn't want to, I erased it from the \ZORTECH\BIN directory. I figured that ZDB would now give me an "Editor Not Available" error. Much to my surprise, it instead went into an endless loop of scrolling the screen up each time I pressed a key, with no apparent possibility of escape. I finally ctrl-alt-del'ed out of it.

It would be especially useful in ZDB's dual monitor mode (then I could turn all the lights down and have an office that looked just like Gordon Letwin's office on the cover of *Inside OS/2*). Of course ZDB does support 43 and 50 line by 80 column modes, but *only* on EGA and VGA cards. If they would just give me one day with their video library source....

Anyway, none of the above problems hamper normal operation. Overall, ZDB is much improved over the original. Just as I did when I first experimented with

BAR CODING SOURCE CODE BAR CODE IN C

Barcodes are now in widespread use. Don't let your application be left out!

Do you or your customers need a faster and more accurate method of data entry? Does your application need to print barcodes?

The *SymbCG* collection of C routines supports UPC (A & E), EAN (8 & 13), Code 39, 2 of 5 (Industrial & Interleaved), and Codabar symbologies. Routines are included for printing on HP LaserJet and Epson dot-matrix printers.

Full source code for *all* symbologies are included. No royalties.

Keyboard wedge barcode readers are also now available.

Only **\$99.00**

Symbologic
953 E. Colorado Bl. #342
Pasadena, CA 91106
(818) 449-7114

BONUS: Order *SymbCG* and a "MiniReader" barcode reader for \$350 and save \$50!

California residents please add 6.75% sales tax.

Foreign orders, P.O.'s, C.O.D.'s please call.

Reader Service Number 202

CodeView, I spent at least an hour last night watching the call stack and local variables bip up and down as it stepped through recursive calls to Factorial(). Maybe it will even be useful, as well as fun.

New Debugging Modes

As mentioned above, ZDB can now debug not only C programs, but also Assembly and C++ programs, and those with no source at all. The only thing it can't do is debug a packed EXE file. (The EXEPACK utility included with MASM compresses these files. Nobody else can directly debug them either.)

Sadly, it also can't debug a smelly cat.

Debugging Without Source

When ZDB is asked to debug a file for which it can find no source, or no debug information, it starts up in raw assembly mode. That means it works like a stroked-and-bored DEBUG, in a screen oriented fashion (with a memory display window, disassembly window, register window, conditional and unconditional breakpoints, tracepoints, etc.)

Debugging Assembly With Source

To test the ability of ZDB to debug assembly language, I used the FACT.ASM and FACT2.ASM programs which I had given as examples when I talked about MASM 5.1 and CodeView in Issue #45. These programs were written using Pascal calling conventions (arguments are pushed left to right, rather than right to left as in C). They still loaded into ZDB and executed with no problem. Almost.

Even things that the ZDB manual said wouldn't work went without a hitch. For example, I had declared a variable to be FAR PTR WORD (yes, MASM can do that). The manual said this wouldn't be displayed properly in the Data window (it would display the value of the pointer as an integer, rather than the value at the address pointed to by the pointer).

Instead, I was cheerfully greeted by a display of the pointer in segment:offset form and, after pressing the Ins key, was shown the value at the location it pointed to. The contents of local variables (automatic, stack) displayed correctly, too.

The only rough spot was that ZDB did not correctly recognize any function with no parameters or local variables. For example, when I called up FACT, which has no locals in main(), ZDB started in assembly display mode rather

than source display mode. I switched to source mode and hit the Single Step key, only to see the program branch off to nowhere. (ZDB did not, however, branch off to nowhere. It politely allowed me to reload the program).

The fact that ZDB expects all functions to set up the BP register to point to their local stack frame (local variable space) causes this problem. For some reason, it gets confused about which source line goes where if this isn't done. Unfortunately, MASM does not set up the BP register if a function has no locals or parameters.

I tried doing the same thing again, this time leaving the display in assembly language mode, and it stepped through the program just fine. It also worked perfectly in source mode with FACT2, which has a local variable in main().

When ZDB loads a program, it looks for the symbol _main (or main). If it finds that symbol, and if the normal

```
PUSH BP
MOV BP, SP
```

instructions are at that address, it starts debugging in source mode with the instruction pointer (IP) sitting on the instruction just after the above stack frame setup code. If it doesn't find one of these conditions (or if you specify the /a switch), it starts in assembly language mode with IP sitting at the very first instruction to be executed.

Starting at main() is useful for C and C++ programs. That way you skip over all the library startup code (which is supposedly already debugged anyway). I usually don't call the starting point in an assembly language program main, though. That is, I didn't use to.

Now, if I want the debugger to start in source mode, I must name my main routine "main." Not only that, but I must assemble with the /mx switch to turn on case sensitivity, since ZDB thinks that MAIN and main are different (and rightly so).

Another problem with FACT and FACT2 was that, as mentioned above, ZDB doesn't recognize 186 opcodes. I had originally set the programs to use 186 opcodes (with the .286c directive). This caused MASM to use the ENTER instruction instead of the function prolog I showed above. ZDB displayed this as "????." I removed the .286c and reassembled, eliminating the problem.

These aren't big problems, though

(with the possible exception of the non-recognition of 186 opcodes). If all I must do is name the start address of my program "main" and use /mx to debug assembly language with ZDB, I can live with it. Happily.

Debugging C++

This is the real exclusive of ZDB. Although other companies may have source level C++ debuggers in development, or maybe even secretly on the market, Zortech has the first source level C++ debugger for MS-DOS that I've seen.

You may think that any multi-language debugger should be able to debug C++ just as well as C, Pascal, or any other language. That's kind of true. For example, I can compile a C++ program with Zortech and load it into CodeView. It will step through the program with no problem.

But just try looking at a function name! Say you have a class called Car with member function Crash(int mph, long *damage). You would expect the label for Crash to be _Crash. Not so! Instead, the label is _Crash_3CarNipl. The "3" tells how many letters are in the class name, "N" means it is a near function, and "ipl" are the argument types (integer, and pointer to long). Obviously this could get tedious.

Also, while many debuggers have facilities for dealing with struct data, no others can properly handle instances of classes. As an example, if you expanded an instance of Car, you might see the value of Model, but not of Engine (if Engine is really a member of the base class DriveTrain).

ZDB has special C++ debugging support to handle these problems. Most important in this support is name unmangling, which simplifies function name display, and a new window: the Class window.

The Class window allows convenient probing through an interrelated set of class definitions. Consider that each class definition may be a derived class based on several other class definitions, which may themselves be based on several other classes.... The class window helps to sift through these connections and, if desired, put up the source of a member function in the debugger's source window.

Of course, the Class window just shows definitions of classes and member functions. You still must use the Data

Figure 1—KBD.C

```

/* kbnc.c - a program to enable the keyboard and optionally
   execute a command, then disable the keyboard.

   written by Laine Stump Jan ??, 1990
   No Rights Reserved. */

#include <stdio.h>
#include <ctype.h>
#include <process.h>
#include <dos.h>
#include <int.h>

extern int _okbigbuf = 0; /* these reduce resident program size */
extern unsigned _stack = 1024;

/*-----*/
void kbnc(void)
{ /* enable input from the keyboard */
  outp(0x64, 0xAE); /* enable keyboard command */
#ifdef DEBUG
  puts("kbd ON");
#endif
} /* kbnc() */
/*-----*/

void kboff(void)
{ /* disable input from the keyboard */
  unsigned ct;

  while(bioskey(1)) bioskey(0); /* clear out keybuf */
  for (ct = 0; ct < 65535; ct++) /* wait for key release */
    ;
  outp(0x64, 0xAD); /* disable kbd port */
  while(bioskey(1)) bioskey(0); /* clear kbuf again to be sure */
#ifdef DEBUG
  puts("kbd OFF");
#endif
} /* kboff() */
/*-----*/

int breaktrap(struct INT_DATA *pd)
{ /* 0 return means "call original handler" */
  return (1); /* 1 return means "don't call original" */
} /* breaktrap() */
/*-----*/

int getkbflag(char *arg)
{ /* process the string at arg to determine if we want the
   keyboard turned disabled (/d) , or enabled (/e)
   Returns 0 for /d, 1 for /e, and -1 for error */
  if (*arg++ == '/')
    if (toupper(*arg) == 'D')
      return(0);
    else if (toupper(*arg) == 'E')
      return(1);
  return(-1);
} /* getkbflag() */
/*-----*/

void usage(void)
{
  puts("usage: kbd /e|/d [command]\n");
  puts(" where /e enables keyboard input, /d disables keyboard");
  puts(" input, and the optional 'command' is a dos command to");
  puts(" execute while keyboard is enabled/disabled. If command");
  puts(" is specified, the keyboard is disabled/enabled after the");
  puts(" command is executed.\n");
  exit(1);
} /* usage() */
/*-----*/

int main(int argc, char *argv[])
{
  int retval, kbflag;

  if ( (argc < 2) || (kbflag = getkbflag(argv[1])) == -1)
    usage();
  int intercept(0x23, breaktrap, 256); /* make sure we don't get */
  if (kbflag) kbnc(); else kboff(); /* ctrl-Broken before we */
}

```

and Automatic (local variables) windows to examine the contents of instances of a class. The functionality of these windows has been upgraded to allow the same traversing operation to see the values of member variables of a class instance.

C++ Tools

The C++ Tools include classes for queues, singly and doubly linked lists, stacks, bit vectors, virtual arrays, hashed search tables, windows, and all kinds of other things. Included is a book which explains each class, gives examples of its use, and gives the complete source listing for the class (the source is on disk as well). If nothing else, the C++ Tools are good examples for those just starting with C++.

I had wanted to play with these classes some and get a feel for how useful they were. But, as with the TSR package, there just wasn't enough time. Rain check.

Documentation

The old version of Zortech had a single manual for the compiler and library and another manual for the debugger. This was more or less adequate, but I still found myself reaching for my old Turbo C 1.0 manual now and then (especially in the old days, when I was weaning myself from Pascal—it has a great little section showing equivalent operations in C and Pascal side by side).

The manual set for 2.0 has a separate manual for the compiler and utilities, the libraries, the debugger, and the C++ Tools. I've done quite a bit of thumbing through them in the last two days and have found most of the information I needed. There is even a thorough explanation of symbol construction in C++, and many other implementation details that almost nobody would need. But when they did....

There are sections both on AT&T C++ compatibility and MS C compatibility. It also has a section titled "Incompatibilities with ANSI C," which deals *not* with incompatibilities between Zortech and ANSI, but with incompatibilities between *any* C++ and ANSI C. This is very useful reading for a new convert from C.

Price

Unfortunately, the \$29.95 JRTs and \$49.95 Turbos are a thing of the past. Well, not really. I've been seeing JRT ads again recently but, not able to believe,

thought that I had entered a strange time warp like one of those episodes from "Twilight Zone." Seriously, I thought the magazine had misplaced some of the copy from their April issue.

The problem is that the amount of development (and the amount of advertising dollars) that has gone into this kind of software has skyrocketed since the days of JRT and Turbo 1.0.

Even Borland realized long ago that \$49.95 per copy wasn't even keeping the lights on, much less paying for the \$10,000 *BYTE* ads and financing all those hot tub parties and hotel suite bashes so essential in modern (marketing) warfare. The price of Turbo Pascal has risen with each new release (it seems like the update fee for the new edition is about the same as the retail price of the previous). I stopped at 3.0.

The retail price of the Zortech Developer's Package 2.0 is \$450. The C++ compiler by itself is \$199. Prices from discount houses will probably run considerably less.

On the other hand, the functionality of JRT Pascal, or Turbo Pascal isn't even close to the functionality of Zortech C++. Although considered breakthroughs in their time, they are now mere toys.

Also, the retail price of Microsoft C 5.1 is about the same. And it's just a C compiler, no C++ (it does include OS/2 support, however, which is a \$150 upgrade from Zortech).

Still, it isn't even fair to compare Zortech C++ with MS C. Or any C, for that matter. Because it isn't C. It's C++. And it has a debugger that is light years ahead of the last version I've seen of CodeView.

Even if the price was a bit higher (which it isn't), it would be worth it in all the development time saved. After all, how many days does it take you to make \$450? Don't you think it's about time you worked a few nights so you could afford to have some *fun*?

Zortech Inc.
1165 Massachusetts Ave.
Arlington, MA 02174
Voice: (617) 646-6703
Fax: (617) 643-7969

Batch File Prophylactics

Before closing this thing up and tossing the smelly cat out for the night (you thought I was just kidding about smelly cats), I want to pass along a little "cute-C" utility I wrote. I compiled it with

```

if (argc > 2) /* get a chance to kboff() */
{ /* execute program */
  retval = spawnvp(0, argv[2], &argv[2]); /* execute command */
  if (kbflag) kboff(); else kbom();
}
else
  retval = 0;
int _restore(0x23); /* restore DOS ctl-Break */
return (retval); /* send back exit value of command */
} /* main */
♦♦♦

```

Zortech, but it will work with MS or Turbo with just a few changes in places I have pointed out in the source. See Figure 1.

KBD

KBD is a program that can enable and disable the keyboard on a PC compatible. This can be useful when writing a BAT file which you don't want the stupid user to screw up, or running a program which you don't want the... You get the idea.

The syntax of KBD is:

```
kbd /e|/d [command]
```

/e enables the keyboard, /d disables the keyboard, and command is an optional parameter, which is a DOS command to execute while the keyboard is enabled/disabled. If command is given, the keyboard is turned back off/on when the command finishes. For example:

```
kbd /d
```

turns the keyboard off. If you type this at the DOS prompt, the only way you can get the machine going again is the reset button or the Big Red Switch! Usually you would do it at the beginning of a sensitive batch file. Another example:

```
kbd /d chkdisk
```

or

```
kbd /d brot ...
```

(I thought Larry might appreciate that second one.) These commands turn the keyboard off while executing *chkdisk* (or *brot*), then turn the keyboard back on. As I said, KBD is mostly useful in BAT files (especially *AUTOEXEC*).

As an example, say that you have a BAT file that gets a user's name and

password with the command *LOGIN* before continuing. For example's sake, *LOGIN* returns 0 if the login was successful, or 1 otherwise. You want to put *LOGIN* in a BAT loop that looks at *errorlevel*. The danger is that the user can simply type *ctl-C* and break out of the batch file. This is where *KBD* comes in. For example, you might use this BAT file:

```

echo off
break off
kbd /d
:loop
kbd /e login
if errorlevel 1 goto loop
kbd /e
break on

```

Notice that, although "break off" minimizes the number of times DOS checks for *ctl-C*, it doesn't eliminate it. That is what makes *KBD* essential. *KBD* also traps the DOS *ctl-C* interrupt (23h) while it is executing the command, making sure nobody can *ctl-C* out of that program.

Of course, you have the problem that, while the first two lines of the BAT file are executing, the keyboard is still enabled. A good way to solve this is to make a device driver that turns off the keyboard. This way the keyboard is disabled before the user has any chance to screw things up. I'll leave that one as an exercise for the reader.

How?

KBD turns the keyboard on and off by outputting a command to the keyboard controller port (64h). *0ADh* turns off the keyboard, and *0AEh* turns it on. This physically disables *anything* from the keyboard. Even *ctl-alt-Del* falls on deaf ears.

Interrupt Package

Another interesting part of *KBD* is

the use of the `int_intercept` function from the Zortech interrupt package. The interrupt package handles all the details of saving the original interrupt vector, putting the new one in place, setting up a stack for the service routine, chaining to the old service routine if requested, and restoring the old vector when finished.

Note that `breaktrap()` (my replacement for the `ctl-C` interrupt) returns a 1. This means "don't chain to original interrupt." If it returned a 0 it would mean "chain to the original interrupt now, then return." Since our entire reason for trapping the interrupt was to keep the original from being called, we don't want to return 0.

DOS Exit Values

In the example above, we used error-level to check the exit value of the LOGIN program. But the program being executed (by the BAT file) is KBD, not LOGIN! How can I do that? With the clever trick of using one of the `spawn()` functions (`spawnvp()` in this case), which returns the program's exit code. Then I return that value from `main()` in KBD

(the return value of `main()` is sent to DOS as the exit value of the program).

`spawnvp()` searches the DOS path for the command and can use the arguments in `argv[]` without modification, but it cannot call a BAT file. If I wanted the ability to execute BAT files, I could use `system()` instead, but then I wouldn't get exit values. (`system()` uses `command.com`, which eats the exit value.)

`_okbigbuf` and `_stack`

The declarations of the variables `_okbigbuf` and `_stack` at the top of KBD.C are included to reduce the resident size of the program. This leaves more memory for the program called with `spawnvp()`.

In Zortech C, when `_okbigbuf` is 1 (the default), the maximum 64K of data space (for small data models) is allocated on program startup. If `_okbigbuf` is 0, data space is only allocated from DOS as needed. `_stack` controls the size of the program's stack. By setting these two variables, I was able to trim the resident size of the program from 72K to 10K. A big improvement for two lines!

XT SCHEMATIC

Micro Cornucopia

Phone Orders:
(503) 382-5060, M-F, 9 AM-5 PM PST

Mall Orders:
P.O. Box 223, Bend, Oregon, 97709

IBM PC-XT Schematic . . . \$15.00

As always, you can do anything you like with KBD. Even swear at it if you want. I simply don't care.

Pseudoepidermatological Meanderings

Sorry, but it's nearly 2 a.m., and you can't expect everything I say at 2 a.m. to be coherent. The column is finished now. But it's cold out, and the smelly cat looks so pitiful. (If she just wasn't about to explode with babies.) I guess she can stay inside tonight. But *just this once*. (Anyone want a Turkish kitten?)

◆ ◆ ◆

ICs PROMPT DELIVERY!!!!
SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN for FEB. 25, 1990

OUTSIDE OKLAHOMA: NO SALES TAX

DYNAMIC RAM			
1MB	MEM DeskPro 386/20		\$295.00
SIMM	AST Prem386/33Mhz		225.00
SIMM	1Mx9 80 ns		92.00
SIMM	256Kx9 100 ns		35.00
1Mbit	1Mx1 80 ns		8.75
41256	256Kx1 60 ns		4.15
41256	256Kx1 80 ns		2.50
41256	256Kx1 100 ns		2.20
41256	256Kx1 120 ns		1.95
4464	64Kx4 120 ns		2.50
41264*	64Kx4 100 ns		7.50
EPROM			
27C1000	128Kx8 200 ns		\$18.00
27512	64Kx8 200 ns		7.80
27256	32Kx8 150 ns		6.50
27128	16Kx8 250 ns		3.75
STATIC RAM			
62256P-10	32Kx8 100 ns		\$10.00
6264P-12	8Kx8 120 ns		4.50
6116AP-12	2Kx8 120 ns		4.25

IIT 2CB7-20 \$395.00
 IIT 2CB7-12 \$295.00
 IIT 2CB7-10 \$260.00

8087-2 \$110.00
 8087-8 \$210.00
 8087-16 \$110.00
 8087-25 \$110.00
 8087-33 \$110.00

* 2-PORT VIDEO RAM

OPEN 6 DAYS, 7:30 AM-10 PM. SHIP VIA FED-EX ON SAT.

SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY: Th: \$2 \$6.25/4 lb Fr: P-1 \$16.50/1 lb	MasterCard/VISA or UPS CASH COD MICROPROCESSORS UNLIMITED, INC. 24,000 S. Peoria Ave., (918) 267-4961 BEGGS, OK. 74421 No minimum order. Please note: prices subject to change! Shipping, insurance extra, up to \$1 for packing materials.
--	--

Reader Service Number 37

DEVELOP YOUR OWN

BOOTABLE OPERATING SYSTEM

WITH BOOTER TOOLKIT™

- Real-time scheduler
- DOS-compatible file system
- Window system
- Memory manager
- Bootstrap loader
- 100+ page user manual
- **ONLY \$99 WITH SOURCE CODE**

ORDER HOTLINE

(206) 391-4285

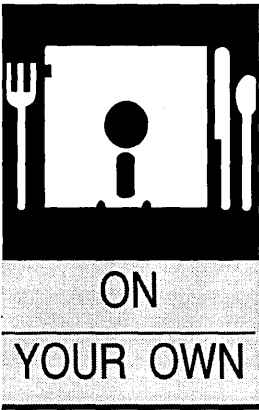
For immediate shipment, send your check or money order for \$99 to:

GENERAL SOFTWARE™

General Software
P.O. Box 2571, Redmond, WA 98073

Telephone orders shipped UPS BLUE COD only. To order by credit card, call Programmer's Connection or Austin Codeworks, authorized General Software dealers. Second-day air shipping provided free to destination only within the United States. Foreign orders add \$20.00 for airmail delivery. Washington Residents add 8.1% sales tax. Copyright (c) 1990 General Software. All rights reserved. Booter Toolkit and General Software are trademarks of General Software.

Reader Service Number 212



You're On Your Own

By David Thompson
Laura Shaw &
Nancy Ellen Locke

Micro C Staff

Well, here it is: probably for the first time in the history of computers, an article in a computer magazine on advertising in other computer magazines. So many advertisers have asked for suggestions of where to advertise that we've put together this information.

Who's Offering The Best Deal?

The standard way of judging what kind of deal you're getting on advertising is to divide the one-time B&W full-page price by the circulation (in thousands). That's the cost/thousand. That way you can compare rates from smaller magazines with rates for the big guys.

Once you figure the cost/thousand readers, you can factor in the less precise variables. What percentage of the audience would be interested in your product? How much training would they require? How much will you have to explain in the ad? (You can say ICE to designers; you have to explain what it means to managers.) Would they pay what you're asking?

Be aware that corporations look for image and support, hackers look for performance and price.

But when you're all done, the real question is: Will advertising your product in *Computer Language* or *Programmer's Journal* or *PC Techniques* be profitable? If you make money in all three, then advertise in all three. If your product doesn't make money anywhere, maybe you need a new product.

As for what to expect, check out "On Your Own" in Issue #51. Don and Kim Jindra's experience advertising their \$25 network should give you a good idea what happens.

My advice for prospective advertisers is:

(1) Figure out how much space you need to describe your product. If you're selling something that's easy to describe, you don't need much space. If it's a totally new idea, or a substantial improvement on an old idea and requires a lot of explanation, plan to buy a lot of space.

(2) Figure out how much image you need to buy. (This is very different from #1.) Let me give

you an example: if you're selling a \$1,500 CAD package, be prepared to buy full-page four-color ads and find the best ad agency to produce them. Your ads must create a very solid image for your product. If you're selling a \$25 network, a simple micro ad may draw more orders than any other size.

(3) Make sure your ad gets seen. (This sounds a bit like #2.) If you only need a tiny ad (a cheap product that's easy to describe), you may find that a micro-sized ad grouped with other micro ads may be better read than a quarter page hidden among the articles. If you need at least a half page to tell your story and you're afraid of getting lost, then go for a full page. Full pages are very visible. Four-color full pages are most visible.

Also, if you're buying a half page or more, you might as well act like an experienced advertiser. Ask for a right-hand page, outside corner, and ask them to place it with a particular column or article. (If it's a very large magazine, ask them to place it in the front third). They'll be glad to charge you extra to guarantee you a particular position, but ad people can often finagle you a visible spot, no extra charge, if you simply mention you'd like it.

Plus, you may as well tell them you're acting as your own agency. That way you'll get the standard 15% agency discount, off the top.

(4) Does the ad have to make the sale? Not always. You can send out a lot of information in a 1-ounce letter. Just make sure your ad is as good at weeding out weak prospects as it is at attracting strong ones.

(5) Something other than ad space? There's direct mail. Figure on spending anywhere from \$.75 to \$1.50 per piece. (That counts label, envelope, return envelope, literature, postage, design....) Also figure on a 2% to 5% response, so you'll spend from \$15 to \$75 on each order.

Generally, magazine ads are a faster, cheaper, easier way to reach a lot of people. Mail works best as a follow up to enquiries generated by your ads and for selling additional products to recent purchasers.

Now That You're Going To Advertise

(1) Order media kits. The kits contain the latest ad rates (rates can change from one month to the next), ad deadlines, ad sizes, how they should be produced....

(2) Watch the deadlines. If you ask for space, you'll probably have to pay for it even if you don't get your materials to them on time. Stay on top of your ad. It may be obvious to you and me that ads need to change as products change. But you'd be surprised how many small companies run outdated ads for six months simply because they keep missing deadlines.

(3) Make sure your ad fits. A lot of people send in incorrectly sized ads.

(4) Make sure it looks good. You want solid black type, clean lines, white paper, no smudges, no grey, no dirt.

(5) Keep a second copy of your ad (a high-quality, printable copy). Remember, Murphy works for the U.S. Post Office and his brother runs Federal Express.

(6) Check everything for accuracy (particularly the phone number). I know more than one old lady who's been inundated with orders for hard drive utilities.

(7) If your product runs under \$100 and your ad's doing the selling, you may not want a reader service number. I know several companies who have been crippled by the costs of replying to reader service numbers. (They assumed most of the recipients would respond; turned out few, sometimes none, ordered.)

(8) Talk to the magazine's editor. Ask him if he thinks you'd do well advertising there. Also, ask him if there's someone you should send a copy to. (Don't,

absolutely don't, mention that your decision to advertise depends on his decision to review the product. You won't feel good about it and he won't either.)

Once The Ad Has Run

(1) Ask everyone who calls where they heard about your product.

(2) Make sure you can respond immediately to orders. We started Micro C out of our house and we lived on a dead-end road. We'd meet the mailman as he stopped at our box and many times we had reply cards addressed and ready to hand back to him by the time he'd gotten back to our house. We had 10-minute turnaround and subscribers loved it.

(3) Be prepared to lose money and patience on some orders. Occasionally you'll run into a real bother. He'll want a custom product, he'll want you to install it, he'll call you day and night instead of reading the manual, he'll erase his disks; and after you finally return his money, he'll write a nasty letter to your state's attorney general suggesting that the death penalty might be too lenient. (The guy was from New York City, of course.)

Note: Circulation figures are, at best, optimistic. When you're on the phone with a new magazine, ask for a breakdown of their circulation. (Ask the questions in the following order.)

(1) Is your circulation audited by ABC or BPI? If so, great; have them send or fax you the latest publisher's statement and skip the following questions.

(2) How many paid subscribers in the most recent issue? (Not *next* issue, or the end of *next* year.)

(3) How many copies of the most recent issue were sent to the newsstands?

(4) What percentage of the newsstand

copies normally sell? (40% to 50% is normal, 60% is high, 70% is almost unheard of)

(5) What was their total print run for the most recent issue?

Listen for pregnant pauses, hemming and hawing, fudging, explaining, etc. If the totals of #2 and #3 add up to #5, then they're probably fudging. Either way, take printed circulation figures with a giant grain of salt.

Also, don't be afraid to call a magazine's advertisers. They'll tell you a lot. Some magazines are comfortable to work with, others aren't. Some will make deals, others won't. Sometimes companies find the best response in brand new magazines, other times they do best in the old timers. In some publications the response is immediate, in others it takes three ads. Sometimes ad placement is very important, other times it isn't.

You'll be amazed how much information you can glean in half a dozen calls.

The Carrot

Of course, there's no feeling like opening your post office box and seeing it full. Really full. Tends to make all the hassles, all the preparation, worthwhile. Reminds me of a little known saying that's been handed down from hacker to hacker over the centuries.

"May your bugs be gentle, may your CRT be warm upon your face, and may your mailbox be ever filled with checks. Good checks."

Writers Market

Speaking of checks in the mailbox, there's more than one way to squeeze cash out of a computer. If writing software isn't your bag, then how about articles? The most effective writers:

(1) Have a specialty: processor design, graphics algorithms, database diagnostics, entrepreneurial experience....

(2) Have a conversational style: An open, easy style rather than a stiff, academic, jargonizing (agonizing) presentation.

(3) Write for the magazines they keep in their bathrooms.

(4) Don't take themselves too seriously. I'm talking about both their dealings with editors, and the way they write.

I wish you well, my friends.



DEVELOPMENT LIBRARIES FOR PC PROGRAMMERS

Source always included!!!

AVAILABLE FOR PASCAL, ADA, MODULA-2 and C

product	per language	all languages
• Btree—core and disk versions	\$42.95	\$139.00
• Lists, stacks and queues	\$34.95	\$109.00
• Trees—binary and heaps	\$42.95	\$139.00
all packages	\$95.00	\$295.00

order direct:

VISA
MASTERCARD
AM EXPRESS

REGAN SALES
PO Box 2204
Bothell, WA 98041-2204
(206) 820-2603

Reader Service Number 210

MARKETING ALTERNATIVES

Magazine	Publisher	Address	Contact	Phone	Circ.	Ad Rates
The C Gazette	Andrew Binstock	1341 Ocean Ave. Santa Monica, CA 90401	Andrew Binstock	213-473-7414	6,000	Full page \$450 1/2 page 230 1/4 page 130
Midnight Engineering	Bill Gates	111 E. Drake Ft. Collins, CO 80525	Bill Gates	303-491-9092	15,000 projected	500 250 175
PC Techniques	Keith Weiskamp	202 E. Greenway Phoenix, AZ 80532	Jeff Duntemann	602-493-3070	20,000 projected	1,270 775 420
Tech Specialist	Robert Ward	2601 Iowa St. Lawrence, KS 66047	Donna Ward	913-841-1631	10,000	1,150 575 290
The C Users Journal	Robert Ward	2601 Iowa St. Lawrence, KS 66047	Donna Ward	913-841-1631	27,000	1,150 575 290
Circuit Cellar	Daniel Rodrigues	12 Depot St. Peterborough, NH 03458	Rose Mansella	203-875-2199	20,000	995 600 350
Programmer's Journal	Liz Oakley	PO Box 31060 Eugene, OR 97403	Jay Moore	503-747-0800	30,000	1,895 1,195 645
Computer Language	Regina Ridley	500 Howard St. San Francisco, CA 94105	Regina Ridley	415-397-1881	70,000	3,270 1,940 1,145

The C Gazette is a code-intensive C and C++ quarterly for MS-DOS users. It specializes in in-depth articles aimed at the experienced C programmer. Code supports the major compilers and is accompanied by clear, cogent prose. The C Gazette is committed to the exploration of serious programming topics. Bruce Eckel of Micro Cornucopia has been the Gazette's C++ editor for the last year.

Midnight Engineering, the journal of personal product development, is the magazine for hardware and software developers who want answers about developing and marketing their own products. Coverage includes just-in-time development, product pricing/marketing, and tips/techniques for working "on your own." Published bi-monthly, a one year ME subscription costs \$19.95 for the introductory period.

PC Techniques is designed for a market of proven buyers of programming languages, tools, application software, hardware, and more. Readers are interested in the practical applications of programming related software products. Every issue will present hands-on software reviews, programming tips, and practical advice from leading software developers and authors.

Tech Specialist is written for advanced PC developers. This powerful group of professionals works in both software and hardware development, using the stand-alone PC as a tool. Tech Specialist gives PC developers a "real world" tool - integrating information about hardware manipulation, software design, and the internals of the PC. Each issue provides an in-depth analysis of specific development problems.

The C Users Journal provides an interactive forum for C programmers. The Journal publishes practical information for C programmers,

including advanced and intermediate programming hints and techniques, tutorials, and information about software tools. Each issue features major "how-to" articles written by experienced C programmers, C product reviews, book reviews, correspondence from members, and information on new products.

Circuit Cellar INK, The Computer Applications Journal, is the premier source of practical technical information for designers and builders of computer hardware and software applications. Circuit Cellar INK enhances the electronics design skills of its readers by offering creative solutions and unique applications through complete projects, practical tutorials, and useful design construction techniques.

Programmer's Journal publishes articles written by the industry's top professional talent - working programmers who are willing to share their valuable technical expertise for the benefit of fellow colleagues. Every other month, professional developers worldwide go to PJ to find language articles with useful source code on Ada, ASM, BASIC, C, C++, COBAL, FORTRAN, FORTH, Modula-2, and Pascal - plus the latest information on OS/2, the 80386/486, the EGA and VGA, new product releases, industry news, and consulting tips.

Computer Language is the productivity magazine for professional software developers. Its mission is to enhance the programmer's ability to generate working applications on schedule. Each issue of Computer Language includes columns devoted to application design and analysis, productivity tools, user interface design, object-oriented programming, and "how to" articles written by the finest professional programmers in the field.

Letters *continued from page 6*

figured it might not be the place for an urban keyboard potato like me. I may have been wrong, but about the rafting: would I actually have to go outdoors for that, or could you digitize the experience and send it to me on a floppy? (By the way, does SOG stand for Sages, Oracles, & Gurus or Silly Old Geezers?)

Anyway, thanks for putting out the best computer mag around. And don't give a second thought to that volcano when you're trying to get to sleep. The chances of it suddenly exploding into a violent torrent of boiling lava and poisonous gases and burying the entire town under glowing cinders are really small. Less than 1 in 10, probably.

M. N. Macleod
3043 S. Laredo Circle
Aurora, CO 80013

Editor's note: SOG stands for Anything You Like—no, that's not it.... I suppose you know that Bend almost closed when you left. Fortunately, the other resident stayed. As for the dead volcano, the humane society picked it up last weekend. (The smell was getting awful.)

Correction

As if the discussion isn't confusing enough already, we managed to slip an error into James Martin's letter on the radar equation controversy (see *Letters*, Issue #52).

At the bottom of the right hand column on page 78, we refer to the total power at the detector as proportional to the inverse cube of R. Wrong. The correct term (as stated in James's original letter) is the inverse *fourth* power of R.

How can this 25% power loss have happened? Perhaps we're seeing proof of the First Law of Preservation of Postal Power.

The PPP theory states that in all Postal interactions, power is conserved. In other words, the Postal Service's proposed 25% increase in rates will be balanced by a corresponding *decrease* in all mail contents. Good news for all bill payers, but be sure to tell your employer to stop sending your checks through the mail.

We hope this will serve to clear up the confusion and save James' sullied reputation.

Micro C Staff

More UNIX

Bob Morein's "UNIX Packages For The PC" in Issue #50 was exactly the type of information I've been looking for. I turned up several other articles as well as two useful books. (See my list at the end of this letter.)

One of the things I'm discovering is that UNIX is quite particular about the hardware. For example, at least three flavors of UNIX (and XENIX) do not support RLL hard disk controllers, nor do they support many video display boards.

I haven't decided which UNIX I'll purchase. That decision will have to wait until I find out more about which flavors support which hardware. Meanwhile, I hope you will have enough responses to Bob's article to encourage more articles in *Micro Cornucopia*.

Fiedler, D., "Future Imperfect," *BYTE*, May 1989, p. 113.

Fiedler, D., "Calm Approach to UNIX," *BYTE*, Aug. 1989, p. 113.

Combau, G., "The UNIX Shell," *BYTE*, Sept. 1989, p. 315.

Unger, J., "One Man's Experience," *BYTE*, May 1989.

Morein, B., "UNIX Packages for the PC," *Micro C*, Nov.-Dec 1989, p. 40.

Fiedler, D., "UNIX on Personal Computers: Why and How," *BYTE*, Sept. 1989.

Christian, K., *The UNIX Operating System*, 2nd edition, John Wiley and Sons, 1988.

Fiedler, D. & Hunter, H., *UNIX System Administration*, Hyden Books, 1986.

Larry Hoffman
10152 Oso Ave.
Chatsworth, CA 91311

Editor's note: Wonderful list, but you missed the original tome on the subject: The Hardware Requirements Of Eunuchs, Xinu, MLVX BC.

Op-amp Exposé

Dear Mr. (smug) Editor—Regarding your comments to my letter in Issue #52: I do not appreciate your condescending and insulting response to what was intended as a constructive correction. Turnabout being fair play, let the games begin!

Obviously my letter isn't the only thing you've ignored; OpAmps-101 must have occurred coincident with

your nap time. For your sake (as well as others who have yet to tame the beast), I shall endeavor to put into words the secret of life, op-amps, and everything.

The op-amp is a device with two (count them) inputs: one inverting (-) and one non-inverting (+). It also has one output (isn't that wonderful?). Now, whatever happens on the non-inverting (+) input causes the output to move in the same direction. Amazingly, whatever happens on the inverting (-) input causes the output to move in the opposite direction. I guess that's why they call it the inverting input.

The real secret (I know you've been waiting for this) is: the op-amp will swing its output, hoping against all hope, waxing brave in this cold, cruel universe, that somehow, somewhere, somehow there is a feedback path that will bring its inputs into balance by virtue of its efforts. This is usually done by providing feedback to the inverting (-) input. Yes friends, indeed, that little booger won't rest until its inputs are at equilibrium. The same. Identical. Twins. You get the picture.

Knowing what we now know, let's walk through the circuit as published with a test case. Suppose we let $V_1=1.010$ volts and $V_2=1.000$ volts. By the equation stated ($V_{out}=V_1-V_2$) we should expect V_{out} to equal 0.010 volts. The (-) input will be at $V_2/2$ (0.500 volts). The (+) input has... well, it depends where the output was to start with. (V_1 and V_2 in the original article should be swapped for the equation to apply. Change the designations in either the schematic or the equation, but not both.)

Let's say the output was at 0.000 volts just a microsecond or two before we applied the input. Since 0.000 volts is as good as ground, the (+) input is at $V_1/2$ (0.505) volts. The inputs aren't balanced so the output will simply rise until they are. (Ooh, we've got me now!) There, see! The (-) input is at 0.500 volts, and the (+) input is uhm, er, even bigger than it was before! (Just a minute, let me grab my calculator.) Nope, the inputs still aren't balanced.

"Hmmpf" says Mr. op-amp, "I'll fix that. I'll just raise my output even more, and hope like crazy that somehow my (-) input will rise to equilibrium with my (+) input." Things are even worse; the (+) input rises higher still.

"Now my inputs are really getting out of balance. I'll show them!" Mr. op-amp zings his output as high as he can go. "Surely my inputs will be balanced now."

Sadly, Mr. op-amp has pegged his output, his (+) input higher than ever, and his (-) input still with only 0.500 volts on it. Mr. op-amp does the only thing he can do. He keeps on holding his output high, for surely somewhere, somehow, someway his (-) input will rise, and he waits. He will be waiting a long time.

Yes, op-amps are a bit more complicated than that, but you won't stray far from what's really happening if you keep "the secret" in mind. Maybe if anyone (else) cared, mine would not have been the only letter pointing out the mistake. After all, this is a magazine about micros; who cares if you lie about anything else?

What you presented in Issue #49 was little more than a comparator (and a poor one at that—too much hysteresis). If you still don't believe me, build it. Perhaps in your haughtiness, *you*, not I, were thrown by the inverting (-) input not being on top. In closing, may I suggest you either stand corrected or sit in shame.

Dave Stojan
10310 Lybert Rd.
Houston, TX 77041

Editor's cursory response: At first I was kind of plus-minus about your letter, but I like the way you amplified the subject.

Oh, Go Take A Flying @##%*

You very nearly blasted me off your list of subscribers—back when you remarked that your Stinson had taken 30 years to become as obsolete as my Big-Board II had become in three. I never owned a Stinson, but I used to park my 7AC in a hangar next to one, so your remark hit me with a zonk of 10 on the Richter scale. However, many months went by before I needed to renew *Micro C*, and time is a notorious healer. I'm still here.

I won't attempt to refute your remark. But I'll soften the zonk ("Obsolescence, where is thy sting?") by recalling the following:

1. Mr. Alan Turing proved a theorem which says that what one computer can

do, *any* computer can do. Of course, any theorem is based on assumptions; and unfortunately no theorem can make my BBII do color graphics. Still, for composing this letter, my BBII is just as good as the latest 32-bit screamer. A "Turing theorem" for aircraft, on the other hand, would have little practical value. There simply aren't many tasks at which a B2 bomber and a Stinson are equally effective.

2. Unless your Stinson spent the 30 years in a museum, I'll wager the fabric and most of the engine aren't original. Someone has shelled out hard cash just to keep that Stinson from becoming *worse* than obsolete. My BBII has consumed negligible cash.

3. If 3 BB years equal 30 Stinson years, then 10 BB years is a century. That spells "antique" where I come from. So if I hang on a bit longer, I can contest the high ground now held by the Imsai.

Of course you'd have to dezonk your Stinson analogies. To atone for past excesses, you might even start a semi-regular old folk's department. "The Slow Lane," or "80s in the 90s," or "Is There Life After Obsolescence," or "The Trailing Edge." Something like that.

I guess I'm not your only reader who wants to know where to load up on 8" disks before antique dealers drive the price up. Maybe there are others who believe that good software won't appear until the hotshots have moved on.

R. W. Hartung
408 Orchard
East Lansing, MI 48823

Editor's Hand-Typed Response: It sounds like you're really up in the air over my analogy, but I won't let that bring me down. Would you accept a takeoff on flying in place of an apology?



See the difference at a glance . . .

textfile.old	textfile.new
hello world good bye	hello good boy

with



- Side-by-side scrollable presentations
- Compare individual text files or entire directories
- Menu-driven interface
- Built-in editor allows file change while viewing the comparison
- Target files can be chosen from directory comparisons
- Ideal for programmers

Requires an IBM compatible with at least 384K. A hard disk is recommended. DELTA runs under DOS 2.0 or higher.

Order Now. \$79.
VISA or MasterCard accepted.

Also from OPENetwork . . .

The Berkeley Utilities

Enhance your DOS environment with this set of powerful UNIX tools

TextLib

A screen & keyboard management library for C developers

Write or call our BBS for information

OPENetwork
POWER TOOLS FOR POWER USERS

215 Berkeley Place, Brooklyn, New York 11217
Voice: 718-638-2240 BBS: 718-638-2239
1-800-542-0938

Reader Service Number 201

Build A TTL To Composite Video Adapter

Have some old composite B&W monitors you'd like to use? Here's a simple project that'll give many of them new life.

When you built your clone, did you wish that you could find a use for your old composite video monitor? I had a TTL monitor for my clone system, but that left me with nothing to do with my old composite monitor. I wanted to add the monitor to a second, 80188-based clone that I had designed, but it just wouldn't operate at the IBM frequencies. It wouldn't, that is, until I built the circuit described here.

Background

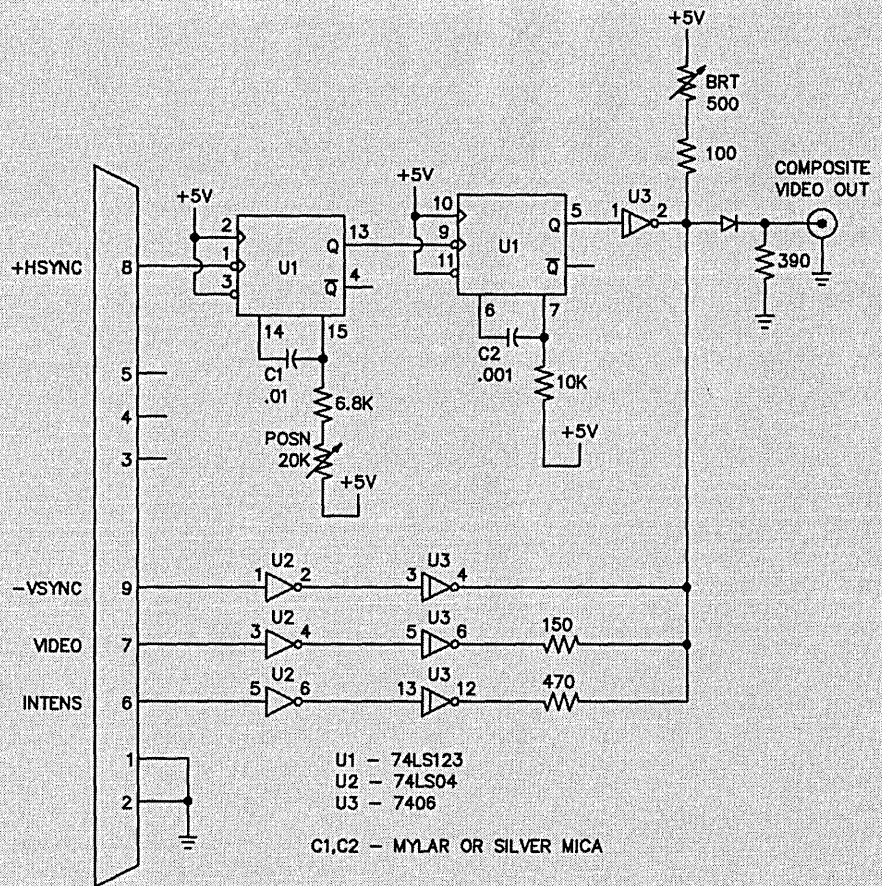
Most CP/M systems use a monitor that operates at a horizontal sweep frequency of 15,750 Hz and a vertical sweep frequency of 60 Hz. This, conveniently, is the same as standard American television. For a standard monitor, the sync, blanking, and video all transmit on a single coaxial cable.

The IBM PC and clones generate a horizontal frequency of 18 KHz and a vertical frequency of 50 Hz on a 9-pin "D" type connector. These frequencies allow more lines and better resolution, but they're a problem for standard monitors.

A circuit to convert the TTL-level signals from a PC or clone is easy. Unfortunately, most composite monitors cannot display the entire picture; they lose the first few characters. The problem is quite simple: the monitors internally generate a horizontal sync pulse that effectively blanks the screen for a few microseconds after the external horizontal sync pulse ends.

The PC begins sending the first characters almost immediately after the sync pulse. The monitor ignores these characters, and the result is a partial display.

Figure 1—TTL to Composite Video Adapter Schematic



So how do you fix this problem? First, let's delve a bit deeper into how the video signal works. A composite video signal consists of three basic components: video, vertical sync, and horizontal sync.

The video signal is made up of the "dots" that the screen displays. The dark areas of the screen are at the *blanking* level. The dots you see are at the *white* level.

The vertical sync occurs once per frame and synchronizes the vertical sweeping of the screen. The horizontal sweep occurs once per line and synchronizes the horizontal sweeping of the screen. We're interested in the horizontal sync.

On each horizontal line, the PC generates a horizontal sync pulse that's a few microseconds wide. As mentioned, the video information begins immediately after that. The composite monitor cannot display anything for a few microseconds.

If we could delay the video until the monitor had recovered from the sync pulse, everything would be all right. The

A circuit to convert the TTL-level signals from a PC or clone is easy.

first character on the line would appear at the left side, and everything else on the line would shift to the right by the same amount. Unfortunately, this approach uses a lot of parts to delay the video signal.

A second approach would be to anticipate the horizontal sync pulse and generate a new one that would occur a few microseconds early. This would have the same effect as delaying the video in-

formation. This circuit works that way, sort of.

I say sort of because I haven't yet discovered the psychic logic that anticipates a pulse. Instead, this circuit takes into account that all the horizontal sync pulses are identical.

Instead of anticipating a sync pulse and substituting an earlier one, this circuit delays the horizontal sync pulse until it's time to start the next line. Each horizontal sync is delayed by almost-but-not-quite a full line. The horizontal sweep frequency generated by the PC is about 55 microseconds.

If the horizontal sync pulse for any given line is delayed by, say, 53 microseconds, and then used to generate the sync pulse for the next line, the second line will apparently shift right by 2 microseconds.

About The Circuit

The circuit (see the schematic in Figure 1) is straightforward. It uses only three common TTL ICs. The DB9P connector at the left connects to the circuit board with a short cable and mates with the DB9S connector on your video board. The composite video connector at the right is a standard phono jack, which connects to the composite monitor with a short piece of coaxial cable.

The 500 ohm potentiometer adjusts the brightness. The 20K potentiometer adjusts the horizontal position of the picture by varying the amount of horizontal sync delay. The circuit generates standard video levels of 0 V for sync, 0.5 V for black, and about 2 V for white level.

Building The Circuit

I constructed the prototype on a small piece of perfboard. The potentiometers are mounted for easy access with a screwdriver. I stole 5 V from my power supply, which is open frame. You could use an external supply, as long as it

Figure 2—Parts List for TTL to Composite Video Adapter

Quan	Description	Comments
1	74LS123	U1
1	74LS04	U2
1	7406	U3
1	.01 uf cap	Use mylar or silvered mica
1	.001 uf cap	Use mylar or silvered mica
1	20k, 10 turn pot,	Position adjust
1	500 ohm, 10 turn pot	Brightness adjust
1	100 ohm, 1/4 w resistor	
1	150 ohm, 1/4 w resistor	
1	390 ohm, 1/4 w resistor	
1	470 ohm, 1/4 w resistor	
1	6.8 k 1/4 w resistor	
1	1N914 diode	
3	.01 to .1 uf bypass capacitors	
1	DB9P connector	
1	Phono jack composite output	

Misc: Perfboard, IC sockets, wire

♦ ♦ ♦

shared a ground with the circuit board and the computer.

Be sure to use .01 to .1 µF capacitors to bypass the circuit. *Don't* use ceramic capacitors for C1 and C2. If you do, the picture will probably have the jitters and will drift. The diode shown in the schematic is a 1N914, but almost any fast switching diode will work.

Once you've built and connected the circuit, adjustment is easy. First, adjust the 500 ohm pot about halfway between both ends. Then adjust the 20K pot to get all the characters on the left side of the screen. If you set the value of the 20K pot too large, it will not receive sync pulses, and the picture will be scrambled.

After adjusting the horizontal position, set the brightness using the 500 ohm pot. If you can't get the brightness adjusted to your satisfaction, or if the picture tends to "tear" at the top, increase the 150 ohm resistor at U3 pin 6 to 220 ohms, and the 470 ohm resistor to 680 ohms.

Adjusting The Monitor

You'll need to adjust most monitors before you can use them. They usually

have internal oscillators that run at the horizontal and vertical frequencies of 15 KHz and 60 Hz, and they must be adjusted.

The easiest way to adjust the monitor is to build the circuit, but leave U1 out of its socket. Jumper U1 pin 1 to U1 pin 5. This will pass the horizontal sync pulse through to the monitor. Then connect the adapter to the video board on the PC and to the monitor, and set the 500 ohm pot about halfway.

The vertical adjustment on most monitors is a pot, and will go down to 50 Hz without any trouble. The horizontal adjustment may be a pot, and may go up to 18 KHz.

If the pot doesn't make it (or there isn't a pot), there's usually a coil with a ferrite core somewhere near the horizontal pot. There may be another coil that adjusts the width. Tinker with the horizontal coil and the vertical pot until you get a stable picture on the screen. Then install U1, remove the jumper, and adjust the adapter.

Cautions

First, be careful working inside the

monitor. The horizontal section of the monitor generates several thousand volts for the anode on the CRT. The CRT acts as a large capacitor and it can hold a charge for days.

(Editor's note: You'll see a well insulated wire running from a transformer on the board to an insulated cap on the side of the CRT. That's the wire carrying thousands of volts to the anode, and I'd be very careful around that entire part of the circuit. Use an insulated tool when adjusting the core of the horizontal oscillator, and keep your hands out of there (even when the monitor's turned off).

Second, keep in mind that some monitors just cannot display the entire picture of a PC. Most can, but some will lose a character or two off one end or the other no matter what you do.

Finally, a (very) few monitors will overheat if you run them at 18 KHz. After adjusting the adapter, let the monitor run for a few minutes, then turn it off and see if anything is excessively hot. If it hurts to touch it, it's probably too hot.

◆ ◆ ◆

Program **FASTER** with ...

CC-RIDER

THE C PROGRAMMER'S COMPANION

NEW!

PROFESSIONAL EDITION



CROSS-REF Editing! Function Prototypes! QuickHelp, too!

Instantly recall your C symbol definitions by hitting a *hot-key* inside your editor! Along with standard CCRIDER features like Source View, Edit and Definition Paste, the new *Professional Edition* has a Symbol Browse mode and even lets you walk through all *usages* of a symbol. It's *on-line cross-referencing as you edit!*

Build a database of useful information for your C code with CCSYM, a powerful *source code analyzer* which can generate fully commented function prototype #include files for your program's static and global function definitions. Imagine! - No more manual maintenance of function declarations - and you *know* they are consistent!

CCSYM can also create a *help database for your program* which is compatible with Microsoft's help system, *QuickHelp*. All symbols in your application are accessible from QuickHelp's menus, together with *documentation extracted automatically from your original source code*.

Reader Service Number 169

Western Wares

Standard	Professional
\$89	\$279

For DOS and OS/2

(303) 327-4898

Box C, Norwood, CO 81423

FREE DEMO DISK

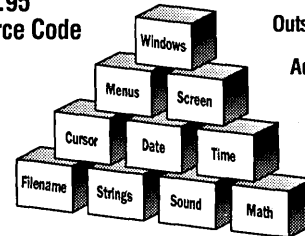
ATTENTION ASSEMBLY LANGUAGE PROGRAMMERS

BUILDING BLOCKS FOR THE ASSEMBLY LANGUAGE PROGRAMMER

QUANTASM POWER LIB

Price \$99.95
With Source Code
\$299.95

S/H \$5.00
Outside U.S. \$15.00
CA Residence
Add 7% Sales Tax
MC and VISA



- ▶ Over 256 Assembly Language Routines
- ▶ No Need to Re-Invent the Wheel Every Time You Write a New Application
- ▶ Permits Faster Application Development
- ▶ Overlapping Windowing System in Less Than 3K
- ▶ Fast and Compact
- ▶ Ideal For Programming TSR's
- ▶ C and Pascal Interfaces Available
- ▶ MASM 5.X Compatible

QUANTASM CORPORATION

19855 Srevens Creek Blvd, Suite 154
Cupertino, CA 95014
(408) 244-6826

ALSO AVAILABLE:
ASMFLOW \$99.95
An Assembly Language
Flow Charting and Source
Code Analysis Tool

Reader Service Number 139

PLAN A SOG VACATION FOR 1990!

Announcing SOGEast '90

The Eastest SOG of them all returns for an all new show on August 17 and 18, 1990, in York, Pennsylvania.

This SOG starts with a picnic on the evening of August 16, around 4 or 5 o'clock. The picnic includes delicious fare fixed on site for your dining pleasure. There will also be frisbees, softball, volleyball, and more. Come in early and spend time meeting fellow SOGees.

The 17th and 18th (Friday and Saturday) will be chock full of seminars, ranging from Hardware Control Using C++, to DOS Internals, ONLine Secrets for Modern Lovers, and More. Friday afternoon will sport a live panel discussing (calmly, I'm sure) favorite programming editors, including QEdit, MultiEdit, Brief, and others. If you want to stand up for your favorite, let us know so we can save you a chair.

Friday evening is the First Annual SOGEast Dinner Party. This is a sit-down, stand-up, have a ball Ball. Don't change your clothes from the SOG sessions, just come for the good time. A special guest speaker will be introduced....

Friday night the infamous Jolt SIG (with real Jolt this year!) will convene for a night of fun and merriment. A special hardware project especially for software people is scheduled, as well as a contest for Programmers vying for the title of "The One Day Wizard of SOGEast." We'll even be awarding prizes! If you are interested in the project or the contest, let us know.

We are looking for speakers on a large range of topics. If you'd like to share some knowledge, pass along some secrets, or amuse us for an hour, please call soon.

Current suggested retail prices are as follows:

Advance SOGEast '90 Registration (before June 1)	\$25
Advance SOGEast '90 Registration (June 1 - July 15)	\$35
SOGEast At The Door (Anything after July 15)	\$50
Picnic	\$10
Official SOGEast '90 Tee shirts	\$10
Friday Evening Dinner	\$15
Tables for Exhibitions	\$25

(All prices are Per Person)

There is more.... For the latest information, contact:

John Ribar
The CDS Group
P.O. Box 7549
York, PA 17404
(717) 792-5108 Evenings
Compuserve 73577,1652
bix jribar

We're proud to announce that SOGEast '91 is already in planning. This will be an annual event. This year, we plan to hold the SOG at one of two local colleges. If you have suggestions for next year, pass them on.

Longhorn SOG '90

Anyone who has attended knows that a SOG is like a family reunion (without the aunts and uncles). Don and I feel it would be sad not to have Longhorn SOG '90 just because Micro Cornucopia will be a fond memory. So come to Denton, Texas, June 29-30 (Friday-Saturday), for two exciting, informative, fun-filled days.

The invitation has just one condition—please holler at us by May 5 so we will have a head count. We're looking for at least 50 attendees not counting kids (kids are welcome), otherwise we will assume there's not enough interest. I've reserved conference rooms and several of you are already giving presentations. Be sure to call soon if you'd also like to speak.

We'll kick things off with a Howdy Party Thursday evening. On Friday and Saturday, we'll hold round table discussions and Jolt SIGs.

Denton doesn't have beautiful scenery or white water rafting, but it is a nice college town just north of Dallas; so if you're interested in continuing the SOG tradition, please contact—

Kim Jindra
Information Modes
P.O. Drawer F
Denton, TX 76202
(817) 387-3339

Robo-SOG '90

Set aside August 9-12 to visit the Seattle area and be a part of our first SOG.

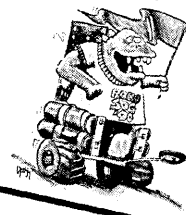
Sponsored by the Seattle Robotics Society, Robo-SOG '90 will offer much of what you expect from a SOG: quality talks (let us know if you'd like to speak), barbeque, banquet, fresh air, and, of course, the Jolt SIGs.

We'll also have maze-running robots, camping (for those so reclined), and a Junque de Jour tour. The robotics maze-run is open to all (robots). If you would like more information on preparing your metal monster to do battle, drop us a postcard.

Also, this SOG will feature microcontrollers and data loggers. If you're into embedded systems (or you've embedded your best friend), be sure to attend, or even better, speak. (Woof!)

We'll be sending out complete details on prices, location, accommodations, and schedules after March 15. We'll also have maze specifications and contest guidelines at that time.

Mike Thyng
11036 40th N.E.
Seattle, WA 98125
Voice: (206) 362-5373
BBS: (206) 362-5267



Micro C Garage Sale

Limited time only!

Special closeout!

While Quantities Last!
No rainchecks!

Genuine Antiques!

No Cliches!

All Sales Final!



1	Visual 50 Terminal (VT-52 compatible) like new	\$60
130	Boxes of new 8" disks, 10 disks per box, SSSD	\$5/bx
1	Commodore 128 w/1571 floppy drive, used 4 hrs.	\$100
10	8 1/2" reels of 1/2" mag tape, unopened	\$10 ea
12	8" SS SD/DD Floppy Drives, Mostly Shugart 801s	\$10 ea
10	Cabinets, new, can hold 2 8" drives, includes PS	\$10 ea
1	Complete Cromemco S-100 sys. w/boot disks, manuals	\$75
1	Kaypro II-83, very used, working	\$100
1	Kaypro 4-84, very used, working	\$100
1	Big Board I with cabinet, two 8" drives & PS	\$75
1	Big Board I in keyboard case with PS	\$50
70	Baby Blue Z80 cards, run in 4.77 MHz XTs, w/utilities	\$10
6	Constant voltage transformers, Sola, 250 V.A. 110V	\$35 ea
900	2764s, Intel 200 ns, new (in tubes of 14)	\$21/tube

Call for other goodies: For instance we have back issues of Life-lines (Lifelines?) and Computer Journal (CP/M anyone?). Plus, we have **numerous** computer books.

All items sold as is. No refunds. All worked the last time we used them (even the books). Call for availability as well as shipping and handling charges. Shipping will be significant on some items so if you plan to be in the Bend area, definitely give us a shout. (Shipping and handling are \$7.50 (in the U.S.) for the Baby Blue Z80 cards and \$3.00 per order on the ROMs.)

To Order, Call Today, 503-382-5060

(9 a.m. to 1 p.m. Pacific Time)

puter magazines (or how far they'd moved away from us). What's important about *CL's* subscription guarantee isn't that the magazine might replace *Micro C*, but that you can, without risk, try it. You can see if it, too, doesn't become a valuable resource.

I must also mention a couple of other magazines:

Programmers Journal (503-747-0800) is a smaller, less formal, version of *CL*. It wouldn't surprise me to see *PJ* really grow, both editorially and in circulation over the next few years.

Midnight Engineering (303-225-0856) could become another *Micro C*. Bill Gates is aiming *ME* at the folks who are working graveyard "On Their Own." Several *Micro C* regulars had articles in his first issue.

Meanwhile, if you want *CL* to look more like *Micro C*, you'll just have to write articles for them. Contact Michelle Williams at (415) 397-1881 for writers' guidelines.

We've already given the articles we've been working on to J.D. at *Computer Language* and referred several embedded authors to Tyler Sperry (a very old friend) at *Embedded Systems*.

Bruce Eckel's already writing for J.D. Karl Lunt, Bob Nansel, John Ribar, and Michael Hunt have already connected with Tyler. Who knows, you might even see the Fogg in one or both.

The Office Will Be Open

Starting April 2, we'll have a skeleton crew (Nancy) manning (womanning?) the office from 9 a.m. to 1 p.m. Pacific Time. Nancy'll keep the office open at least through April, probably through May. Sandy and I will no doubt be hanging around the office, too.

We'll still take orders for back issues, XT and Kaypro schematics, as well as issue disks. Speaking of back issues, we've hired a professional indexer to thrash through issues 1 through 52. It'll be on disk (on paper, too, we hope). We'll send you the index for only \$6. Or you get the index disk free with an order for 10 or more back issues. It's best to order soon; we're getting low on quite a few issues.

If you're into bargains (I'm talking historical treasures here), check out the *Micro C* garage sale ad on this page. We have lots of new 8" SSSD disks, 10 to a box (make great bases for little league). We have 8" single-sided drives (take a rope, tie a drive on one end, tie a boat on the other....).

We have (shudder) Big Boards for sale, a terminal, constant voltage transformers for those of you with dirty power, and something very, very special. We've got about 70 Z80 plug-in cards. I was hoping to announce them at the end of the LIMBO series (but we ended before the series did). You could use them to develop ROMs for the 64180 (or run any other CP/M software on your PC/XT/clone). One problem, they don't seem to work on any system running faster than 4.77 MHz. However, they're only \$10 each (including software) plus \$7.50 each, shipping and handling.

It's best to order by phone (503-382-5060) so you can be sure we've got what you want. (Or that you want what we've got.)

AROUND THE BEND

Where Am I Headed?

Nowhere.

I want to have some time that isn't already spoken for. I've spent my whole life in student mode. You know, if I'm awake and not working, then I should be.

I can't think of anything more enticing right now than spending two weeks or two months on a sailboat anchored off a deserted island. By the end of two months, I'd be totally stir crazy. I hope.

Finally, I've been feeling at odds with everything lately. I've been tense, uncomfortable, absent-minded, jumpy, irritable—all reminding me that Micro C and I are very close. But this issue's edit—struggling line by line through every article—has been nearly impossible. Each evening I take home the articles which need my touch. Each morning they return untouched.

My heart's not in it.

Sorry.

My Secret Wish

People who've called after hearing rumors of the end have asked how I'm feeling ("sad" is the best I've come up with) and what I'm doing next (nothing). No one has asked if something has been left out—if there's been something I've hoped would happen, but hasn't.

Yes, there is something. For most of these nine years, I've secretly wished that just once, for a day or an hour, every subscriber, every newsstand reader, every library browser who regularly thumbs through *Micro C*, every one of you who has followed and shared this journey—could get together. All of us in one place at one time. Twenty-five thousand incredible people.

That was my wish.

Solid Expectations

As of January 1, Miniscribe Corporation is under court protection from its creditors. The company threw in the brick after a year of shipping half-baked clay blocks from warehouse to warehouse, pretending they were hard drives.

I'm sure the bankruptcy surprised folks from the building trades; after all, the demand for good, new bricks has never been higher. However, these small red clay thingies aren't commonly used to hold data.

That may change shortly. We've received late word (very late word, actually) that Taiwanese computer manufacturers are scrambling to retool their 256K production lines. Apparently there's a rumor running rampant around the island that bricks are the latest in solid storage technology.

Short Note

The biggest Taiwanese exporter is IBM. (Wait until they hear about bricks.)

More Bricks

I was talking to Allyn Franklin, my favorite hard drive person (who knows, he may also repair bricks), about his "On Your Own" article when he offered me a few tips on dealing with a recalcitrant hard drive.

NEW

Develop Powerful Image-based Applications Quickly and Easily

Introducing VICTOR, video capture and image processing library

Create image-based applications that display live video on the VGA and offer powerful image processing features

With Victor your applications can show live video on any portion of the VGA screen and combine live video with text and graphics. Use sophisticated image processing tools, create lineart, TIFF/PCX files, more. Victor handles all the low level functions so you can concentrate on your application.

Rapidly test and prototype

Victor comes with ZIP Image Processing software to get you up and running quickly -- test and prototype functions before you write any code.

Video frame grabber support

Victor supports EGA and VGA up to 800 x 600, and video digitizers by: IDEC, CCI, Micromint, DAK, HRT, Electrim, Willow, VG, and others.

Victor supports Microsoft C, QuickC, and Turbo C, includes demonstration and prototyping software, full documentation, and source code for device control routines... all for only \$195. Victor with video frame grabber \$349

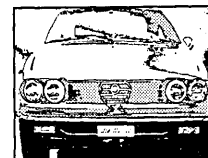
Call (314) 962-7833 to order
VISA/MC/COD

Catenary Systems 470 Bellevue St Louis MO 63119 (314) 962-7833

Reader Service Number 217



Image-based applications can be developed in QuickC and Turbo C environments.



Your applications will have powerful image processing and handling capabilities.

Now ON NEW PLATFORMS

VAX/VMS
HP 9000/300
XENIX/386
Motorola UNIX V.3
Sun-3
NCR Tower

OREGON C++

- A true compiler - not a translator
- Three compilers in one: C++, ANSI C or K&R C
- Source-level debugger
- Conforms to AT&T V2.0 C++

Motorola UNIX V.3
VAX/VMS & ULTRIX
Sun-3, 386i
XENIX/386

PASCAL-2

- Full ISO Level 1 standard
- Comprehensive error checking
- Access to command-line arguments
- Extensions include full string type, separate compilations, external variables & more



XENIX/386
Sun-3
VAX/VMS

OREGON MODULA-2

- Conforms to emerging ISO standard
- Excellent support of coroutine and interrupts
- Ideal for embedded systems

680X0 CROSS-COMPILERS ALSO AVAILABLE

CALL 1-800-874-8501

(503) 245-2202 FAX: (503) 245-8449

OREGON SOFTWARE

6915 SW Macadam, Suite 200, Portland, OR 97219 USA

The following are trademarks: Oregon Software, Oregon C++, Oregon Modula-2, Pascal-2, Oregon Software, Inc., Xenix, Sun-3, 386i, VAX, VMS, Motorola, UNIX, AT&T, ULTRIX, NCR Tower, HP 9000

Reader Service Number 193

MICRO CORNUCOPIA, #53, May, 1990 75

BIOS SOURCE CODE

The AT BiosKit gives you a complete Bios with source code you can modify for your own applications! The BiosKit includes a Bios on diskette ready for programming an Eprom, and includes the utilities you need to Rom the source code. The Bios also has a Rom Monitor/Debug and Setup. At last you have control over the core of your system. Over 380 pages, with diskette, \$199. The XT BiosKit is only \$99, or get both for \$279. The Intel Wildcard Supplement for the XT BiosKit is \$49.

FREE We'll include a free copy of the pocket-sized **XT-AT Handbook** by **Cholsser and Foster** with each BiosKit if you mention this ad when you order. Of course, this \$9.95 value is also available by itself. Or buy five or more for only \$5.00 each.



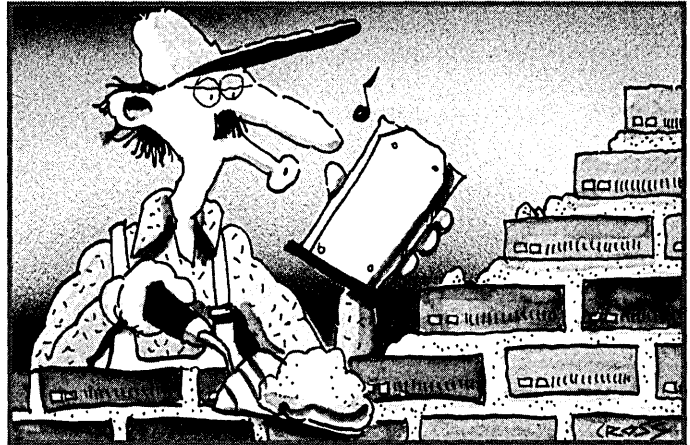
800-462-1042
In California 619-271-9526

Annabooks

12145 Alta Carmel Ct Suite 250-262
San Diego, California 92128

Money-back guarantee

Reader Service Number 160



"If something seems weird, don't turn the drive off. You may have a damaged boot sector. Just back up everything you hold near and dear.

"Often, however, you don't know there's been a problem until you power up the next day. And, what often happens is the system doesn't realize it has a hard drive. That's when I run FDISK. Try all the versions of FDISK you've got. FDISK might see the drive." (I just tried it. It works!)

"If you run FDISK with the same parameters as you did during the original format, it won't eat any data. Normally you have to do a high level format after FDISK, but in this case you want to avoid that. With any luck, after you've run FDISK, the system will again see your drive."

"If FDISK doesn't help, try reformatting just track 0. Mace's FORMAT-H is the tool I use a lot for this.

"You can reach Western Digital at (800) 832-4778. Key in the type of drive and WD controller number and the system will tell you how to jumper the card, and will give you head and cylinder counts for the drive.

"There's an even longer list inside the newest Speedstor, and Tandon has recently come out with DRIVES.COM. DRIVES.COM queries your AT and your hard drive, telling you which drive number to enter, etc.

"Both Mace and Norton have books out on data recovery, but neither goes far enough. They are really aiming at the average user."

Drives That Should Have Been Mortared

My brother grabbed me the other day, desperately looking for a letter we'd run. He wanted the issue and page number of the bit about 15 Seagate 238s that died. He wanted the copy for a customer who wouldn't accept an RLL drive (and didn't much care for MFM either).

Don had asked the guy if he was running 238s and 225s. Turned out he was. And they'd failed.

If I were buying a hard drive right now, I'd get: A Seagate full height like the 8096, 8048 or 8038; a Miniscribe full-height; or, even better, a Mitsubishi. The Mitsubishi I'm thinking of can run either 40 meg MFM or 60 meg RLL. If I were concerned about speed and space, I'd run the Mitsu RLL. On the other hand, there's nothing safer than an RLL certified platter plodding along with MFM data.

"It's Glorious!"

"No color PCompatible is complete without a copy of this; get one and see what I mean." - Jerry Pournelle, *BYTE*

"Infinite... Neither of us (the chief engineer of the Galileo Jupiter probe and I) may ever be heard from again." - Arthur C. Clarke

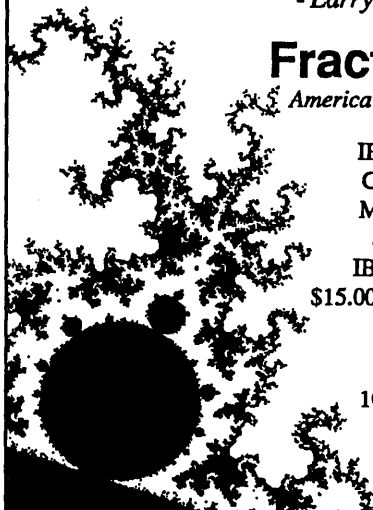
"I could watch an animated image for hours." - Larry Fogg, *Micro C (MIA 89)*

FractalMagic 5.0


America's Premier Fractal Program

IBM PC, PS/2 (3.5" or 5.25")
CGA, EGA, & VGA: \$35.00
Macintosh: Plus, 512KE, SE,
SE/30, and Mac II's: \$35.00
IBM Expansion Modules 1-4:
\$15.00 each, or all four for \$45.00

Sintar Software
"Software for the Mind"
1001 4th Avenue, Suite 3200
Seattle, WA 98154
(206) 625 - 1213
Visa - MC - AmEx



Final Issue! Stock Up on Back Issues Now!

- ISSUE #1 (8/81)**
Power Supply
1/2 PFM.PRN
16 pages
- ISSUE #2 (10/81)**
Parallel Print Driver
Drive Motor Control
16 pages
- ISSUE #3 (12/81)**
Configuring Modem 7
Reverse Video Cursor
FORTHwords Begins
16 pages
- ISSUE #4 (2/82)**
Keyboard Translation
Modems, Lync, and S10s
Undoing CP/M ERASE
20 pages
- ISSUE #5 (4/82)**
Two Text Editors
Double Density Review
20 pages
- ISSUE #6 (6/82)**
BBI EPROM Programmer
Customize Your Chars
Double Density Update
24 pages
- ISSUE #7 (8/82)**
6 Reviews Of C
Adding 6K Of RAM
On Your Own Begins
24 pages
- ISSUE #8 (10/82)**
SOLD OUT
- ISSUE #9 (12/82)**
BBI EPROM Program
Relocating Your CP/M
Serial Print Driver
Big Board I Fixes
32 pages
- ISSUE #10 (2/83)**
ISSUE #11 (4/83)
SOLD OUT
- ISSUE #12 (6/83)**
Bringing Up BBI
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
Cheap Fast Modem
BBI Printer Interface
44 pages
- ISSUE #14 (10/83)**
BBI Installation
The Perfect Terminal
BBI Video Size
Video Jitter Fix
48 pages
- ISSUE #15 (12/83)**
Screen Dump Listing
Fixing Serial Ports
Playing Adventure
48 pages
- ISSUE #16 (2/84)**
Xerox 820 Column Restarts
BBI Double Density
BBI 5"/8" Interface Fix
Recovering Text From Memory
52 pages
- ISSUE #17 (4/84)**
Voice Synthesizer
68000-Based System Review
56 pages
- ISSUE #18 (6/84)**
Kaypro EPROM Programmer
I/O Byte: A Primer
Serial To Parallel Interface
Business COBOL
60 pages
- ISSUE #19 (8/84)**
Adding Winchester To BBI
6 MHz On The BBI
Bulletin Boards
Track Buffering On Slicer
4 MHz For The 820-1
64 pages
- ISSUE #20 (10/84)**
HSC 68000 Co-Processor
DynaDisk For The BBI
Serial Printer On BBI Sans S10
Cheap & Dirty Talker For Kaypro
72 pages
- ISSUE #21 (12/84)**
Analog To Digital Interface
Installing Turbo Pascal
Low Intensity BBI Video
80 pages
- ISSUE #22 (2/85)**
Xerox 820-II To A Kaypro-8
Sound Generator For the
STD Bus
Reviews Of 256K
RAM Expansion
88 pages
- ISSUE #23 (4/85)**
Automatic Disk Relogging
Interrupt Drive Serial Printer
Smart Video Controller
Review: The 32032 Modula II
RAM Disk
86 pages
- ISSUE #24 (6/85)**
C'ing Into Turbo Pascal
8" Drives On The Kaypro
68000 Versus 80x86
Soldering: The First Steps
88 pages
- ISSUE #25 (8/85)**
Why I Wrote A Debugger
The 32-Bit Super Chips
Program 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)**
Rescuing Lost Text From
Memory
Introduction To Modula-2
Inside The PC
104 pages
- ISSUE #29 (4/86)**
Speeding Up Your XT
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
PC-DOS For Non-Clones
104 pages
- ISSUE #31 (8/86)**
RAM Resident PC Speedup
Practical Programming In
Modula-2
Game Theory In PROLOG, 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)**
ISSUE #34 (2/87)
ISSUE #35 (4/87)
ISSUE #36 (6/87)
ISSUE #37 (8/87)
SOLD OUT
- ISSUE #38 (11/87)**
Parallel Processing
Laser Printers, Typesetters
Build A Graphics Scanner
For \$6, Part 2
Writing A Resident Program
Extractor In C
96 pages
- ISSUE #39 (1/88)**
PC Graphics
Drwing Mandelbrot / 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 (7/88)**
Maintaining PCs
Keeping Your Hard Drives
Running
Troubleshooting PCs
XT Theory of Operation
Simulating A Bus
96 pages
- ISSUE #43 (9/88)**
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 Program-
ming**
A Taste of Smalltalk
Actor
Thinking Objectively
Building MicroCad
Peripheral Technology-
PT68K-2
96 pages
- ISSUE #45 (1/89)**
Computer Aided Design
CAD In A Consulting Business
Choosing PCB Layout Systems
Building Circuits With Your
Computer
Secrets of Optimization
Finding Bargains in the
Surplus Market
MASM 5.1
96 pages
- ISSUE #46 (3/89)**
Software Tools
The Art of Disassembly
Handling Interrupts With Any C
Hacking Sprint: Creating
Display Drivers
Greatest C Compilers
Turning A PC into An
Embedded Control System
96 pages
- ISSUE #47 (5/89)**
Robotics
The LIMBO Project
Starting A Robotics Company
How To Write and Use A
SystemProfiler
Problem Solving and Creativity
Turn Your XT Into A Controller
Writing Code For Two
Operating Systems
96 pages
- ISSUE #48 (7/89)**
**Tools For The Physically
Impaired**
The Adventure Begins
Selecting A Talking Computer
For A Blind Friend
Writing Software For The Blind
File Transfer Via Parallel Port
The LIMBO Project—Part Two
PCX Compatibility
A 68000-Based Multitasking
- Kernel
The Very Early Days of
Computing
96 pages
- ISSUE #49 (9/89)**
I/O, I/O...
Build A Computer The Easy
Way
PostScripts
Prog. Logic Controllers
Driving Stepper Motors
Writing TSR Programs
Low Cost I/O For The PC
Interfacing 16-Bit Devices
96 pages
- ISSUE #50 (11/89)**
3-D Graphics
3D Surface Generation
PC Video Frame Grabber
LIMBO, Part Three
PostScript, Part Two
UNIX For The PC
Capturing & Graphing A Voice
In Real Time: Part 1
96 pages
- ISSUE #51 (1/90)**
Embedded Systems
Embedding An XT Motherboard
Writing A Neural Network in C
LIMBO, Part 4
Getting Started in Hardware
Capture & Graph a Voice,
3D Surface Generation, Pt 2
Marketing Your Own Software
96 pages
- ISSUE #52 (3/90)**
C++ 2.0
Training A Neural Network
Debugging C Pointers Using
MEM
The AT Keyboard Interface
Filling In The Holes On Your XT
Logic Families
EPROM Programming
Controlling Runaway C With
CC-Rider
96 pages
- ♦ ♦ ♦
- 
- To Order:**

Phone: 1-503-382-5060
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.

Tired of the limitations of MS-DOS?
Thinking about switching to another OS?
Well DON'T, now there is:

PRO-CLI v4.1

CLI is an intelligent front & back end processor that enhances the MS-DOS system kernel. CLI provides many features of advanced OS's, like VMS and UNIX, to MS-DOS without any re-training.

The core of the CLI kernel is based on recursive definitions and references to Alias, Logicals, Symbols, and Lexical-Functions from the command line, batch programs, and existing application programs (without any modifications).

Powerful command programs can be quickly written for a wide range of purposes such as:

- flexible control of code & data management
- code development & code support tools
- automated file/device maintenance utilities
- creation of generic command procedures
- installations & system re-configurations

For the first time, designers, developers, and managers can coordinate software development and production efforts without the major limitations and restrictions in the MS-DOS environment.

- VMS is a trademark of D.E.C.
- UNIX is a trademark of AT&T
- MS-DOS is a trademark of MicroSoft



Alpha Systems Inc

1005 Sussex Blvd
Broomall, PA 19008

Price: \$149
S/H: \$ 6
215-543-2658

Reader Service Number 187

16 MHZ 68000

SINGLE BOARD COMPUTER

The new PT68K4 computer board has all the features of the K2 series plus extra benefits. It is available in 2 speeds - 12MHZ and a super fast 16MHZ - with increased memory capacity to a maximum of 12 MB!

- ★ 4 RS232 Ports
- ★ Floppy Disk Controller
- ★ Real-Time Clock
- ★ 7 XT Expansion Slots
- ★ 1 Memory Expansion Slot
- ★ 2 Parallel Ports
- ★ Optional Operating Systems

BASIC KIT (12MHZ) - 4 layer board, Processor, Humbug Monitor, 4K SRAM, 2 Serial Ports	\$220
K4 COMPLETE KIT (16MHZ) - with 512K of DRAM	\$545
ASSEMBLED BOARD (16MHZ) - with 1MB DRAM	\$649
PROFESSIONAL OS9 with C COMPILER	\$299
only valid if purchased with board	
SK★DOS OPERATING SYSTEM:	\$ 80

COMPLETE INFORMATION AVAILABLE UPON REQUEST

PERIPHERAL TECHNOLOGY
1710 CUMBERLAND POINT DR. #8
MARIETTA, GA 30067
404/984-0742 •FAX 404/984-8248
COD/MASTERCARD/VISA/CHECK

SK★DOS IS A TRADEMARK OF STAR-K SOFTWARE SYSTEMS
OS9 IS A TRADEMARK OF MICROWARE AND MOTOROLA

Reader Service Number 119

AROUND THE BEND

NEC Can't Defuse Printer Problem

It's 3:57 p.m. It's quiet in Carol's office—her NEC 890 Silent Writer is unusually silent.

"Dave, my printer doesn't work."

The power, which had been off for an hour, had just come back on and all the computers were doing their deep knee bends.

"Dave, my printer doesn't work."

"So give it a chance, already."

Unfortunately, it needed more than a chance. Nothing, dead, no fan, no lights, no smoke, no whimper, no nothing. I looked for a fuse. None in sight. None underneath, on top, by the plug, anywhere. I opened the back and pulled out the power supply. No fuse.

I called NEC. After being referred to three different sites, I finally reached someone who would talk to me....

"Where's the fuse on the 890 Silentwriter?"

"Before we talk about that, what are the symptoms?"

"It's dead."

"It could be something other than a fuse."

"Right, but for now I'd really like to know where you put the silly fuse."

"I'm not sure. Have you tried to reboot it?"

"Reboot it? REBOOT IT?"

Of course, what was I thinking. Of course, I'll need an operating system for a dedicated 68000. Let's see, DOS 3.1 doesn't run very well on an 8088.

Now, without a floppy drive, I'm not sure where to put the 3.1 boot disk (and I'm certainly not open to suggestions). And with a totally dead system, how will I know when the silly thing's been booted sufficiently? (Don't say it, I'd just hurt my foot.)

800 Frustrations

I can name two significant events in the financial life of Micro C. First, we got okayed as a VISA/Mastercard merchant, then we got an 800 number.

We had a dickens of a time becoming a VISA merchant. I was talking to the last bank in town when I finally got the okay. Banks seem awfully gun shy about small mail order businesses like Micro C. Once we'd been okayed by one bank, however, everyone wanted to step in. (No way.)

Working with VISA has been interesting. Wives cancel husbands' charges. Card numbers that are okay one day are not okay the next. People use other people's charge accounts. But by and large, VISA has been great.

Getting an 800 number was a lot easier but a lot more expensive. We pay a monthly service charge plus a per-minute fee during calls. The fee is often higher than the normal direct dial rate, and like standard long distance, we pay more for east coast calls than for west coast. But, an 800 number makes it possible for people to order subscriptions or disks from work, from a friend's house, or anywhere.

But I was worried that "orders" might become half-hour tech calls. What I didn't worry about was misdialled calls. I should have.

None of the 800 services tell you they assume no responsibility for wrong numbers. Initially that wasn't a problem. We received maybe two or three erroneous calls a week. Lately, however, we've gotten as many as 15 wrong numbers per day

AROUND THE BEND

(besides our usual half-dozen calls for subscriptions).

A travel club called Excellence in Exercise is advertising tours to east coasters. Their number is 800-888-TOUR. Great. That's 800-888-8687. But if people dial zero instead of the O in TOUR, they get 800-888-8087. That's Micro C.

Calling the company hasn't helped, after all they're doing nothing illegal. So don't be surprised if you dial our 800 number after April 1 and you hear it's been disconnected. You can still use our regular order number: 503-382-5060.

(But maybe we're missing a much better solution. We could just put out the word there's a new toll-free Lotus 3.0 support number—something ending with 8687.)

Who's Selling Computers?

I've been watching *The Wall Street Journal* again. IBM, Apple, and Radio Shack are all reporting that computer demand is drying up. Meanwhile, MicroSphere is scrambling to keep up with orders.

I called my brother.

"Hello, is Don there?"

"I'm sorry, he's tied up with a customer."

"But I'm his brother."

"Then he's especially tied up."

Busy is one thing, but being tied up with a customer is pretty kinky.

Later, after things unraveled a bit, I asked him what was happening in the industry.

"You know, Intel's profits are way up. It's all those 386s they're shipping. People are buying 286 and 386 systems like they're going out of style.

"Meanwhile, Radio Shack is now suffering from its reputation for incompatibility. They set up their systems so that you have to buy their drives, their memory cards, their everything.

"And, Apple has finally dropped its low-performance products. The variations on the Apple II, for instance, were cash cows for Apple. Now they're gone. Plus, the older MACs are selling just as slowly as they run.

"Even some of the big clone houses have set up their house brands to require custom memory boards. Really helps their bottom lines—for a while."

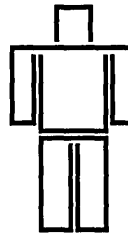
So it appears that the demise of the computer industry is really more of the decline of its figureheads.

PostScript

There's nothing more worthless than a PostScript printer when you want to output regular stuff. So you connect a silly dot matrix printer to your PostScript machine. Nothing wrong with that except dot matrix is slow, noisy, and its output isn't very clean looking.

When I heard that Legend had come up with a simple TSR package that automatically grabs anything sent to, say, LPT2: and sends it out, say, LPT1: wrapped inside PostScript, I called them up.

I haven't tried all the graphics or font options available on an Epson FX-85 or IBM graphics printer, but what I have tried has worked. Painlessly. You can run it as a TSR or as a filter. Either way it's a godsend for outputting the ordinary stuff on a super printer, super fast.



ROBOTIC SYSTEMS

LIMBO Parts Kit Series

Issue #50	<i>Mobility Base</i>	\$219
Issue #51	<i>Superstructure & Controllers</i>	\$289

Please allow 3-4 weeks for delivery. Prices subject to change without notice. International orders must be paid in advance in US funds drawn on a US bank. Visa, Mastercard, personal check, and COD accepted. Shipping & Handling, \$10. Wisconsin residents please add sales tax. Call or write for more information.

(414) 769-9070

3620 East Layton Ave, Suite 25 • Cudahy, WI 53110-1462

Reader Service Number 166

Turbo Pascal Programmers...are scraps of paper keeping you awake nights?

Tired of digging through old notes? Can't remember whether a function returns a real or an integer? Want to get procedure calls right the first time?



Then you need **MyFLIN**.

- It's a TSR. Hit the hot key to display procedure information right beside the code you're editing.
- It's easy to find information. Type part or all of a procedure or function name, and press the hot key **MyFLIN** will pop up with the nearest match(es).
- To save new procedure information, just place your cursor on the same line as the procedure, pop up **MyFLIN**, and tell it to add the procedure to the database. Requires PC/XT/AT Computer, MS-PC-DOS 2/3.xx. That's it.

Throw away the paper, get **MyFLIN** and get going.

OpalFire
Software Inc.

329 North State Street,
OREM, UTAH 84057
Phone 1-800-336-6644

Price \$30.00 PPD

Visa/Mastercard/American Express

Reader Service Number 161

MICRO CORNUCOPIA, #53, May, 1990 79



YOU WANT THE SOURCE?!

WELL NOW YOU CAN HAVE IT! The **MASTERFUL DISASSEMBLER (MD86)** will create MASM compatible source code from program files (EXE or COM). And the files are labeled and commented so they become USEABLE. MD86 is an interactive disassembler with an easy to use, word processor like interface (this is crucial for the REAL programs you want to disassemble). With its built-in help screens you won't have to constantly refer to the manual either (although there are valuable discussions on the ins and outs of disassembling which you won't want to miss).



MD86 is a professionally supported product and yet costs no more than "shareware". And of course, it's not copy protected. **VERSION 2 NOW AVAILABLE!**

MD86 V2 is ONLY \$67.50 (\$1.50 s&h) + tax

C.C. Software, 1907 Alvarado Ave., Walnut Creek, CA 94596, (415) 939-8153

Reader Service Number 31

We got you covered...

... with complete embedded system support for popular PC compilers on Intel 80x 86 and NEC V-Series microprocessors. Now you can develop an embedded application with your choice of compiler—with full support for source level debuggers and in-circuit emulators.

Paradigm LOCATE supports popular C, Pascal, BASIC & Modula-2 compilers from Microsoft, Borland and others.

- comprehensive user's manual
- compiler startup code & examples
- free technical support
- math coprocessor emulation support
- extensible MS-DOS emulator
- full Intel OMF output

Lack of an emulator got you down? Not a problem with our Turbo Debugger interface option. Now you can debug your target hardware from the comfort of your PC.

Paradigm LOCATE \$295.00
Turbo Debugger Interface \$195.00



Satisfaction Guaranteed
Orders: (800) 537-5043
Technical Support: (508) 478-0499
BIX: join paradigm
Visa/Mastercard/C.O.D. Accepted

Paradigm Systems
P.O. Box 152 Milford, Massachusetts 01757
Turbo Debugger is a registered trademark of Borland International.

Reader Service Number 113

AROUND THE BEND

PSFX \$85

Legend Communications
54 Rosedale Avenue West
Brampton, Ontario
Canada L6X 1K1
(800) 668-7077

Thanks

I can't finish without thanking a whole lot of you.

Thanks to all the folks who attended our local SOGs (especially to Andy Bakkers from Holland, who attended every one). You made those SOGs very special for me.

Thanks to all who organized, staffed, and attended the regional SOGs: including a special thanks to Dave, Randy, Mike, Karl, Bob, Kim, Don, and John—those of you who did it, and you who are crazy enough to do it again.

Also, thanks to all who subscribe. Many publications spend most of their money and time selling—going door to door to find new advertisers and mailing zillions of promotional pieces to find new subscribers. Though we haven't done any circulation promotion for 2½ years, we're very close to our peak circulation. That doesn't happen very often in this industry.

The reason we've done so well is you. You've stayed with us. You've encouraged others to subscribe. In addition, you've been willing to read the world's longest editorials year after year after year without complaint. (Okay, you caught me. But three of you haven't complained.)

I'd like to thank my mother and father, without whom I wouldn't have parents. And, I'd like to thank my father-in-law, who was one of Micro C's biggest supporters and a wonderful friend.

I'd like to thank Jennifer and Erin. (They're like family to me.) It's one thing to have a job, it's another to conceive and raise a magazine. Over the past nine years, they've played second fiddle (and second piano) to 53 editorials, 53 deadlines, and 53 hassles with the printer. Now I'm going to have some time, and they're going to have a Dad. (They may insist I start another magazine, real soon now.)

Plus there's Sandy. If everyone had a Sandy, there'd be a lot more successful businesses. I'm not sure which of us has worked harder to make Micro C go, and I'm not sure which of us is more burned out.

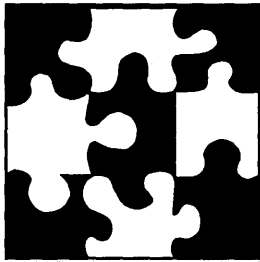
Finally, there's the staff. What can I say about this collection of miscreants who—put up with, cover for, tolerate, even occasionally get a laugh out of—what's left of me. They've produced magazines, harassed writers, filled pages with advertising, and still found time to party. A truly great group.

If You're Bummed

I'm sure many of you won't appreciate this change. I understand. It's part of the sadness I feel. If you feel strongly about it, write to me, call me, or leave a message on the BBS. I'll be here.

For a while.

David Thompson
Ex-Editor & Publisher



UNITS & MODULES

By Michael Hunt

2313 N. 20th
Boise, ID 83702
(208) 336-7413

Shortest Distance

Michael concludes his column with a look at connections: tying together network nodes, for instance, with a minimum amount of cable. An interesting problem.

Everyday we use our reasoning ability to find the shortest distance between two points. The trip to work, to the store—you know how routine it is. But how good are we at finding the shortest distance between many points? If you've ever set up a computer network, you were probably concerned with keeping cable cost down. The nodes in the network are the many points that we need to connect.

For a small network, the solution can be simple and straightforward. As the number of nodes increases or offsite locations are introduced, the problem of efficient connectivity grows quickly. We need an algorithm to find the shortest or lowest cost method of connecting all the nodes in the network.

Abstraction

You can solve the network problem by using a mathematical structure known as a graph. A graph consists of a set of points or vertices, and the lines connecting the vertices, called edges. The nodes of the network can be viewed as the points or vertices of the graph. The edges represent the connections between the nodes. You assign a cost to the edge or connection between two nodes, representing the distance between the two nodes.

The graph is a collection of edges, each with a cost, between vertices. The set of edges is called a spanning tree. The sum of the costs of the edges in the tree is the cost of the spanning tree. Our goal

is to find a minimum cost spanning tree.

Two algorithms address this problem. The first is known as Prim's algorithm. Prim's begins with one vertex and builds the spanning tree one edge at a time. It begins with a set of one vertex.

For each iteration, it finds the shortest edge to a new vertex and adds the vertex to the initial set until all vertices have been included. Prim's uses an adjacency matrix to store the edges and their costs.

The other algorithm is Kruskal's. I've chosen to implement it because it uses a linked list of the vertices and edge sets. The linked list is dynamic, not restricted by some upper array limit like the adjacency matrix.

If there are many fewer edges than the square of the number of vertices, then Kruskal's algorithm is best. But if the number of edges is about the same as the number of vertices squared, Prim's performs better.

Kruskal's begins with a graph consisting of all the vertices and none of the edges. Each vertex is in a connected component by itself. The algorithm builds a collection of connected components, each component consisting of a set of vertices and the edges that form its spanning tree.

The list of edges is sorted by cost and then examined in order of increasing cost. If the edge connects two vertices in two different components, the edge is added to the spanning tree and the two components are combined. If the edge connects two vertices in the same component, it is discarded. When all the vertices are in one component, the spanning tree is of minimal cost.

'HC705 Simulator

The simulator is complete, and by press time Karl Lunt and John Ribar will

We need an algorithm to find the shortest or lowest cost method of connecting all the nodes in the network.

be testing it. John is finishing up the back end of the compiler. Look for it in *Embedded Systems Programming*. For you Amiga fans, a friend of mine is porting the simulator to the Amiga using Benchmark Modula-2.

Last Column

Well, after nine issues, this is the last installment of the Units & Modules column. After talking with Karl Lunt at Rocky Mountain SOG, I have been bitten by the hardware bug. Recently I've been gearing up for the change. Let's see, multimeter, breadboard, power supply, 'HC705 programmers board, soldering pencil, solder, and back issues of *Micro C*.

The simulator for the MC68HC705C8 microcontroller that Karl writes so much about will be the first of many articles about microcontrollers and embedded systems.



Commenting Disassembler!

SOURCER™ 486

- SEE HOW PROGRAMS WORK
- EASILY MODIFY PROGRAMS

SOURCER™ creates detailed commented source code and listings from memory and executable files directly suitable for reassembly. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Includes a definition file facility to include your own remarks and descriptive labels, force data types, and more. Complete support for 8088 through 80486, 8087 to 80387, and V20/V30 instruction sets.

We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER. "Sourcer is the best disassembler we've ever seen." PC Magazine 1/17/89 page 101.

BIOS SOURCE

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

- CHANGE AND ADD FEATURES
- CLARIFY INTERFACES

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

ASM ProPak™

- ASSEMBLY TOOLS PACKAGE

Save \$50 with the ASM ProPak™ assembly tools package. It includes Sourcer with the BIOS Pre-Processor; Unpacker™ for unpacking packed EXE files; ASMtool™ to automatically generate flowcharts and tree diagrams from assembly source code; and View-It™ to quickly view files of any size and see up to 8 times more information on a single screen using EGA/VGA equipment.

Sourcer-Disassembler	\$ 99.95
BIOS Pre-Processor	49.95
Unpacker-Unpacks packed files	39.95
ASMtool-Source code analyzer	89.95
View-It-Zoomable file viewer	69.95
ASM ProPak-All Above (save \$50)	299.75
Sourcer w/BIOS-(save \$10)	139.95

Shipping & Handling: USA \$5; Canada/Mexico \$10; Other Countries \$15; CA Residents add sales tax. Visa/MasterCard/COD accepted.

NOT COPY PROTECTED

30-DAY MONEY-BACK GUARANTEE

If within 30 days of purchase you find our product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.

For orders or information, call:

1-800-662-8266

V COMMUNICATIONS, INC.

4320 Stevens Creek Blvd., Suite 275, Dept. MC
San Jose, CA 95129 (408) 296-4224

Reader Service Number 62

Figure 1—Kruskal's Algorithm Implementation

```
PROGRAM Kruskal;

(* Author : Michael Hunt
   This source code is released into the public domain. *)

TYPE
  keystr = STRING[5];
  edgeptr = ^edge;
  edge = record
    bptr, fptr : edgeptr;
    key1, key2 : keystr;
    cost : REAL
  end;
  compptr = ^comp;
  comp = record
    bptr, fptr : compptr;
    compname : INTEGER;
    edge : edgeptr
  end;

VAR
  I, J, K, P, compcount : INTEGER;
  maincomp : compptr;
  mainedge : edgeptr;
  fill : text;

PROCEDURE readfile; (* proc to read in data file and *)
VAR (* create initial list *)
  e : edgeptr;
BEGIN
  assign(fill, 'KRUSKAL.DAT'); (* open data file *)
  reset(fill); (* set file pointer to beginning *)
  if not eof(fill) then begin (* if not empty file *)
    new(e); (* create new pointer *)
    mainedge := e; (* set it to first edge *)
    readln(fill, e^.key1, e^.key2, e^.cost); (* read in edge data *)
    e^.bptr := mainedge; (* set bptr to list start *)
    e^.fptr := NIL (* set forward pointer to NIL *)
  end;
  while not eof(fill) do begin (* while still not end of file *)
    new(e); (* create new pointer *)
    readln(fill, e^.key1, e^.key2, e^.cost); (* read in edge data *)
    mainedge^.bptr := e; (* insert at front of list *)
    e^.fptr := mainedge;
    mainedge := e
  end;
  close(fill) (* close data file *)
END;

PROCEDURE writelst(p : edgeptr); (* writes out any edge list *)
VAR
  next : edgeptr;
  cost : REAL;
BEGIN
  cost := 0; (* set list cost to zero *)
  next := p; (* set work pointer to list start *)
  while (next <> NIL) do begin (* while not end of list *)
    writeln(next^.key1, next^.cost:5:3, (* list data *)
            ', next^.key2); (* list data *)
    cost := cost + next^.cost; (* increment cost by current edge *)
    next := next^.fptr (* inc edge pointer to next in list *)
  end;
  writeln('Cost is ', cost); (* output list cost *)
END;

PROCEDURE writecomp; (* proc to write out all components *)
VAR
  next : compptr;
BEGIN
  next := maincomp; (* start at first component *)
  WHILE (next <> NIL) do BEGIN (* while still components in list *)
    IF (next^.edge <> NIL) THEN BEGIN (* if component no empty *)
      writeln('Component ', next^.compname); (* print component name *)
      writelst(next^.edge) (* output edge list for component *)
    END;
    next := next^.fptr (* inc component to next in list *)
  END
END;

PROCEDURE sortlst; (* proc to sort initial edge list *)
```

```

VAR
tcost : REAL;
tkey1,tkey2 : keystr;
I, J, count : INTEGER;
e, cur, prev : edgeptr;

BEGIN
e := mainedge; (* start at beginning of list *)
count := 0; (* then implement bubble sort *)
WHILE e^.fptr <> NIL DO BEGIN (* bubble sort is O(n**2) so *)
count := count + 1; (* this implementation of Kruskal's *)
e := e^.fptr (* would have to have a O(nlogn) *)
END; (* sort to be O(nlogn) *)
FOR I := 1 TO count-1 DO BEGIN
cur := e;
prev := e^.bptr;
FOR J := count DOWNT0 I DO BEGIN
IF cur^.cost < prev^.cost THEN BEGIN (* swap code *)
tcost := cur^.cost;
cur^.cost := prev^.cost;
prev^.cost := tcost;
tkey1 := cur^.key1;
cur^.key1 := prev^.key1;
prev^.key1 := tkey1;
tkey2 := cur^.key2;
cur^.key2 := prev^.key2;
prev^.key2 := tkey2
END;
cur := prev;
prev := prev^.bptr
END
END
END;

PROCEDURE buildlst; (* proc to build minimal spanning *)
VAR (* tree if it exist *)
cur : edgeptr;
comp1, comp2 : compptr;

PROCEDURE find(key : keystr; (* proc to find the component that *)
VAR comp : compptr); (* a key of an edge is in *)
VAR
found : BOOLEAN;
edge : edgeptr;
BEGIN
found := FALSE; (* set found flag to false *)
comp := maincomp; (* start a first component *)
WHILE (comp <> NIL) AND (NOT found) DO (* while not empty *)
BEGIN (* component & key not found *)
edge := comp^.edge; (* start at first edge *)
WHILE (edge <> NIL) AND (NOT found) DO (* while not empty *)
BEGIN (* edge & key not found *)
IF (edge^.key1=key) OR (edge^.key2=key) THEN (*if key found*)
found := TRUE (* set flag *)
ELSE edge := edge^.fptr (* else increment to next edge *)
END (* while *);
IF NOT found THEN comp:=comp^.fptr(* if not found-next compon *)
END (* while *);
IF NOT found THEN comp:=NIL (* if still not found-no component *)
END;

PROCEDURE joincomp(VAR c1, c2 : compptr); (* proc to join compon *)
VAR
l : edgeptr;
BEGIN
l := c1^.edge; (* set temp to first edge list *)
WHILE l^.fptr <> NIL DO BEGIN (* while not last edge in list *)
l := l^.fptr (* inc to next edge in list *)
END;
l^.fptr := c2^.edge;(*last in 1st list points to 1st in 2nd list*)
c2^.edge^.bptr := l;(*1st in 2nd list points back to last in 1st*)
c2^.edge := c1^.edge; (* 2nd comp points to 1st in 1st list *)
c2^.edge^.bptr := NIL; (* clean up back pointer *)
c1^.edge := NIL (* 1st list now empty *)
END;

PROCEDURE newcomp; (* proc to create a new component *)
VAR
cur : compptr;
BEGIN
new(cur); (* create new pointer *)

```



Get organized
instantly!

Create, Find, View, & Edit
notes, addresses,
documents, data.

- * No setup.
- * No keywords to define.
- * Configurable,
Full-screen editor.
- * Turbo fast & little

WIZARD

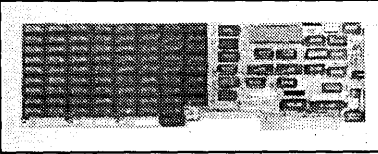
The Intelligent Information
Processor
Only \$49

ACQUIRED INTELLIGENCE
PO BOX 2091 DAVIS, CA 95617

916-753-0360

Reader Service Number 72

YOU WANT REVOLUTION NOT DISK ROTATION!



Blue Flame II Solid State Disk

- For the PC, XT, AT, 386 DOS computers, no mechanical wear
- Revolutionary way to solve your disk access bottleneck - Rebel against hard drive delays
- Like a hard disk, but it's 10 to 20 times faster
- Expandable - 1 to 8 MB per card, up to 32 MB per drive
- EMS for LeSS - LIM/EMS 3.2 & 4.0 emulation software available
- Battery Backup option, makes the Blue Flame II non-volatile, \$130
- Concurrent DOS compatible too
- 2-year warranty, parts and labor



1 MByte - \$495
2 MByte - \$695
4 MByte - \$call
8 MByte - \$call

SemiDisk Systems, Inc.
11080 S.W. Allen #400
Beaverton, Oregon 97005
(503) 626-3104 FAX (503) 643-0625

Reader Service Number 162

Micro Cornucopia

Phone Orders:

(503) 382-5060 M-F, 9 AM-5 PM PST

Mall Orders:

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

PC TIMER TOOLS

PCHRT is the definitive answer to execution profiling and embedded timer requirements in a PC/MSDOS environment. 46 functions manage three distinct classes of high resolution timer functionality:

21 functions implement multiple microsecond timers. Timers may be stopped, started, suspended, resumed, named, queried, and complete reports of all timer activity may be generated with a single function call.

16 functions manage microsecond timers that may be inserted in virtually any PC interrupt vector. Interrupt timers may have results listed by interrupt function requested, so BIOS, DOS, and EMS interrupts may be profiled effortlessly!

9 functions implement precision microsecond delays, perfect for interfacing data acquisition and process control hardware where precise timing is required.

PCHRT includes libraries for TC, TP, and MSC, example programs, manual, full source, and is \$49.95 ppd USA, elsewhere add \$4.00. VISA/MC accepted. A 30 day "No Questions Asked" money back guarantee insures your satisfaction. Call or write for our seven page brochure.

RYLE DESIGN

POB 22, Mt. Pleasant, MI 48804
517-773-0587

Reader Service Number 171

```

P := P +1; (* increment component name *)
cur^.compname := P; (* set component name *)
cur^.bptr := NIL; (* clean up back pointer *)
cur^.fptr := maincomp; (* insert new component a front of list *)
cur^.edge := NIL;
cur^.fptr^.bptr := cur; maincomp := cur
END;
PROCEDURE addedge (VAR edge : edgeptr; VAR comp : compptr);
BEGIN (* proc to move edge form intial list to component *)
  mainedge := edge^.fptr; (* set begin edge to next edge *)
  edge^.fptr := comp^.edge; (* begin insertion *)
  edge^.fptr^.bptr := edge;
  comp^.edge := edge;
  edge^.bptr := NIL (* clean up back pointer *)
END;

BEGIN (* begin of BuildList main *)
  cur := mainedge; (* set work pointer to begin edge list *)
  WHILE cur <> NIL DO BEGIN (* while still edges to process *)
    find (cur^.key1, comp1); (* find component that key is in *)
    find (cur^.key2, comp2); (* find component that key is in *)
    IF (comp1 = NIL) AND (comp2 = NIL) THEN BEGIN (*if neither key*)
      newcomp; (* found in components, create new component *)
      addedge (cur, maincomp) (* add edge to new component *)
    END;
    IF (comp1 = NIL) AND (comp2 <> NIL) THEN BEGIN (*if key only in*)
      addedge (cur, comp2) (* component 2, add edge to component 2 *)
    END;
    IF (comp1 <> NIL) AND (comp2 = NIL) THEN BEGIN (*if key only in*)
      addedge (cur, comp1) (* component 1, add edge to component 1 *)
    END;
    IF (comp1 <> NIL) AND (comp2 <> NIL) THEN
      BEGIN (* if both keys found in components *)
        IF (comp1 <> comp2) THEN
          BEGIN (* if keys not found in same component *)
            addedge (cur, comp1); (* add edge to one of the components *)
            joincomp (comp1, comp2) (* join the two components *)
          END
        ELSE (* cycle found *)
          (* discard edge *)
          mainedge := cur^.fptr
        END;
      cur := mainedge (* increment to next edge *)
    END (* endwhile *)
  END (* buildlist *)

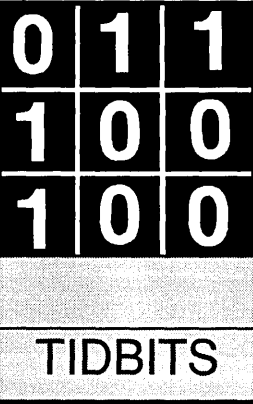
BEGIN
  P := 0; (* initialize component name *)
  maincomp := NIL; (* initialize component list *)
  readfile; (* read in edge list from data file *)
  writeln ('Input list...');
  writelist (mainedge); (* output initial edge list *)
  sortlist; (* sort the initial edge list *)
  writeln; writeln ('Sorted list...');
  writelist (mainedge); (* output sorted edge list *)
  buildlist; (* build the minimal cost spanning tree *)
  writeln; writeln ('Minimal cost spanning tree..');
  writecomp (* output the components *)
END.

Data File: kruskal.dat

A B 2.0
A C 8.0
A G 10.0
A H 14.0
B C 8.0
B F 15.0
B G 17.0
C D 12.0
C E 4.0
C F 8.0
C G 14.0
C H 14.0
D E 4.0
D F 13.0
D H 12.0
E F 10.0
F G 4.0
G H 16.0

```

♦ ♦ ♦



By Gary Entsminger

P.O.Box 2091
Davis, California 95617

Finding Text Faster, An Optimizing TSR

Development Tool, & TAPCIS

Searching for the ultimate search routine? Gary looks closely at Power Search and at searches in general.

Just when I thought I'd figured out how to search a string for a pattern (or substring) fast (see Tidbits, *Micro C*, #52), Blaise Computing comes along and blows my doors off. Their system, called Power Search, is a search compiler that finds text faster by building object code *at runtime* for each search. A better idea!

The object code, executed as a function, gets the address of a buffer to scan, the buffer length, and the pattern, then goes to work. When I added the compiler to Wizard, my search model from last issue, its speed improved considerably.

Depending on the length of a search, the search compiler improved Wizard's performance by over 20% from its second best effort (using a good lookup table) and by 120% (using unoptimized brute force).

For those of you who missed last issue, the brute force search does nothing fancier than begin at the beginning of a string and move through character by character looking for a match. Lookup table algorithms look for ways to eliminate some of the target string from the search.

Refer to Figures 1 and 2 for a comparison of four search methods:

- (1) brute force;
- (2) optimized brute force;
- (3) table lookup system (Boyer-Moore);
- (4) Blaise's search compiler.

In Figure 1, we're searching through 1

meg. In Figure 2, we're searching through 3 megs. For more details about methods 1, 2, and 3, see last issue. In the next few paragraphs I'll try to show you why a search compiler is faster.

A typical search through a buffer for the occurrence of a string might take the following approach (in C):

```
#define LEN      32767 /*32K buffer*/
#define PATLEN   20

char   Buf [LEN];
char   Pat [PATLEN];
unsigned i, j;

for (i = 0; i < LEN - PATLEN; i++)
{
    for (j = 0; j < PATLEN; j++)
    {
        if (Buf[i+j] != Pat[j])
            break; /* no match! */
    }
}
```

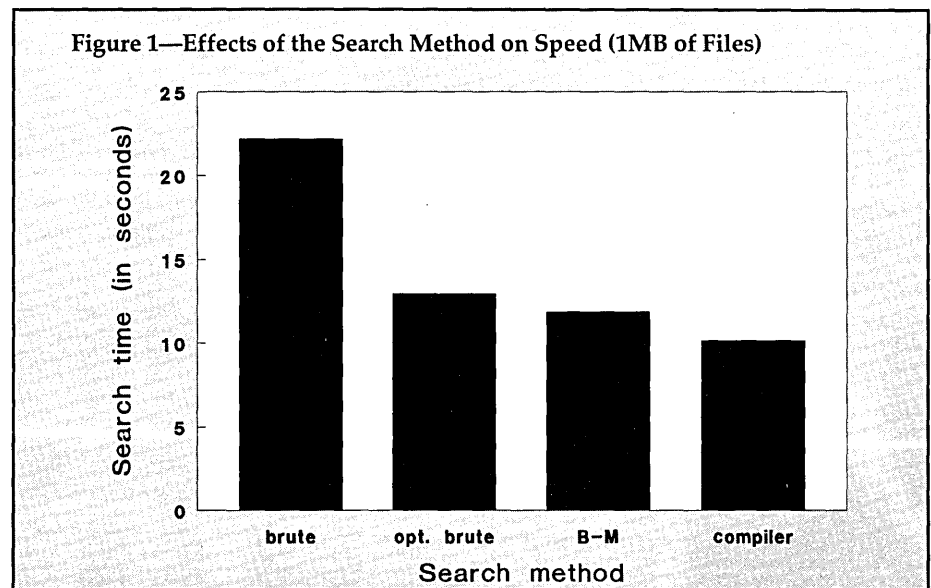
```
}
if (j == PATLEN)
    break; /* match at (i) */
}
```

Note that most of the code (and time) is used to manage the array indices. Power Search gains speed by eliminating this overhead, by (effectively) hand coding each lexical element of a pattern. This hand coding builds a production (or set of bytes that correspond to machine instructions) to scan the buffer for the pattern.

Each time a user submits a pattern for a search, the search compiler builds this object code—*at runtime* very, very fast.

To get an idea of what Power Search is up to, consider the following example (from the Power Search manual). In this example, a user submits a character that she wants matched (either upper or lower case). First, a typical method to

Figure 1—Effects of the Search Method on Speed (1MB of Files)



compare characters, hand-optimized in assembly.

The method uses the following assumptions:

- (1) DS:SI points to the next character to scan;
- (2) ES:DI points to the next pattern character;
- (3) The direction flag is cleared;
- (4) AL can be scratched.

We can't use EPE CMPSB because we want the comparison to be case-insensitive. So the code looks like this:

```

lodsb      ;(AL) = buffer char
xor al,es:[di];(ES:DI)=pattern char
inc di    ;advance pattern char
           ;do they match?
je match  ;yes; exactly.
cmp al,20h ;case problem?
jne failure ;no; they just don't
           ; match.

match:
.....    ;next production.

```

Now, the search compiler's optimized code:

```

lodsb      ;(AL)=next buffer ch;
cmp al,'X' ;upcase match?
je continue ;yes. next production
cmp al,'x' ;lowcase match?
jne failure ;no.

continue:
.....    ;next production.

```

"Both methods have to load the next character from the buffer into AL with a LODSB, a fast one-byte instruction. But the general method has to read the pattern character to compare with the buffer character, and to increment the pointer to the pattern (DI). These two steps are unnecessary in the search compiler's production because the character is known when the search begins."

In effect, the Power Search method leaves one compile until runtime. This last compile is blazingly fast—fast enough to increase search speed significantly on long searches.

You can add any of the seven functions to your C code very easily. Blaise gets you started with several examples and precompiled libraries for Microsoft and Turbo C. The manual includes a function reference and an illuminating discussion about how the search compiler works.

Although I'm impressed (can you tell yet?) with the speedy string search

speed I've discussed, I'm selling this program short. Power Search can also handle regular expressions, including negated character classes, subexpressions, and repetitions. A regular expression, for you noncompiler-writing types, describes a language. I won't go any farther into that here. For more information, check one of the classics, such as *Principles of Compiler Design* by Aho and Ullman.

In short, you can easily incorporate functions to handle regular expressions into your C applications.

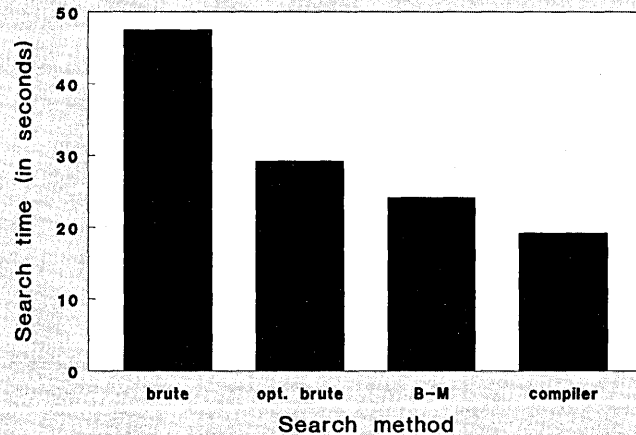
If I'm leaving you with too positive an impression about Power Search, I'll throw in one complaint—one I'll direct at many other companies as well as Blaise. The manual's index isn't bad, but it's incomplete. An index should be useful most of the time. If not, users won't use it! 'Nuff said.

The CodeRunner function library includes:

- tools for installing and (safely) uninstalling TSRs;
- support for up to 256 TSR entry points (so you can allow users to go directly to specific parts of an application);
- an event scheduler (for timing popups);
- and an ultra-compact BCD math library.

TSRs developed with CodeRunner can be very small (<2K) because CodeRunner lets you discard initialization code when it's no longer needed. You put your startup display, messages, etc., in a disposable data module and the code and data you want to remain resident in a permanent module. Then using a 4 function sequence similar to the following, you set the TSR in motion.

Figure 2—Effects of Search Method on Speed (3MB of Files)



For more information:

Power Search \$149
Blaise Computing
 2560 Ninth St., Suite 316
 Berkeley, CA 94710
 (800) 333-8087

Optimized TSRs

You C'ers searching for a concise package of TSR tools should check out CodeRunner from Microsystems Software. CodeRunner is a compact library of over 200 functions written in assembly language that you can call from your Turbo C and Microsoft C programs.

```

/*Define stack, enable TSR on ret*/
stay_resident(isr_stk,STK_SZ*2);
/* Enable init data disposal */
idata_end=init_data_end;
/* Enable init code disposal */
icode_beg=init_code_start;
/* Install hotkeys */
install_hk(hk_list,service,
          2*STK_SZ,3);

```

The CodeRunner package includes a template for creating TSRs, tips on optimizing C programs, and several useful examples which include source code. The Microsystems Software BBS (free when you register) also has additional source code and tips from CodeRunner

developers. The CodeRunner manual (here I go again) unfortunately is a little too concise. It's obviously intended for seasoned C programmers. If you're not one, expect to work a bit getting a TSR up and running.

If you're looking for a professional development tool that'll let you create compact, fast TSRs, CodeRunner might be your ticket.

It's \$149 from:

Microsystems Software
600 Worcester Rd., Suite B2
Framingham, MA 01701
(508) 626-8511

Note: Microsystems Software specializes in developing software for the disabled. If you need information of this sort, get in touch with them.

TAPCIS

It took Bruce Eckel (and others) a long time to convince me to go ONLINE nationally. I had my reasons. In particular, I hate rummaging around bulletin boards; I'm neither a shopper nor a shopper. So seven years after my first

about with a Kaypro, I joined CompuServe. It was more or less what I expected: forums, marts, conferences, weather reports, etc.

Bruce had argued that EasyPlex alone (CompuServe's electronic mail service) was worth the price of admission. My first few experiences ONLINE suggested otherwise. I figured out how to send and receive mail, but it was more trouble than it was worth, and certainly not worth the price of admission. Then I discovered TAPCIS.

TAPCIS is an access program for CompuServe that does indeed take the pain (and time) out of sending and receiving mail ONLINE. It's fast and virtually automatic.

I run the program, enter N (for New), and TAPCIS calls CompuServe, logs on, checks mail, downloads any that's waiting, and logs off while I make coffee.

To upload a file or a message, I tell TAPCIS where the message is on my hard disk, and it logs on, uploads, logs off, etc., while I drink my coffee.

TAPCIS, you're wonderful. If you use CompuServe, you can download it and play free for 21 days. If you decide to

keep using it, you register your copy for \$79 and get a manual and the latest version. You can even have TAPCIS register for you ONLINE. Now that's friendly.

For more information:
Support Group, Inc.
Lake Technology Park
P.O. Box 130
McHenry, MD 21541
CompuServe 74020,10 (Go TAPCIS)
(800) USA-GROUP

And, you can reach me on CompuServe, 71141,3006

Postscript

As the sun sets, Tonto and the Masked Man ride off on their beautiful horses to cook a beans and bread supper in the old west. With a "Hi ho Silver, away," I'm reminded of another great saying—Thanks for all the memories.

It's been good getting to know you Micro C'ers. Let's meet again someday—at a soggy get-together, a conference, or on some other pages. And that, friends, ends Tidbits.

◆ ◆ ◆

Turbo Pascal/Turbo C Users:



**Don't Miss
The Boat!**

Just knowing where to start can be a challenge. And once you have experience — where do you look for inspiration?

Since 1984, TUG has offered its members access to a worldwide network of thousands of Turbo Pascal and Turbo C users through *TUG Lines*, its lively, 48-page bimonthly journal. Tutorials and help for beginners. Tips and techniques for experienced programmers. Bug reports with workarounds. Product reviews. Volunteer consultants you can call for answers.

But TUG is much more than a newsletter! Membership options include a multiuser BBS, videos, contests, an extensive library of hand-picked and tested software, and an annual "GeTUGether" where experts gather to share their knowledge at all levels.



Turbo User Group

Post Office Box 1510 • Poulsbo, Washington 98370

(206) 779-9508

Annual dues (US funds): U.S. \$27.00; Canada \$32.00; Overseas \$39.00

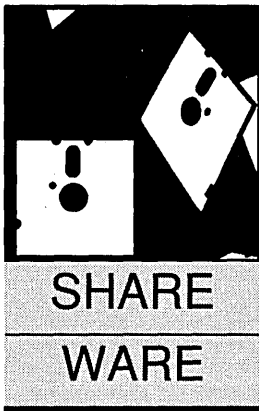
FREE ISSUE!

Micro C readers are invited to receive a FREE sample issue of *Midnight Engineering*.

The Journal Of Personal Product Development provides a unique mix of practical hardware and software engineering mixed with the business aspects of small scale product development. Your guide to developing and marketing your own products.



Midnight
Engineering
Suite 7041
111 E. Drake Rd.
Ft. Collins, CO 80525
303-491-9092 (anytime)



My Favorite Tools

Helpful Shareware

By Nelson Ford

The Public (software) Library
P.O. Box 35705
Houston, TX 77235-5705
(713) 665-7017
CIS: 71355, 470
FAX: (713) 524-6398

In this, his first shareware Micro C column, Nelson covers his favorite utilities. Great stuff.

Prior Shareware columnist Anthony Barcellos is going to be a tough act to follow. For the last two and a half years, Tony has done an admirable job of coming up with the real gems.

Over the past eight years, I've tested and reviewed thousands of public domain and shareware programs for the Public (software) Library.

I've written a dozen shareware programs and am one of the founding members of the Association of Shareware Professionals.

Terms

First, let me clarify a few terms.

When I say "pd/shareware," I include "freeware," too. Public Domain is software which the author has donated to the public domain. (The absence of a copyright notice is *not* sufficient to make a program pd.)

Freeware used to be the same as shareware. Andrew Fluegelman trademarked the name freeware, so other programmers invented the name shareware. After Fluegelman's death, the term freeware fell into generic use, meaning software that is free, but copyrighted, as opposed to free and publicly owned.

Since most of you spend your time on DOS machines, I'll start out with a discussion of...

The Ten Greatest DOS Tools In The History Of Man

I would hate to think of having to work my 14+ hours a day on a computer without the following:

(1) LIST, by Vernon Buerg. You probably already know about this file viewing utility, but you may not know that Buerg keeps making it better. If you haven't seen version 7, it's time to dial up your favorite BBS.

(2) FGREP, by Christopher J. Dunford, is the world's fastest text search. Sure, you can get a fancy TSR like Gofer, but you can't beat FGREP for speed and convenience.

(3) SD-whatever, anyone's sorted directory utility is better than what DOS provides. I prefer HotDIR, or one of the others that color-codes file names by extension. It's easy to spot all the OBJ files in a crowded directory.

(4) MOVE FILES 4.2, by Bryan Higgins, helps you move files between sub-directories or disks. After years of struggling with deficient file movers, it's great to find one that's very close to perfect. The only question is why it took so long to show up.

(5) CED, by Christopher Dunford, is to the DOS command line what a full-screen editor is to Edlin. It lets you "cursor up" previous DOS command lines and edit them using all the editing keys (e.g.: home, end, word left/right). CED is freeware, but there's a commercial version that's even better.

(6) HindSight, by Christopher Dunford. Anyone who works in DOS needs to see what's just scrolled off the screen. Though there are several video backscroll utilities, HS works with Dunford's CED, so I use it. Also check out FANSI-Console, which includes a wealth of video related features along with backscroll.

(7) LHarc, a freeware file archiver from Japan. LHarc is the best file compression I've seen. Some files will compress to a fifth their original size.

(7a) I must sneak a related cleanup utility in here. LHdel, by Duane Hendricks, examines the contents of an LHarc archive file and deletes matching files in the current directory.

(8) Newkey, by Frank Bell. If you don't have a keyboard macro program, you need Newkey.

(9) DESQview. Oops; okay, this multitasking program isn't shareware. Worse yet, I lied in (8): I don't use Newkey anymore, since DV can automatically load a different set of macros each time you open a window. Since I only use DV for task swapping, you might think that Software Carousel could do just as well. But DV has too many nice extras to give up.

(10) SitBack, by SitBack Technologies. This TSR auto-backup utility isn't shareware either, but I am the world's worst at backing up my work and there's no shareware solution. SitBack will backup preselected files in the background at specified times. This TSR uses a modest 16K.

The closest shareware equivalent is AutoSave, by Biologic Corp. It records your keystrokes to disk, plus it saves your work for you as often as you wish.

How To Write Shareware And Make A Million Dollars

Step One: Write a shareware program. Step Two: Win a lottery.

A half-dozen shareware authors have been able to combine the two. Tom Smith and his partners turned a shareware communications program, Procomm, into \$15 million a year. Jim Button (PC-File) and Bob Wallace (PC-Write) make over \$2 million a year. Marshall Magee (Automenu) and Dave Berdan (File Express) are grossing over \$1 million a year.

These big bucks have put stars (and dollar signs) in the eyes of thousands of programmers. That's why thousands of people make little from shareware while a handful prosper. If you have ideas about making real money from shareware, consider these:

(1) Writing the program is the easy part. Sending out disks to vendors, uploading to BBSs, responding to users, fixing bugs, adding requested features, and improving the documentation are not only more work, they aren't as much fun. They take up time and cost you money.

Two classic examples of proper marketing are Datastorm's Procomm and Magee's Automenu. Procomm was not the first shareware communications program, nor was it significantly better than Qmodem. But Datastorm's people knew how to market the program.

Marshall Magee took a program in one of the most crowded arenas of PC software—DOS menu programs—and turned it into a winner. In shareware circles, Magee is known as Mr. Image. He knows how to market, and he learned it on his own.

(2) You don't *have* to choose between shareware and nonshareware marketing. There is nothing sacred about shareware; it's just another way of marketing. As the prices of retail programs have approached shareware prices, programmers have had to be aware of both arenas.

If you market through both retail and shareware channels, don't cripple the shareware version. It's better to sell retail only than to pawn off a crippled shareware program.

(3) Write an outstanding program that businesses can use. Businesses are still the top shareware supporters. Games and DOS utilities may bring in a few bucks, but that's about all. Yet Another Text Editor or Yet Another Flat File Database Manager is not likely to generate much cash, either.

(4) Don't reinvent the wheel. If you want to rewrite the same old utilities over, fine. If you want to make money, know what's already there and be prepared to either out-program it, out-market it, or forget it.

(5) Hang in there. Many people write a program, get all excited while they send the program out and upload it to BBSs, and then get discouraged and quit when a month or two goes by without hearing anything. The shareware mill

grinds exceedingly slowly.

Eventually, your program will become ingrained in The System of pd/shareware distribution. If you have a good product and you continue to improve it and promote it, you can profit from it for years. I still get weekly registrations from a 1984 version of Diskcat at a P.O. Box that I have not publicized in over three years.

None of the top moneymaking shareware authors of today made much money their first six months. It takes that long for programs to start getting passed around. If you need quick income, advertise. Unfortunately, advertising's not likely to be worthwhile.

(6) Shareware is not a way to make money from inferior software. Some folks try marketing their program as shareware because it's not slick enough to sell on the commercial market. They shouldn't bother.

In theory, shareware has to be better than retail-only software because people get to try shareware before paying for it.

I could go on for pages and pages. In fact, I already have in the *Programmers Guide*, an on-disk guide for shareware programmers. It contains tips on how to polish a program and documentation, how to protect your software legally, how to find necessary supplies and services at the best prices, how almost *anyone* can get a MC/VISA merchant account, and more. The *Programmers Guide* is on CompuServe (GO IBMPRO) as GUIDE1.ARC and GUIDE2.ARC. You can also get it by sending \$5 to my address and asking for the *Programmers Guide* disk.

The Tax Man Cometh

By the time you read this issue, you should be starting to think about your tax return. My longtime favorite tax program, commercial or shareware, is AM-Tax.

PC-Tax is another veteran of the shareware world. It usually supports more forms than AM-Tax, but it's line-oriented, an approach which is more appropriate to a teletype terminal than a full-screen computer.

AM-Tax uses a full-screen, fill-in-the-form approach in which you scroll through replicas of the tax forms, fill in the blanks, and watch AM-Tax do calculations and carry totals forward for you.

AM-Tax works so intuitively you'll probably never read the manual. Any time you need a subschedule, all you

have to do is press F6. If no IRS subschedule exists, it creates a worksheet. After you use the worksheet or subschedule, you can return to the original schedule bringing along the balance.

In contrast, the last time I tried Turbo Tax, a popular retail-only tax program, I had to work through a confusing form menu to create a subschedule. Arbitrary worksheets were not an option. I got so bogged down in Turbo Tax that I never could complete the tryout.

The shareware version of AM-Tax supports form 1040 and schedules A, B, C, D, E, F, R, and SE. All printouts can be filed directly to the IRS. If you need more schedules, corporate or state forms, etc., they have more advanced versions.

Finding These Programs

All pd/shareware programs mentioned in this column are available from: public BBSs, CompuServe, user groups, and disk distributors such as PsL. In particular, PsL coordinates quite a bit with IBMNET on CompuServe.

IBMNET is divided into many different Forums and each Forum has many different Data Libraries. The quickest way to find a program on IBMNET is through the File Finder (GO IBMFF). Once there, you can search by program name or Key Word. For example, to find AM-Tax, you could search for the name or the Key Word "TAX."

CompuServe limits file names to six characters plus a three-character extension, so longer names may be abbreviated in ways you could never imagine. When this happens, the actual program name will often be recorded as one of the Key Words, so search for the program name in the Key Word field.

References

Chip Merchant
9285 Chesapeake Dr., Ste. L9285
San Diego, CA 92123
(800) 426-6375
(619) 268-4774

SitBack Technologies
9290 Bond, Ste. 210
Overland Park, KS 66214
(913) 894-0808

◆ ◆ ◆



8088 Hardware Development

Aiding The Ailing 8088

I discovered a way to diagnose a recalcitrant 8088 recently while working on controller cards. (It turned out that the steps outlined by Ed Ninsley for the 8031 in a back issue of *Circuit Cellar INK* can be modified for use with 8088s.)

My situation was pretty down and dirty: building an 8088 board and trying to get it working. I couldn't even get the 8088 to execute a single instruction.

Ed's steps boil down to the following.

- Disconnect the data bus from all memory devices and ground all the data lines. This forces the instruction `ADD AL,[BX]` onto the bus.
- Ground both of the interrupt pins (NMI and INT) and also the HOLD request pin (used by DMA and things like that).

This way the CPU will free-run, incrementing addresses through segment `FFFF`. Remember that most of these accesses will wrap around to `0000`; only the first 16 bytes come from the high end of memory.

Basically this gives you the ability to make sure that address latches and CPU clock are working before you add more things (such as memory decoding and memory).

Nathan Engle
6465 Piping Rock Lane
Indianapolis, IN 46254

High Memory Update

Since the original incarnation of my high memory card ("Filling In The Holes On Your XT," *Micro C*, Issue #52), I've rearranged and added things to my computer. Specifically, I added an expanded memory board.

Expanded memory is one of the greatest things since sliced bread, but it does require 64K of open memory space. For me, the simplest place to map it was in block D (`D000:0000` to `D000:FFFF`). With the commercial memory cards referenced in the article, this would mean separate cards if you wanted to fill in the remaining A and E blocks.

The SRAM card, however, may be modified as follows:

- Interchange address line A18 and A17 at the I/O bus interface;
- Move the chip select lines from pins 12 and 13 of the '138 to pins 14 and 15. (See Figure 1.)

This mod will give 128K, split between the A and E blocks. You might also want to rethink just which blocks, if any, you want to battery back-up. The change to E-DISK to make it a 64K RAMdisk is relatively trivial.

Also, in the eternal search for extra memory, a special note to Hercules (and clone) card owners. When operating in text mode, it is simply not possible to make the video access anything beyond the first 4K.

Let's call this an overly slavish emulation of the IBM mono adapter. When running one of the CGA simulation programs, only the top 32K is accessed.

Notice the hole between 4K and 32K (`B000:1000` to `B000:7FFF`). This memory gets used only when doing native Hercules graphics. For ordinary text applications, this 28K sits there idle. Scandalous! Quickly, load something up there and use it.

While on the subject of the HILOADing software mentioned in the article, I must say it's one of the more useful utilities I've ever written. But you should use some common sense in its application.

Some TSRs are designed to speed up various PC functions, such as video BIOS accelerators. Since they would be executing out of I/O bus memory, which is typically slower than motherboard memory, it doesn't make sense to load them up there, even though it's possible.

And, some TSRs (cache programs come to mind) are smart enough to "look around" where they are. From their point of view there isn't much memory left, so it wouldn't make much sense to install themselves. If they refuse to load, it may very well be something like that.

Finally, in the never-ending saga of living with SideKick, I've gleaned a few more tidbits

Micro Ads

Searching for that IMSAI? Need to sell your 286 to buy 386?

Private Listings is a national publication for computer and computer related items for sale by private owners. Savings galore on all types of recent and vintage items! Home of the \$5 two line one year ad!

Call or write for a free subscription!
Gateway Technologies, Inc.
180 Church Road
King of Prussia, PA 19406
(215) 354-0330

Reader Service Number 216

Experience Embedded Power

with the new Control-R II 8031 μ Controller Module

- ✓ 5 volt single supply operation (even with RS232)
- ✓ Data/Address/Control Bus Headers for expansion
- ✓ Direct Access to 8031 Ports 1 and 3
- ✓ 8K on board EPROM space
- ✓ Socket for 8K of on-board Static RAM (6264)
- ✓ MAX232 socket to provide RS232 compatible port
- ✓ Memory decoded to allow use of 8052-BASIC μ P
- ✓ Assembled and Tested, *not a kit*
- ✓ Compact size: 3.5" x 4.5"

Control-R II, Enhanced 8031 Module (SRAM & MAX232 optional) 64.95

Control-R II, Bare PC Board w/docs 25.00

Control-R I, 8031 Module (Port 1 & 3 access and MAX232 option only) 39.95

Control-R I, Bare PC Board w/docs 14.00

8K Static RAM I.C. 10.00
MAX232 for serial I/O 6.95

PseudoSam 51, 8031 family cross-assembler for MSDOS computers (5.25 inch disk w/docs) 50.00

Shipping \$3.00 US / \$5.00 Canada
Illinois Residents Add 6.25% Sales Tax

Cottage Resources Corporation

Suite 3-672C, 1405 Stevenson Drive
Springfield, Illinois 62703
(217) 529-7679

Reader Service Number 158

Learn a Modern Language

Our tutorials use a unique method to teach you to program in a modern language. Pascal, C, C++, Modula-2, beginning Ada, and advanced Ada tutorials are available for MS-DOS. Each tutorial is \$39.95 + \$3.00 P&H (in US). M/C or Visa accepted.

Coronado Enterprises

12501 Coronado Ave NE, Albuquerque,
NM 87122
(505) 293-5464

Reader Service Number 206

Technology for the Arts

Vision-16 Video Digitizer displays
32,768 colors, w software \$1495
Willow VGA-TV w NTSC Output \$495
Publishers' VGA/Frame Grabber \$595
Panasonic CCTV B&W Camera \$225

MIDI (music) Interface \$119
MIDI Programmers Toolkit \$39
Cakewalk 3.0 Sequencer \$150

joel Sampson Engineering
P.O. Box 550363 (214) 328-2730
Dallas, TX 75355 BBS 328-6909

Reader Service Number 176

HEXED?

Get HEXED™-The HEXadecimal Editor

Display and edit ALL characters in ANY file.

- Full-featured, menu-driven binary file editing
- Simplicity and power of a text editor
- Split-screen, simultaneous Hex value/ASCII symbol editing
- Easy access to file data formats, headers, etc.
- For all PCs & compatibles; 256K; DOS 2.0+
- \$49.95 USA, Canada. \$59.95 elsewhere.

Ideal for engineering, program development, hacking.

Technica REX Call for FREE Demo Disk.
141 Peyton Rd. Sterling, Va. 22170
703-444-TREX

Reader Service Number 218

NEW!

Now Micro Cornucopia has microform availability; 16mm microfilm, 35mm microfilm, 105mm microfiche copies of issues and articles provided through...

University Microfilms International.

300 North Zeeb Road, Ann Arbor, MI 48106-1346

313/761-4700

SWAP PROGRAMS

XSPAWN replaces the standard C spawn functions with functions that can transparently swap most of the parent process to expanded memory or disk before executing the child process. The XSPAWN functions are as easy to use as the standard C spawn functions. Includes all source code and libraries for Microsoft C and Turbo C.

Price \$79.95 plus \$4 shipping and handling. CA residents add sales tax.

Whitney Software, Inc.
P.O. Box 4999
Walnut Creek, CA 94596
415-933-9019

Reader Service Number 164

DSP32C COPROCESSOR BOARD

- 25 MFLOP Floating Point DSP
- High speed NUMERICS and GRAPHICS
- 32 BIT on board data bus
- 16 bit PC/AT interface
- ALL ON BOARD MEMORY DUAL PORTED
- Parallel and codec serial IO
- 15 ms 1024 point FFT
- Assembler, monitor, and libraries

Base board and ALL software \$950
640K 85ns memory \$300

SYMMETRIC RESEARCH

15 Central Way, Suite #9, Kirkland, WA 98033
(206) 828-6560

Reader Service Number 182

WISDOM OF THE AGES

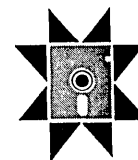
First electronic book of quotes, sayings & ideas brings over 1,000 of the world's greatest minds to PC screens & printers.

Select a subject. Use the vivid parade of timeless knowledge to act, write & speak better; earn more.

Subjects are organized into sections to give you the most benefit in the least time. Uses Filters to customize output; Dynamic mode to stimulate creativity.

It's never too early to get a head start.
Introductory Offer: \$79.00 for all 5 disks.
30 day money-back guarantee.

Requires 256K, DOS 2.0 or later & 2 floppies or hard disk. Add \$2.00 S/H. CA residents add 7%. Outside U.S. add \$10.



MCR Agency, Inc.

6116 Merced Ave. #81MC
Oakland, CA 94611
1-800-767-6797
FAX 415-444-6561

Reader Service Number 181

8748 EMULATOR

Simulate your 8748 programs in software before burning the EPROM. See all internal registers and i/o pins. Script files can simulate complex external events. Includes source. \$15

Other Products

DOS Source Fully commented 1.1, 2.1 \$15/\$45

Hercules Tools Graphics stuff w/source \$15

Math Library ascii Floating pt. w/source \$15



Information Modes

P.O. Drawer F
Denton, TX 76202
817-387-3339

1-800-628-7992

Reader Service Number 149

EGA FRACTAL MASTER

Whether fractal novice or fractal hacker, this easy to use yet powerful package lets you explore:

The Mandelbrot set	Julia sets
Dragonland	Self-squared dragons
Biomorphs	Newton's method
Peano curves	Fractal landscapes

Interrupt, save and resume plots at any time. Easily change colors, set boundaries, animate colors, create mirror images. Supports fast integer computations (even WITHOUT an 80386) and 80x87 math co-processors. Includes a slideshow program. Specify 5 1/4 or 3 1/2 disk.

Only \$25 (postage included)

Paul W. Carlson
602 North Avenue, #23
Burlington, Vermont 05401

Reader Service Number 185

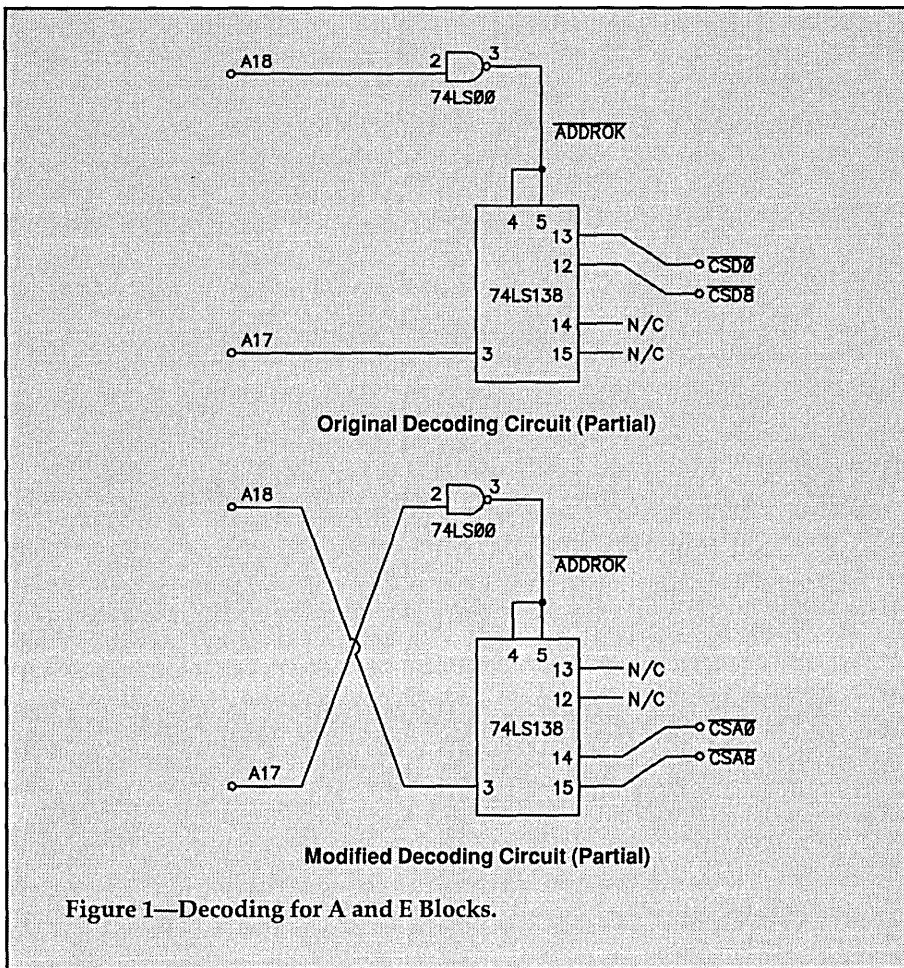


Figure 1—Decoding for A and E Blocks.

of its inner workings. Yes, it does use undocumented INT 21h calls (function 34h, at least: the INDOS flag) but many TSR writers do that. While surprising at first, I now use it myself!

Second, it seems that on every clock tick SideKick scans the interrupt vector table. If it sees someone else grabbing an interrupt it wants, it rearranges things its way. Every clock tick! Talk about chutzpah!

Oh yes, SideKick is unloadable. Don't try it, though, if you've HILOADED it. A guaranteed lockup.

Larry Shannon
5615 Truscott Terrace
Lakeview, NY 14085

Comment Comments

I agonized on whether to submit this as a technical tip, as it is merely an undocumented feature of a popular compiler. I decided to go ahead on the basis that it is a potentially useful feature that may increase programming productivity (but I may be overstating its usefulness).

I've always found it somewhat awk-

ward to enter C-style comments with their slash-star, star-slash pairs (*/* */*). So I was quite surprised to find out that Microsoft QuickC 2.00 accepts C++ style comments! This feature, as far as I can tell, is completely undocumented in the QuickC manual. I haven't been able to verify this feature in MS C 5.1. Have any MSC owners out there seen this feature?

I stumbled upon the feature while looking through the on-line help system. It's only used in one place: the sample program LOCK.C (to comment out two lines of code). There are no other references to C++ comments in the help system. (Note: C++ comments consist of a single slash-slash pair (*/***) with the comment text to the right, similar to assembly comments.) I've tested this feature with flawless results.

Things like this make you wonder if the programmers at Microsoft don't have a few other things they haven't told us about. Enjoy!

Steven Byrnes
10510 Emnora
Houston, TX 77043

//Editor's comment: Larry discovered C++ comments in FRACTINT (see the Shareware column in Issue #52) when converting the code to compile under Turbo C. FRACTINT was written in MS C 5.1, so there's your answer, Steven.

Friday The 13th!

I spoke to you Micro C folk a few weeks ago about some odd happenings on a Haupage 386 mammy board: that was, it appeared to kick into low gear approximately 30 minutes after boot. Well, I've tracked that dog down, and had my suspicions confirmed by one of the Houston area gurus.

Lo and behold, the poor thing had a virus. The infamous Friday the 13th virus, according to the guru. It's difficult to chronicle the sequence of events as nothing made any sense until it was over. I ignored the first law of troubleshooting (well, all right, the third law; it was (1) plugged in and (2) the power supply was fine). I should have known better when I saw that interrupts 08h and 21h were hooked. Unless you're running a legitimate program, there's no excuse for them to be hooked.

Anyway, to make a long story short, here are some of the clues for sniffing out the beast:

1) Interrupts 08h and 21h get hooked from the get-go. No need to run anything first. It gets into the boot code.

2) It expands uninfected COM and EXE files as soon as they are run. Note that it's sneaky about this as it restores the original time and date of the file, so no clues via time stamp.

3) In every case, the string "sUMs-Dos" was embedded in the infected files. Using the Norton Utility disk search or grep are two ways to find victims.

There may be easier ways to rid the beast, but I found DOS's FORMAT effective, saving the agony of real low level formatting (I can't always remember G=C800:5, or something like that). Anyway, there ye have it. May the fleas of a thousand camels invade the armpits of the worms that invent these demons. Disinfectedly yours,

Dave Stojan
10310 Lybert Rd.
Houston, TX 77041

Determining Required Stack Size

I used to find it difficult to determine the required stack size for my C programs. Then it occurred to me that the

Micro Ads

PSFX TSR prints any Epson FX-85 or IBM Graphics Printer compatible output as POSTSCRIPT, with graphics and IBM screen characters. \$85.00 Also available for Novell's NetWare.

PSPlot Convert HP-GL to POSTSCRIPT or EPS, any line width or color. POSTSCRIPT oriented editor with test downloader and error display. ASCII file print, any font or size. AutoCAD to POSTSCRIPT. Batch download of PC and MAC fonts. \$175.00

EPScreen Capture PC screens as tiny EPS files, frames optional. Includes IBM screen font. \$99.00

Legend Communications, Inc.
54 Rosedale Avenue West
Brampton, ON, Canada L6X 1K1
(416)450-1010 30 day guarantee (800)668-7077

PostScript

Reader Service Number 198

FIGHT PIRACY & PROTECT YOUR PROGRAM \$\$\$'s!

Since 1986, companies worldwide have been choosing Az-Tech security products. If you demand the strongest protection available, why not choose one of these "proven leaders":

- EVERLOCK Copy Protection
- EVERKEY Hardware "Key" Software Security
- EVERTRAK Software Security

For IBM and Compatibles. 30 day money back guarantee. Free info and demo disk available.

AZ Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800) 227-0644 FAX (816) 776-8398 (816) 776-2700

Reader Service Number 197

XenoCopy-PC \$79.95+S/H
PC-DOS program lets your PC Read/Write/Format over 350 formats

XENOFONT \$49.95 +S/H
high quality text screen printouts ideal for use in software documentation Bold face and reverse video supported.

XenoSoft 2210 SIXTH STREET
BERKELEY, CA 94710
415-644-9366

Reader Service Number 39

UPPER DECK FORTH \$49
Based on Forth-83 Standard Fully segmented architecture Fast compile, no link required Uses standard DOS files Integrated multi-file full screen editor Assembler, decompiler, source-level debugger Turnkey application support for IBM PC/XT/AT and compatibles with 256K, hard disk or floppy, DOS 2.0 or later Add \$3 for shipping and handling (outside USA \$15). CA residents add sales tax.

Upper Deck Systems
P.O. Box 263342,
Escondido, CA 92026 (619) 741-1075

Reader Service Number 211

LOW-BAND PC LOGIC ANALYZER

- Designed for general applications (below 1 MHz) where high-cost analysis is not needed
- Works with 5V digital logic
- 8 color-coded data lines & external clock input
- 6 range internal clock (1Hz-100KHz) for period sampling
- 8-bit logic probe mode
- Card plugs into IBM PC/XT/AT/386

36" cable with micro-clips Full-featured software Microsoft-mouse compatible Fast assembler data acquisition Help system & manual

PHOTONICS \$99.95
109 Camille Street
Amite, LA 70422
504-748-7090

MC,VISA, Company PO, COD

Reader Service Number 189

Please see us at Electro Booth #1121
IN CIRCUIT EMULATORS

8051 FAMILY
68HC11

CALL OR WRITE FOR FREE DEMO DISK AND BROCHURE!

NOHAU CORPORATION 51 East Campbell Avenue
(408) 866-1820
FAX (408) 378-7869
Campbell, CA 95008

Reader Service Number 186

PrestoPrint
A utility that sends print files to disk fast Lets you work and print simultaneously

- Uses less than 6k
- Uses any LPT/COM port
- Provides printer selection
- Supports Xon/off
- Auto-releases disk space when file is printed
- You select amount of disk space to be used
- Doesn't interfere with computer usage

Only \$49⁹⁵ 1-800-274-3007
Send check or money order to:
Gribbs & Associates
331 Glencrest Dr. Solana Beach CA 92075

the buffer of the 90's!

Reader Service Number 214

FLOPPY DISK FILE

- Stackable - For Convenient Storage
- Adjustable Metal Follower Keeps Disks Upright and Neat
- Sturdy Fiberboard Construction
- Easy Pullout Drawer With Metal Brasstone Handle and Card Holder
- 10 Dividers Included
- Available for 5.25" and 3.5" Floppy Disks

Model	Size	Capacity
33-1	3.50"	100
55-1	5.25"	150

Price Size (HxWxL) In.
\$15.70 5.25x4.25x15
\$17.20 7x6.5x15

Plus shipping \$3.20 Per Unit - In Florida, Add 6% Sales Tax
No shipping charges with orders of 12 or more boxes
WE ACCEPT VISA AND MASTERCARD

LOALCO Inc.
P.O. Box 7117, Delray Beach, Florida 33484
Telephone: 800-869-9241 • (407) 243-6222

Reader Service Number 215

Get All the Software in Issue 51 on Disk!

3 1/2 or 5 1/4 Disks Available
\$6 ppd (U.S.) or \$8 ppd (Foreign)

MICRO CORNUCOPIA
1-800-888-8087 (orders only) or (503) 382-8048

sPORTt
serial PORT tester

sPORTt is a serial port tester for people who don't want to be data communications experts, but use or write software for serial devices, such as plotters, mice, digitizers, modems, serial printers, or computer to computer links.

- displays all data going in and out
- displays ASCII, decimal, hex, binary
- lets you choose handshaking lines
- sends/receives disk files
- automatically runs port diagnostics Order today only \$80

WHS
3037 Grass Valley Hwy #8201 Auburn, CA 95603
916-885-2480

Reader Service Number 192

Hercules Console Driver

Fully emulated text output in graphics mode on your Hercules Graphics Card. Great for programmers debugging & working with graphics programs. Write directly to the screen or run DEBUG without leaving HGC graphics mode.

Just \$24 (Shipping & Handling Included)

Winter City Software
8723 - 162 St
Edmonton, AB Canada T5R 2M2
(403)484-3187

Reader Service Number 203

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 **Ptech**
907 North 6th St. Lake City, MN 55041 (612)345-4555

Reader Service Number 3

Why you want BATCOM!

BATCOM is a batch file compiler that compiles your ".bat" files to ".exe" files to make them faster, more professional, and more capable. BATCOM extends DOS with new commands so you can read keyboard input, perform arithmetic, use subroutines, and much more. In addition, BATCOM protects your source code, and you can distribute your compiled programs without royalties. For IBM PC. Only \$59.95. Order today!

Wenham Software Company
5 Burley St.
Wenham, Ma. 01984
(508) 774-7036

Reader Service Number 124

NEW COMPUTER BOOKS

Select from thousands of new titles in C++, C, ADA, Pascal, Modula, Assembler, Basic, Lisp, Fortran for PC's, MAC, Unix and VAX. CAD, Desktop Publishing, Dbase, Word Processing, Business & more. Not a book club, no membership fee or required purchases. Service for businesses & professionals.

Quality at low prices.

up to 40% OFF LIST
SMALL MINIMUM ORDER
send for FREE information to

Computer Book Co.
PO 7076
Sunnyvale, CA
94086-0746

Reader Service Number 167

Coming Up In Issue #54

The meaning of life, the universe and everything. This series will take a quantitative look at this engrossing subject rather than mucking about in silly philosophical mud puddles.

You won't find this issue at your local newsstand, nor will it be available by subscription. Issue #54, coming soon to a place very close to you. Watch for it. (Thanks, Dave Stojan.)

SHAREWARE AUTHORS: Give your programs the chance they deserve! With **Megapost**, your program is uploaded to CompuServe, GEnie, Delphi, BIX, and EXEC-PC. The low fee is probably less than you would pay just to open accounts with all these services. Included are one half-price update and notification of the downloading procedure and 3-month download count for each service. 15% discount for ASP members. Write for a free brochure.

Andrew M. Saucchi, Jr.
641 Koelbel Ct. Baldwin, NY 11510-3915

Reader Service Number 200

Micro Ads

Figure 2—STKSIZ.ASM

```

.MODEL SMALL

EXTRN  _end:WORD           ; stack bottom
EXTRN  __amsg_exit:PROC    ; write message and exit

.DATA

PUBLIC STKHQQ
STKHQQ dw offset DGROUP:_end+256 ; stack bottom + slop
PUBLIC __stksiz
__stksiz dw -1                ; larger than possible

.CODE

PUBLIC __chkstk
__chkstk LABEL PROC

    pop     cx                ; return offset
    IF     @codesize
    pop     dx                ; return segment
    ENDIF

    mov     bx,sp
    sub     bx,ax              ; subtract local requirement
    jc     overflow           ; stack overflow
    cmp     bx,[STKHQQ]
    jb     overflow           ; stack overflow
    mov     sp,bx              ; new stack pointer
    sub     bx,[STKHQQ]        ; get free stack space
    cmp     bx,[_stksiz]
    jae    return
    mov     [__stksiz],bx     ; replace with smaller value

return:
    IF     @codesize
    push    dx                ; return segment
    push    cx                ; return offset

dummy    PROC FAR
dummy    ret                  ; return to DX:CX

ELSE     jmp     cx            ; return to CX
ENDIF

overflow: xor     ax,ax
           jmp     __amsg_exit ; write message and exit

END

```

◆ ◆ ◆

"stack probe" routine (called at every function entry point to verify that there is enough remaining stack space to allocate local variables required by the function) could be modified to also maintain a global variable that would contain the smallest amount of free stack space up to that point.

The routine presented here (see Figure 2) was written for Microsoft C and is functionally equivalent to the Microsoft stack probe routine (called `_chkstk`) except that a global variable called `_stksiz` is maintained. My routine is only a little larger and slower than its counterpart.

To use my routine you must first delete the `chkstk` module from the Microsoft run-time library you are using. This is because `__chkstk` is not a procedure label.

When testing a program, I print

`_stksiz` just before exiting. Assuming stack checking is enabled (see your compiler documentation), `_stksiz` will contain the smallest amount of free stack space for the portion of the program that was actually executed.

The routine presented here is for the small memory model and was written for MASM 5.0. To change memory models you need only change the `.MODEL` directive.

Jack Whitney
Whitney Software, Inc.
P.O. Box 4999
Walnut Creek, CA 94596
(415) 933-9019

◆ ◆ ◆

ADVERTISER'S INDEX

ISSUE 53

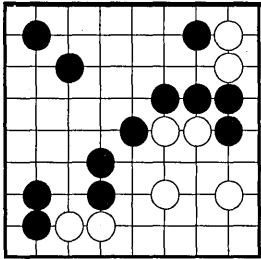
72 Acquired Intelligence	83	181 MCR	91	** Sintar Software	76
187 Alphax Systems, Inc.	78	** Micro Cornucopia	74, 77	219 Software Blacksmiths, Inc.	56
160 Annabooks	76	37 Microprocessors Unlimited	63	129 Software Science, Inc.	33
4 Austin Codeworks	31	2 Microsphere, Inc.	94, Inside Front	40 Star-K Software Systems	18
197 Az-Tech Software, Inc.	93	** Midnight Engineering	87	202 Symbologic	59
				182 Symmetric Research	91
147 Berry Computer	49	186 NOHAU	93	** Turbo Users Group	87
214 Bluebird Systems	93	110 Nu-Mega Tech	2	218 Technica Rex	91
				178 Traxel Labs	32
31 CC Software	80	201 OPENetwork	69	194 Turbo Power	51
185 Carlson	91	193 Oregon Software	75		
15 Cascade Electronics	39	216 OSCS	53	211 Upper Deck Systems	93
217 Catenary Systems	75	161 Opal Fire Software	79		
167 Computer Book Co.	94			62 V Communications	43, 82
7 Compuview Products, Inc.	11	3 PC Tech	93, Back Cover	208 Vernier Software	44
206 Coronado Enterprises	91	113 Paradigm Systems	80		
158 Cottage Resources	91	119 Peripheral Technology	78	124 Wenham Software Co	93
		189 Photonics	93	192 Western Hydrologic Systems	93
10 Emerald Microware	29	** Programmer's Journal	Inside Back	169 Western Wares	72
93 Erac Co	45			164 Whitney Software	91
		139 Quantasm Corp	72	203 Winter City Software	93
204 Gateway Technologies, Inc.	91	** RJSwantek Inc	37		
212 General Software	63	210 Regan Consulting & Programming	66	39 Xenosoft	93
** Genus Microprogramming	19	** Revolution 2	25		
		166 Robotic Systems	79	207 Young Software Engineering	23
149 Information Modes	7, 91	171 Ryle Design	84		
22 Integrand	6			70 Zortech	1
213 JB Software	5	176 Sampson Engineering	91		
		200 Saucci	94		
198 Legend Communications	93	127 SemWare	13		
215 Loalco	93	162 Semi-Disk Systems	84		

** Contact Advertiser Directly.

When you write for information, please tell these folks you read about their products in **Micro Cornucopia**.

What's this I hear, no more Micro C?
 No more around the Bend or SOG?
 What will become of the human race
 ("All watched over by machines of loving grace")
 With no PC mags of culture and taste
 to keep computers in their place?
 Whate'er the reason, I know it's just.
 Close down the rag if you think you must.
 But know your humor will be sorely missed—
 If you start up another put me on your list.
 (sob)
 Old friend, I hardly knew ye.
 (Quotation courtesy Richard Brautigan)

—Buddy McManus



THE
LASTPAGE

Man-Machine Interactions—

The Edges of Reality

By Gary Entsminger

P.O. Box 2091
Davis, CA 95617

Can you control your own reality—even a little bit? Here's a way to find out.

Thanks in part to the “strange” point of view cast by quantum physics in this century, philosophers, psychologists, other-ologists, and just plain folk have readdressed the classic question: *What role does consciousness play in the creation of reality?*

In particular, the Princeton Engineering Anomalies Research Program (PEAR), has been challenging the common sense view of reality through carefully controlled experiments.

Robert Jahn and Brenda Dunne have collected ten years' worth of PEAR results in a fascinating book, *Margins of Reality, the Role of Consciousness in the Physical World*. *Margins* attempts to:

- (1) produce a history of scientific attempts to credit and discredit paranormal events;
- (2) define parapsychology;
- (3) review the design, operation, and results of an ensemble of engineering experiments. These experiments address the interaction of consciousness with both inanimate devices and systems embodying random processes;
- (4) create a theoretical framework to explain the PEAR results.

The story *Margins* relates is, to say the least, *ver-y in-ter-est-ing*.

Man-Machine Connections

The PEAR Program's experiments fall into two broad categories:

- (1) man-machine interactions;
- (2) precognitive remote perception.

PEAR's interest in man-machine interactions stems from concern about the sensitivity of technology. One possible

subtle influence is human consciousness.

I found two of the PEAR experiments especially intriguing. Both utilize random event generators (REGs) to create random patterns. One uses microelectronic circuitry; the other uses a random cascade apparatus (a macroscopic contraption, 10 feet wide and 6 feet high, which lets $\frac{3}{4}$ " balls funnel down through a quincunx array of 330 pegs).

In both experiments, PEAR establishes a control by running the REGs unattended. These unattended runs establish the likelihood of random output. Then an operator positions herself several feet from the REGs and tries to affect the output by thinking about it.

In the microelectronic experiment, the experiment simulates a coin toss (the outcome is binary: 1 or 0). The operator tries to increase the output of either 1s or 0s (in the controlled runs the output of 1s and 0s is roughly equal).

In the macroscopic experiment, the controlled runs produce a cascade of balls that's bell-shaped. In this scenario the operator now tries to shift the cascading of balls either left or right.

The results, as you might expect, vary among operators. Some had an extraordinary ability to affect the output, particularly as the number of experimental runs increased.

Typically, an operator's effect wasn't significant in the first few hundred or so runs. But as the number of runs increased, the operators got better.

My Test/Your Test

During the past 15 years, I've encountered many paranormal claims, most of them seriously flawed one way or another. Most often, experimenters claim more than their experiments show, fail to completely isolate variables, or are

unable to replicate their results.

PEAR avoids the common pitfalls of paranormal research, doesn't claim too much, and seems well on track to discovering important clues to the role consciousness plays in creating reality.

Just for fun, try an experiment. Set up an REG using the randomize and random functions in Turbo Pascal or C. Make sure you call randomize before each call to random, otherwise each run is really only one random event and seriously flawed as a control for the experiment. Calling *randomize* reshuffles the deck by calling the system clock to reseed the random number generator. For example—

```
uses CRT;
var
  I, R, Tot1, Tot0:integer; C:char;
begin
  Tot1 := 0; Tot0 := 0;
  for I := 1 to 10000 do begin
    randomize; R := random(2);
    if R = 1 then Tot1 := Tot1 + 1
    else Tot0 := Tot0 + 1;
  end;
  writeln('Total1: ', Tot1);
  writeln('Total0: ', Tot0);
end.
```

Generate ten groups of ten thousand samples. Call it the control. Then repeat the experiment, but this time concentrate throughout the process (an event at a time) on either a 1 or 0 outcome. Who knows—maybe the consciousness of Micro C affects reality?

References

Jahn, Robert G. & Brenda J. Dunne; *Margins of Reality*; Harcourt, Brace, Jovanovich; 1987; \$17.95 (paperback).

◆ ◆ ◆

The complete *Programmer's Journal* "On Graphics" series—assembled in one volume for the very first time!

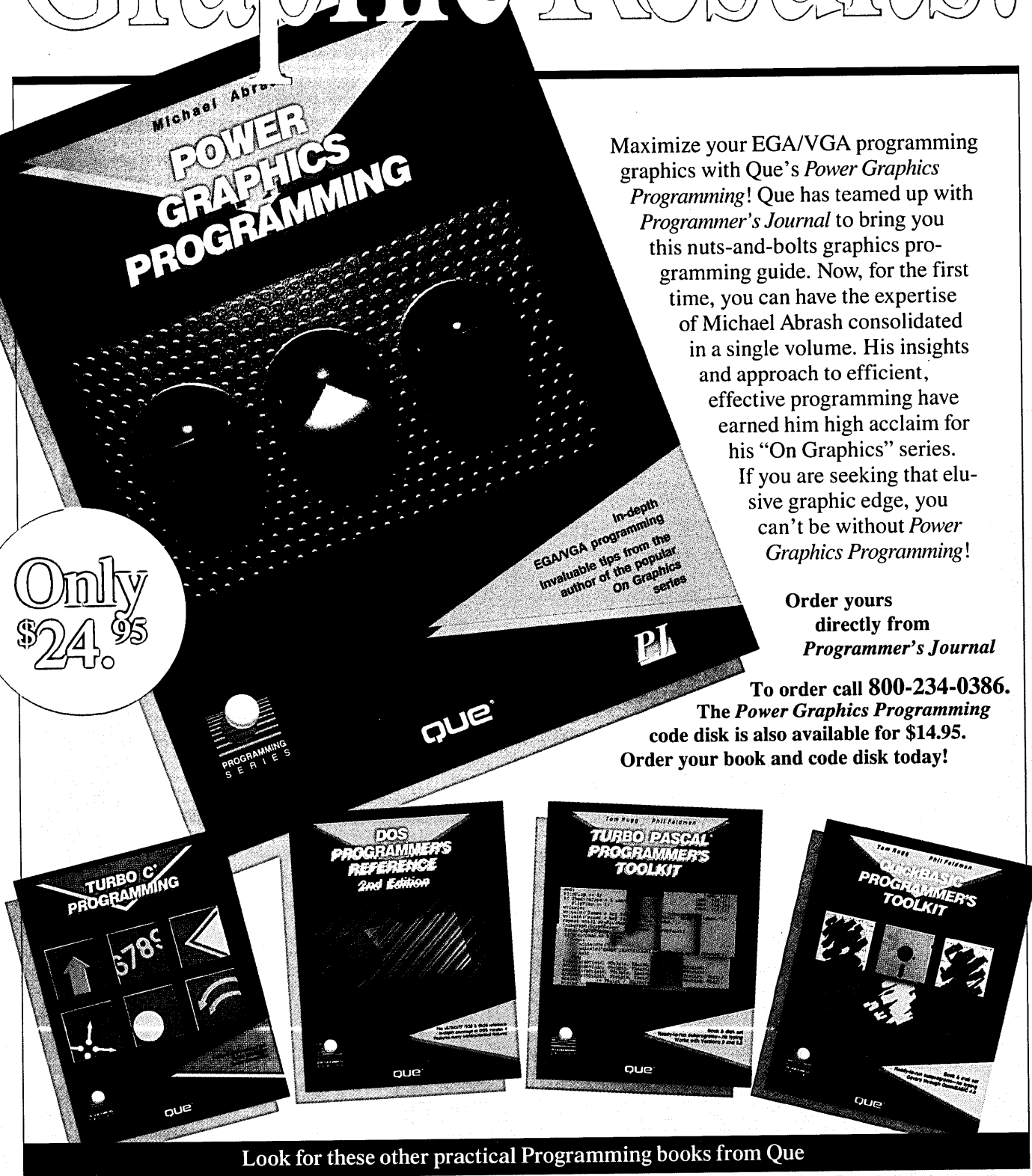
Graphic Results!

Maximize your EGA/VGA programming graphics with Que's *Power Graphics Programming*! Que has teamed up with *Programmer's Journal* to bring you this nuts-and-bolts graphics programming guide. Now, for the first time, you can have the expertise of Michael Abrash consolidated in a single volume. His insights and approach to efficient, effective programming have earned him high acclaim for his "On Graphics" series. If you are seeking that elusive graphic edge, you can't be without *Power Graphics Programming*!

Order yours directly from *Programmer's Journal*

To order call 800-234-0386. The *Power Graphics Programming* code disk is also available for \$14.95. Order your book and code disk today!

Only \$24.⁹⁵



Look for these other practical Programming books from Que

Available in better bookstores and computer stores everywhere. Or call 800-428-5331, ext. PJ01.

que[®]

Contact Advertiser Directly

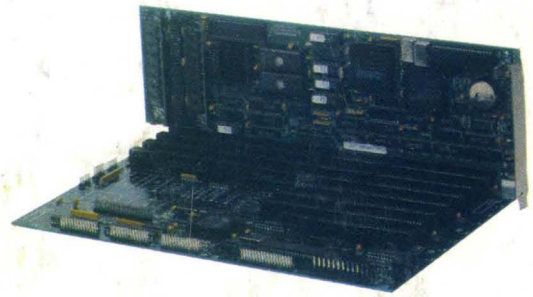
VERY HIGH PERFORMANCE

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.



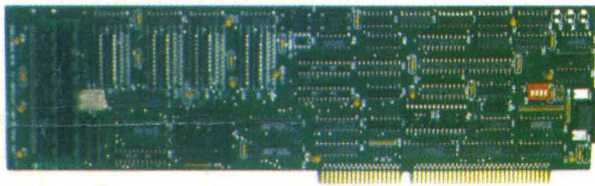
PC Tech X24 and ASMB

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.

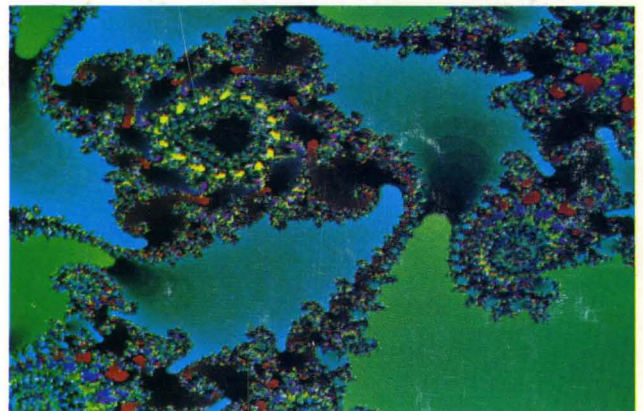
About PC Tech

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