

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Embedded Systems

Looking for an easy way to put together an embedded controller? Then don't put down this issue.

Embedding An XT Motherboard

This is for those of you who want it all. Cheap hardware, a familiar (MS-DOS) environment, and complete tools for development and debugging.

Writing A Neural Network In C, Part 1

Neural Nets are delivering nearly everything that AI promised. Our series includes all the software needed to experiment with this fascinating technology.

LIMBO, Part 4

page 30

Give your robot a body (and part of a mind).

Marketing Your Own Software

page 66

How and where should you advertise your new product? What kind of response should you expect when you do advertise?

And More . . .

Getting Started in Hardware
The Poet and The Computer



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 44MB Mitsubishi MR535 voice coil hard drive with an access time of 28ms, and a super fast 1:1 interleave HD controller card.

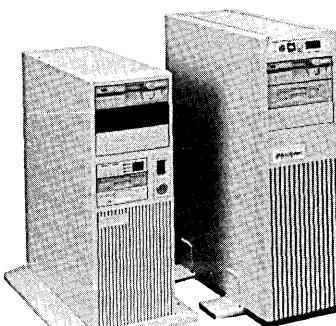
- 25Mhz 80386 CPU, AMI Bios, Full Size MB
- 4MB of RAM, expands to 8MB on MB
- 1.2MB and 1.44MB Floppy Drives
- 12" Amber Monitor w/Mono Graphics Card
- 101-Key Enhanced Keyboard
- Serial/Parallel/Game Ports, Clock/Calendar
- 200-Watt Power Supply
- Socket for 80287, 80387 or Weitek
- Full 1 Year Warranty

w/Std AT Case.....\$2395
w/1MB.....\$250 Less
20Mhz 386.....\$150 Less

\$2495

(Mega Tower)

Tower Cases w/Power Supply



Mini Tower
\$225

Standard Tower
\$269

1MB AT SYSTEM

Includes: 1MB RAM, 1.2MB & 1.44MB FD, 44MB Mitsubishi MR535 Voice Coil HD (28ms), Fast 1:1 Interleave HD Controller, Mono Graphics Video Card, 12" Amber Monitor, 101 Key Keyboard, Serial/Parallel/Game Ports, Clock/Calendar, Full 1 Year Warranty. FREE assembly and testing.

6/10Mhz \$1249
6/12Mhz 1295

XT SYSTEM

4.77/10Mhz w/2 360K Floppies..... 725
4.77/10Mhz w/20MB HD 929
4.77/10Mhz w/30MB HD 955

Color options for our systems
(includes video card and monitor)

ATI Graphics Solution 39
CGA Color 175
EGA Color 350
VGA Color (Analog
w/Mitsubishi monitor) 495
CGA/EGA/VGA (Multisync) 495

FLOPPY DRIVES

TEAC 360K 74
TEAC 1.2MB 85
TEAC 3 1/2" 720K 79
TEAC 3 1/2" 1.44MB 90

HARD DRIVES

XT 20MB Miniscribe/Kalok
8425 (65ms) 239
8425 w/controller 279
XT 30MB Miniscribe/Kalok
8438 (65ms) 249
8438 w/controller 299
AT 44MB Mitsubishi MR535,
Rotary Voice Coil (28ms) 459
AT 71MB Miniscribe 3085
Rotary Voice Coil (25ms) 699

VIDEO CARDS

Mono Graphics w/Parallel port 39
CGA Graphics Card 45
ATI Graphics Solution
Mono w/CGA Emulation 79
CGA/EGA w/256K 95
CGA/EGA/VGA 8-Bit 149
CGA/EGA/VGA 16-Bit 189

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.
NCL, 1:1 Interleave, MFM	522 KB/Sec.
DTC 7287 1:1 Interleave, RLL, w/Cache	799 Kb/Sec.

- Changing from a 61ms Hard Drive to a 28ms Hard Drive increases performance by 10%!
- Changing from a standard AT controller to 1:1 interleave controller improves performance by 300%!

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

CONTROLLER CARDS

NCL 1:1 MFM Controller \$110
DTC 7287 1:1 RLL w/Cache 165

CABINETS

XT Slide Case, Lock, LED 38
AT Slide Case, Lock, LED
3 half ht., 1 full ht. drives 72
Baby Tower, w/200 watt PS,
2 half ht., 2 - 3 1/2" Drives,
Baby Motherboard ONLY 145
Mini Tower, w/200 Watt PS,
3 half ht., 2 - 3 1/2" Drives,
holds Full or Baby size MB 225
5 Bay Std. Tower, 250 Watt PS,
3 half ht., 1 full ht drives 269
6 Bay Mega Tower, 200 Watt PS 239

MONITORS

12" Amber Monochrome TTL 89
CGA Color RGB 249
EGA/CGA Autoswitch .31 dot 362
CGA/EGA/VGA
Multisync (.31 dot) 489
VGA Analog 489
(Mitsubishi .28 dot) 489

KEYBOARDS

Chicony Enhanced 101-Key \$67
Keytronic Enhanced 101-Key 67
Focus 101-Key Tactile, Switchable
Control/Caps Lock, Dust Cover 89
(*#1 Find by Micro C Staff)
* All Keyboards, XT/AT switchable*

SOFTWARE

MS-DOS 3.21 w/GW Basic 49
MS-DOS 3.3 w/GW Basic 95
DR-DOS 3.3 w/GEM 49
SpinRite Disk Optimizer 49
386Max Memory Manager for
386 Systems 69

CHIPS

XT/AT/386 Memory Call
Math Co-Processors Call

POWER SUPPLIES

150-Watt XT Power Supply 50
200-Watt AT Power Supply 78

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

XT

XT Multi-IO, Ser/Par/Clock/
Game/2 Floppy Drives 47
XT Floppy Controller 19
XT 640K RAM Card (w/0K) 25
XT 2MB EMS Memory Board (0K) 49

AT/386

AT Multi-IO, Ser/Par/Game 33
2nd Serial Port Kit 20
AT 2MB EMS Memory Board (0K) 99

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.

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# _____

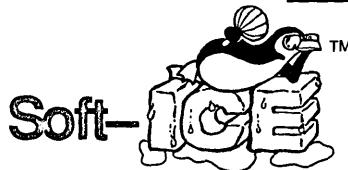
Exp.Date _____ Tel _____

All MicroSoft trademarks are acknowledged.

- DEVELOPER'S EDITION \$450 (Save \$200)
 - C++ COMPILER \$199.95
 - C++ DEBUGGER \$149.95
 - C++ TOOLS \$149.95
 - LIBRARY SOURCE CODE \$149.95
 - COMPILER & LIBRARY SOURCE \$299.95
- For US orders please add \$5.05 shipping
Overseas orders at international mail rates.

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

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

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 2.0 includes EMS 4.0 driver.
Attention Windows Developers!
Version available for CVW.

MICRO CORNUCOPIA

JANUARY/FEBRUARY 1990 - ISSUE NO. 51

FEATURES

8 Gene Toner

Embedding An XT Motherboard

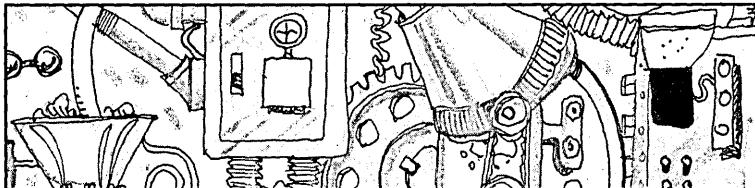
Wouldn't it be wonderful if you could use one of those \$50 XT cards as an embedded system? You can, ducky. You really can.

16

Russ Eberhart and Roy Dobbins

Writing A Neural Network in C, Part 1

Russ and Roy are applying neural nets to some incredibly practical projects. In this series you'll get the theory, the application, and you'll get the code so you can try your own applications. (Also check out our neural oriented Tidbits.)



26

Gregory K. Landheim

3D-Surface Generation, Part 2

Greg finishes up his 3-D project by getting right into the code.

28

Norman Cousins

The Poet and The Computer



30

Bob Nansel

LIMBO, Part 4

42 Bruce Eckel

Capturing & Graphing A Voice, Part 2

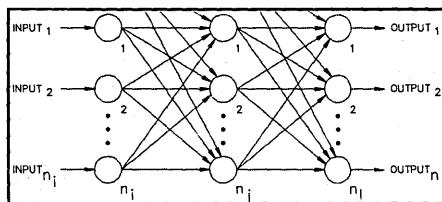
Bruce tackles the digital half of his speech problem.

50

Karl Lunt

Getting Started In Hardware

This is for all of you who've asked for an entrance level article on hardware.



COLUMNS

58 C'ing Clearly

64 Culture Corner

66 On Your Own

78 Units and Modules

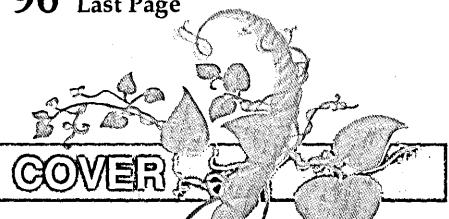
82 Shareware

90 Techtips

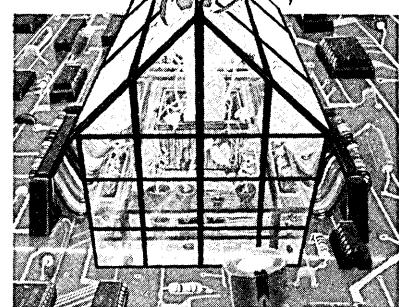
FUTURE TENSE

84 Tidbits

96 Last Page



COVER



Cover Illustration by Jerry Werner.

MICRO CORNUCOPIA

Editor and Publisher
David J. Thompson

Associate Editors
Gary Entsminger
Larry Fogg
Cary Gatton

Contributing Writers
Anthony Barcellos
Bruce Eckel
Michael S. Hunt
Scott Ladd
Laine Stump

Advertising & Distribution
Gareth Thompson

Accounting
Sandy Thompson

Reader Services
Nancy Ellen Locke

Graphic Design & Production
Carol Steffy

MICRO CORNUCOPIA (ISSN 0747-587X) is published bi-monthly for \$18 per year by Micro Cornucopia, Inc. 155 NW Hawthorne, Bend, OR 97701. Second-class postage paid at Bend, OR and additional mailing offices. POSTMASTER: Send address changes to MICRO CORNUCOPIA, PO Box 223, Bend, OR 97709. Printed on 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 1-800-888-8087, 9 am to 5 pm, Pacific time, M-F. FAX your VISA or MC order to us, our FAX number is (503) 389-6833.

TECHNICAL ASSISTANCE

For help call 503-382-8048,
9 am to noon Pacific time, M-F
BBS - 24 hrs. 300-1200-2400 baud
8 Bits, No Parity, 1 Stop Bit

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



**AROUND
THE BEND**

By David J. Thompson

Not So Slick**Our Paper**

You've probably noticed that our paper has changed. This paper isn't quite as white as our old paper and it's coated. That's the bad news. The good news: it's made of 50% recycled pulp and it's the least glossy coated paper of this type we've found.

In the past we paid top dollar because we insisted on the highest quality uncoated paper. (No coating—no glare.) But in the last six months, paper companies have raised the price for our wonderful paper a bunch, while they've substantially reduced what they're willing to pay for recycled paper.

Maybe we can save some money and (because of the coating) improve our reproduction. Plus, since we purchase over 14,000 pounds of paper each time we print a magazine, we'll save about 42,000 pounds of trees a year. Just by using recycled paper.

I broached the subject of coated versus uncoated at SOG East. Everyone voted for uncoated. Then I added that it contained recycled pulp. It was unanimous again. They voted to try the new paper.

Futures, Bricks, And The Wall Street Journal

The call I answer most often isn't a tech call (and it isn't nature), it's a sales call.

"I'm calling with the information you requested on commodities futures."

"Commodities futures? I didn't request any information on commodities futures."



Contented Timber.

Continued on page 73

You asked for a place to put your things...



The TeleTM FILE SYSTEM is just the thing

BerryComputers presents The Tele Toolkit — a complete Operating Systems Kernel

If It's Data, It Must Be A File

Tele's file system is modular at several levels. FS is responsible for all storage and transfer of data.

The physical interface to disk devices is through MS-DOS installed device drivers (MS-DOS itself is by-passed). Therefore, Tele will work with the same devices that MS-DOS supports.

Separate from the physical interface is the directory structure. Tele supports installable file systems; each device can have a unique media format. Only MS-DOS compatible media are supported in FS.

Some other Tele components involve installable file systems. For instance, the UX component emulates the Unix kernel. Most of its code supports Unix media. Networks are supported by an installable file system that causes directory operations to be performed on a device in a remote computer system. FS itself only supports MS-DOS media, but it provides the main hooks by which any other file system can be emulated.

The bulk of Tele FS code supports hierarchical directory structures and file redirection. Because MS-DOS is not involved, you can use FS to avoid its restrictions.

Tele FS also includes serial communications support. 8250 controllers are supported in bidirectional interrupt mode. Ring and break indicators are also supported. Serial ports can be accessed directly, or redirected through the file

system. Files can also be redirected from the keyboard and to the console display and printer.

To support efficient communication and storage, FS contains a modified Huffman compression algorithm. The modification automatically recognizes fields within records and applies a different compression tree to each type of field. Compression can be processed directly on blocks or continuously and transparently within the file system.

All source code, in C and assembly, is included. Tele SK is required for FS. CD is also required for console device support.

Demo Diskette	\$ 5	(refundable with purchase)
SK system kernel	\$50	(multitasking)
CD console display	\$40	(windows, requires SK)
FS file system	\$40	(MS-DOS media, requires SK)
OS core	\$130	(SK, CD, and FS)

Telephone support is freely available.

The Tele Toolkit is available from:

Crosby Associates
P.O. Box 248
Sutter Creek, California
95685

CALL NOW TO ORDER:
(209) 267-0362 (FAX) (209) 267-0115

Visa, Mastercard, American Express & Discover Card accepted.

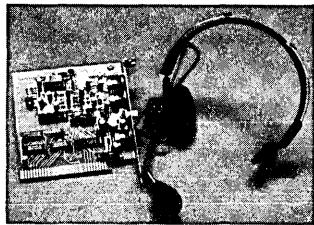
MS-DOS is a trademark of Microsoft Corporation.
Unix is a trademark of AT&T

VOICE MASTER KEY®
VOICE RECOGNITION
SYSTEM
FOR PC/COMPATIBLES &
TANDY 1000 SERIES
A FULL FEATURED VOICE I/O SYSTEM

GIVE A NEW DIMENSION TO PERSONAL COMPUTING...The amazing **Voice Master Key System** adds voice recognition to just about any program or application. Voice command up to 256 keyboard macros from within CAD, desktop publishing, word processing, spread sheet, or game programs. Fully TSR and occupies less than 64K. Instant response time and high recognition accuracy. Voice recognition tool-box utilities are included. A genuine productivity enhancer!

SPEECH RECORDING SOFTWARE...Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files you can add to macros for voice recognition verification response. A complete, superior speech and sound development tool.

SOFTWARE CONVERSION CODES...The **Voice Master Key System** operates a growing list of third party talking software titles using synthesized phonetics (text-to-speech) or digitized PCM, ADPCM, and CVSDM encoded sound files. **Voice Master Key System** does it all!



EVERYTHING INCLUDED...**Voice Master Key System** consists of a plug-in card, durable lightweight microphone headset, software, and manual. Card fits any available slot. External ports consist of mic inputs and volume controlled output sockets. High quality throughout, easy and fun to use.

ONLY \$149.95 COMPLETE

**ONLY \$89.95 FOR TANDY 1000 SL/TL MODELS—
SOFTWARE PACKAGE ONLY.**

Requires Tandy Brand Electret microphone.

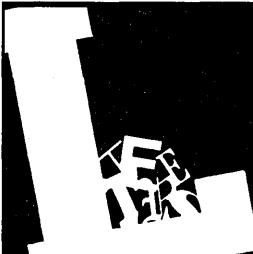
ORDER HOTLINE: (503) 342-1271

Monday-Friday, 8AM to 5PM Pacific Time

Visa/MasterCard, company checks, money orders, CODs (with prior approval) accepted. Personal checks subject to 3 week shipping delay. Specify computer type and disk format (3½" or 5¼") when ordering. Add \$5 shipping charge for delivery in USA and Canada. Foreign inquiries contact Covox for C & F quotes. **30 DAY MONEY BACK GUARANTEE IF NOT COMPLETELY SATISFIED. ONE YEAR WARRANTY ON HARDWARE.**

CALL OR WRITE FOR FREE PRODUCT CATALOG

 **COVOX INC.** 675-D Conger St.
Eugene, Oregon 97402 U.S.A.
TEL: 503-342-1271 • FAX: 503-342-1283



Letters

Long Lawn, Laments Loyal Loiterer

My motorcycle vacation took me through Bend. I have not been in Bend since PµP (pre-microprocessor) days, and there have been a few changes! A tour of Bend would not have been complete without a swing by Micro C.

I don't make a habit of visiting publishing headquarters, but Micro C is an exception. Alas, no one was home—probably out rafting/tubing instead of mowing the lawn. (Gosh, I hope that wasn't too metaphysical.)

Keep up the good work....

Nils R. Olson
419 Woodhaven Drive
Vacaville, CA 95687-5941

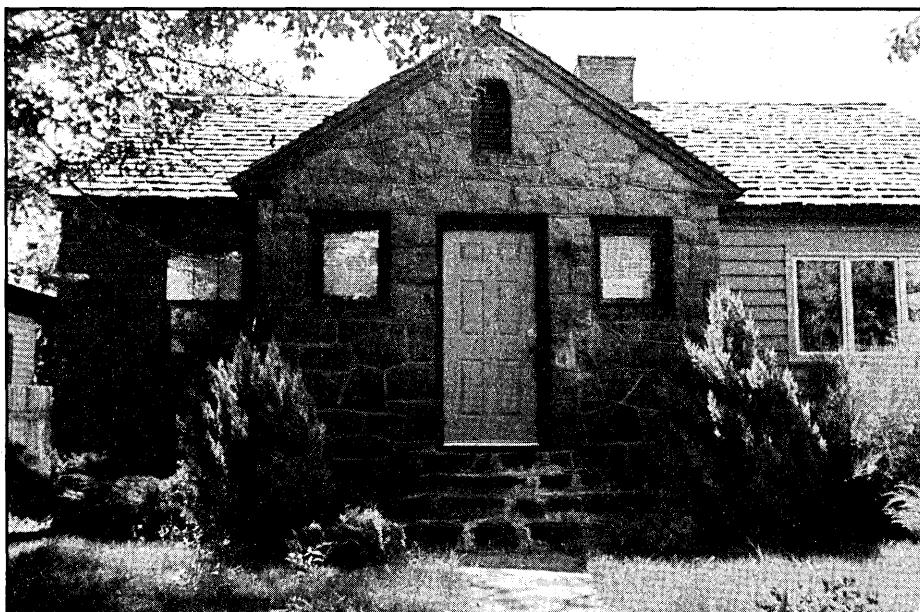
Editor's note: Thanks Nils for the photo. Yes it does look like it's time for our annual mow. (It's more physical than metaphysical.)

SpinRite Defended

I've been a faithful follower of *Micro C* from the day I bought my first microcomputer (my old Kaypro II), but I think that you goofed when you trashed SpinRite in recent issues. I bought SpinRite as a last resort to save my ailing hard disk (a Miniscribe 32 MB, 65 ms, on a card—the kind you get from CompuAdd) in my XT clone (built from a kit from, where else, MicroSphere).

I was getting a lot of read/write errors and an increasing number of bad sectors when running the surface analysis part of PCTools. Doing a destructive reformat (and, of course, reloading all my files from backup) was the only way to get the disk back into reliable shape. Then the problem would reappear and get worse within a month or so.

I ran SpinRite on the flakey drive (full, deep testing with nondestructive



Reader Service Number 143

```

Block Edit File Goto Help Disc Print Search Undo Window Config
-WINDOW E-INSTALL.C
while (TRUE) { /* process
    j=getseq(keybuf,i);
    if (i == 0) {
        /* Check for delimiter
        if (*keybuf == delimiter) {
            /* Check if displayable
            if ((*keybuf >= ' ') &
                (*keybuf <= 'z') &
                (*keybuf >= 'A') &
                (*keybuf <= 'Z')) {
                printf("%c",*keybuf);
                *codbuf++ = *keybuf;
                *codbuf++ = 00; /* High byte 00 for chars */
                *codbuf = KFF;
                return(TRUE);
            }
        }
    }
}

```

-WINDOW Y
***** INSTALL.C(675) : error 65: 'codbuf' : undefined
-WINDOW H

Edit source file; then press <CTRL-E> for next error, <ESC> for menu

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), TI Professional and others.

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

Introducing . . .

The 1st Family of Low Cost, Powerful Text Editors

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

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

Compare Features and Speed

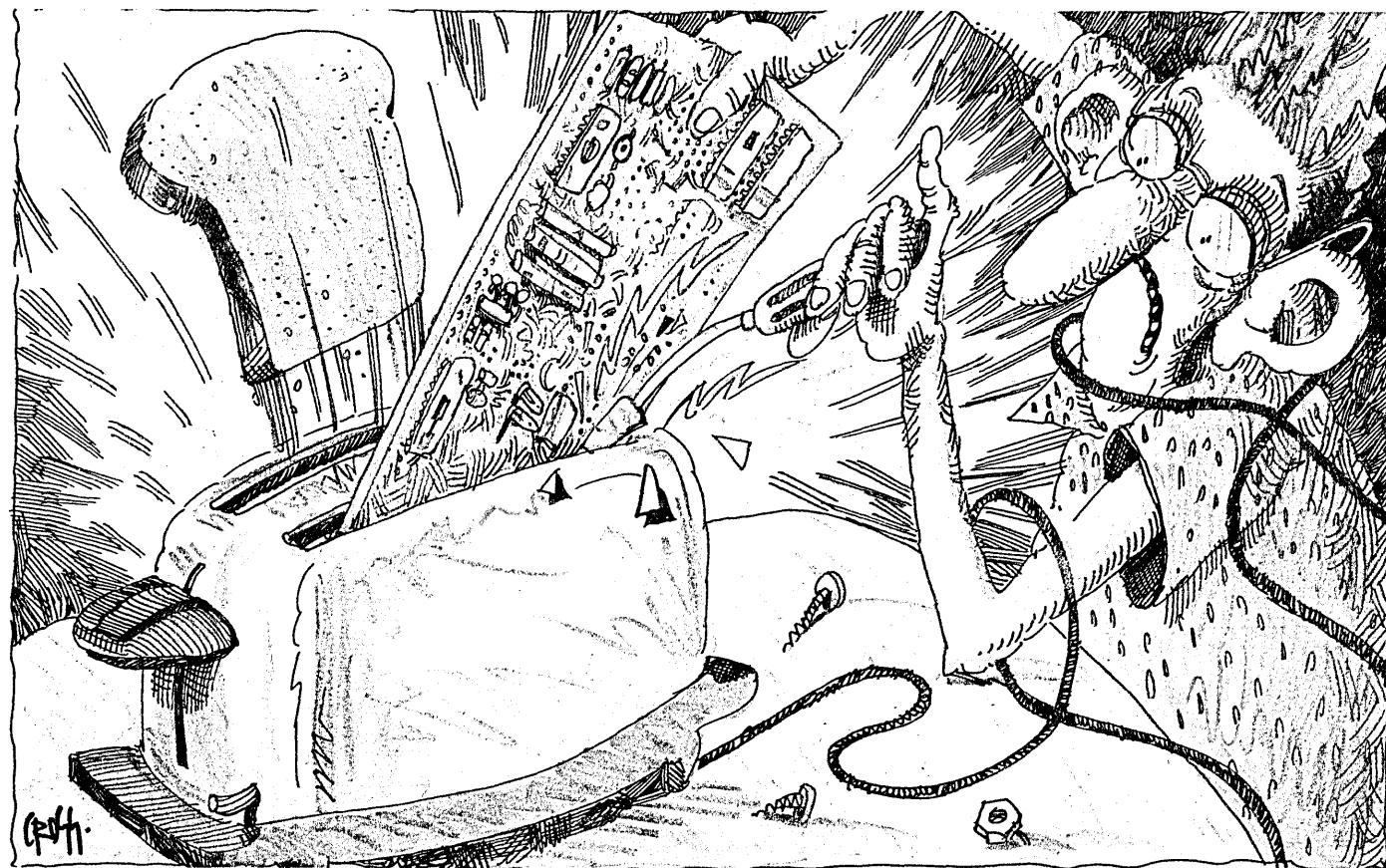
	VEDIT	BRIEF 2.10	Norton 1.3	QEdit 2.07
Pull-Down menus	Yes	No	No	Yes
Pop-Up ASCII table	Yes	No	No	No
Keystroke macros	100 +	1	No	100 +
Regular Expressions	Yes	Yes	No	No
"Cut and Paste" buffers	36	1	1	100
Text (book) markers	10	10	No	No
Undo keystroke by keystroke	Yes	Yes	No	No
Undo line by line	Yes	No	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

Embedding An XT Motherboard

*Putting Together A Complete Development System
Without Trading In Your House*



There are fancy chips and fancy boards. But however you do it, there's one big impediment to developing an embedded system—it's putting together a complete set of tools and then learning how to use them.

And ICE: you know, in-circuit emulation. Boy, that's wonderful when you're debugging code on a deaf-mute board (especially when it pretends to be brain dead). So, when Gene offered to do this article, what could I say? (ICE is nice.)

My experience with embedded systems goes way back. Most of my early recollection is clouded by the memory of intense pain. The pain centered on the following questions. Why do little low cost single chip microcontrollers require a large high

cost in-circuit emulator to make them function? Why does a microcontroller that costs \$5 for the mask programmed part cost \$50 for the EPROM version?

I spent a lot of my early days in engineering wrestling with these questions. I also had a lot more courage than brains in those days. My first experience was with a Z80 single board computer on the STD bus (it was memorable). I hoped the Navy would use the machine to test jet engines.

This was around 1978, and I also had a TRS-80 model 1. I used to say TRS-80 model 1 computer, but I know better now. The TRS-80 had a workable assembler, but the EPROM programmer supplied by the Navy couldn't talk to it. It couldn't talk to anything. It had a hex keypad. Humans cursed it.

In-circuit emulators for the Z80 cost a

fortune then, a small fortune now. The project requirements for this device changed as it developed (this is an advanced engineering management technique called "the moving target"). The device would have to calculate in floating point. I wrote a four-function floating point arithmetic package in Z80 assembly. What a pain.

Next I used an 8048 single chip microcontroller in a temperature controller. I only needed integer math for this beast. The cost of an 8048 (the version they program at the factory) then was about \$5, but unless you wanted at least 5000 pieces, Intel wouldn't consider the program mask.

The version with internal EPROM was about \$50, so I designed a card to supply external EPROM and RAM (with an address latch). It looked like a single

By Gene Toner

IDE, Inc.
P.O. Box 69
Fountainville, PA 18923

board computer when I finished. The worst part was that I still couldn't afford an in-circuit emulator for debugging the code.

There's a neat little circuit you can build around the 8048 to cause it to single step through a program. You can monitor the state of the address bus with LEDs. You can have the 8048 write intermediate results to an output port with more LEDs. Then you can single step your way through the bugs, burn new EPROMs, single step, burn new EPROMs, single step....

The Prescription For Embedded Pain

These were just two early experiences. I also wrote a cross assembler for the National Semiconductor 8073 microprocessor in TRS-80 model 1 BASIC. I did this to avoid hand assembly.

Interestingly, this processor had an on-board integer BASIC interpreter and interfaced to a terminal. National designed a board with this processor, EPROM, RAM, I/O, and one of the EPROM sockets would program 2716s!

It was a dream come true until my BASIC program grew beyond about 20 lines. You needed a calendar to time the execution. They threw in a BASIC instruction that jumped to machine language routines, but no assembler. I wrote the assembler.

You might think I would have learned, but I still do embedded controllers. Fortunately, I now have real solutions for development.

What is the ideal platform for an embedded controller? It should be inexpensive, flexible, expandable, available, and reliable. Plus, there should be great software support, such as assemblers, debuggers, and high level languages, all inexpensive. We don't want to have to sell the kids to control the temperature in the aquarium (hmmmm...).

The answer, of course, is the XT! No,

You might think I would have learned, but I still do embedded controllers. Fortunately, I now have real solutions for development.

not the whole XT, just the motherboard. Embed the mother to handle any control task you want. *Editor's note:* Micro C is not always a safe place to make offhand comments about the fairer sex.

A 12 MHz motherboard costs about \$80 these days, an 8 MHz unit about \$65, usually less without the BIOS (more about this later). These boards have eight expansion slots for adding anything and power supplies are only \$35.

What really makes this board attractive, however, is the C-THRU-ROM by DATALITE. This software package does things that \$20,000 in-circuit emulators just dream about.

It lets you write your programs in Microsoft C 5.X or Turbo C 2.X, cross load the executable and the C symbol table to the target system, and DEBUG ON THE TARGET SYSTEM IN C (using a debugging environment very similar to Microsoft's CodeView). Finally, it lets you burn the whole works into EPROM, all for less than \$500.

XT Motherboards

The usual XT motherboard will hold up to 640K of parity checked RAM and 40K of ROM. This ROM space is usually mapped as 8K of BIOS residing at paragraph FE00 and 32K of ROM BASIC at F600. Both of these ROM spaces are available for embedded programs.

In most embedded applications, 640K is overkill since RAM is usually only used for variables and stack. Other applications, such as data loggers and pattern recognizers, make good use of everything they can get. Also, RAM is a

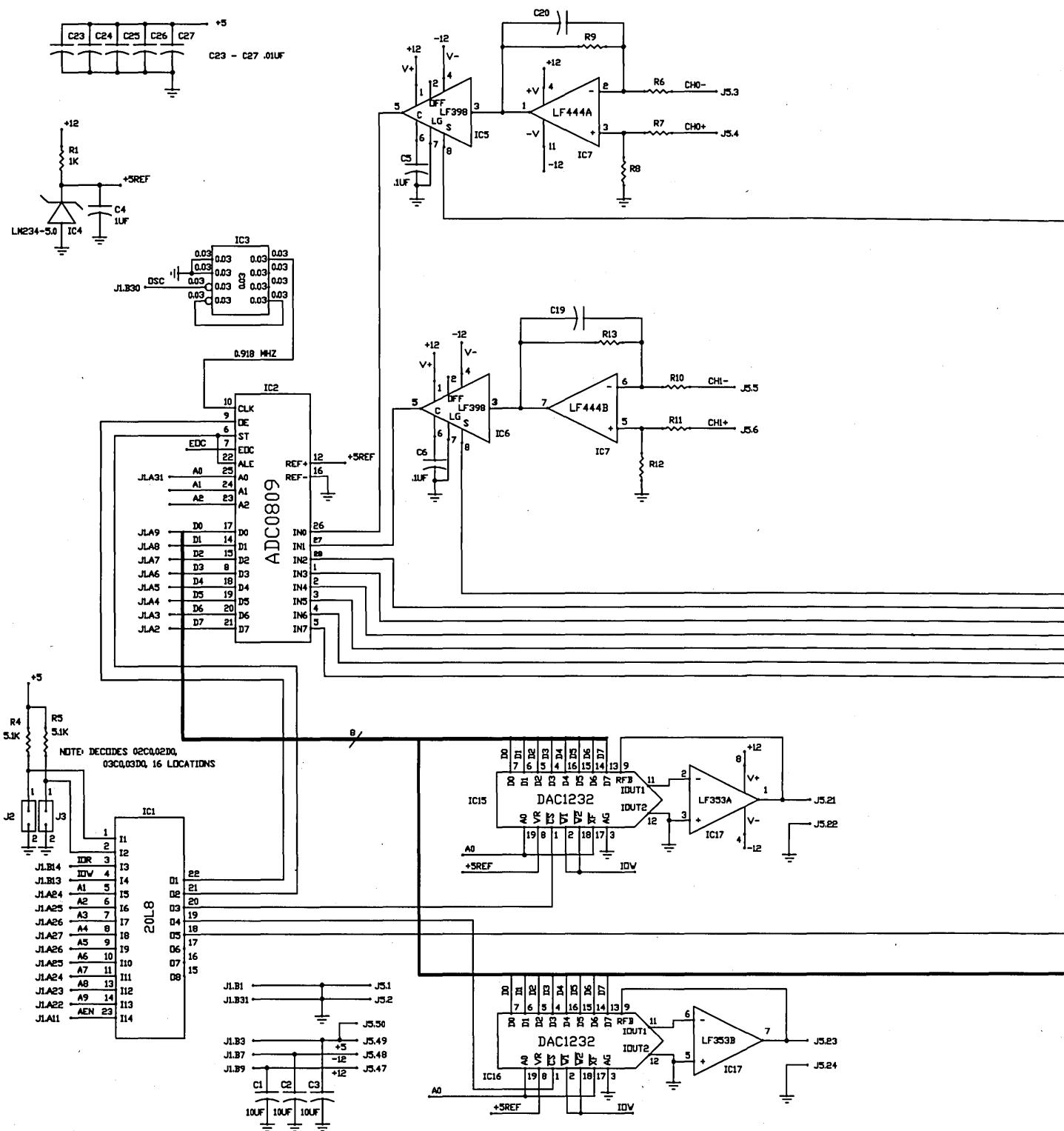
Text Continued on page 12

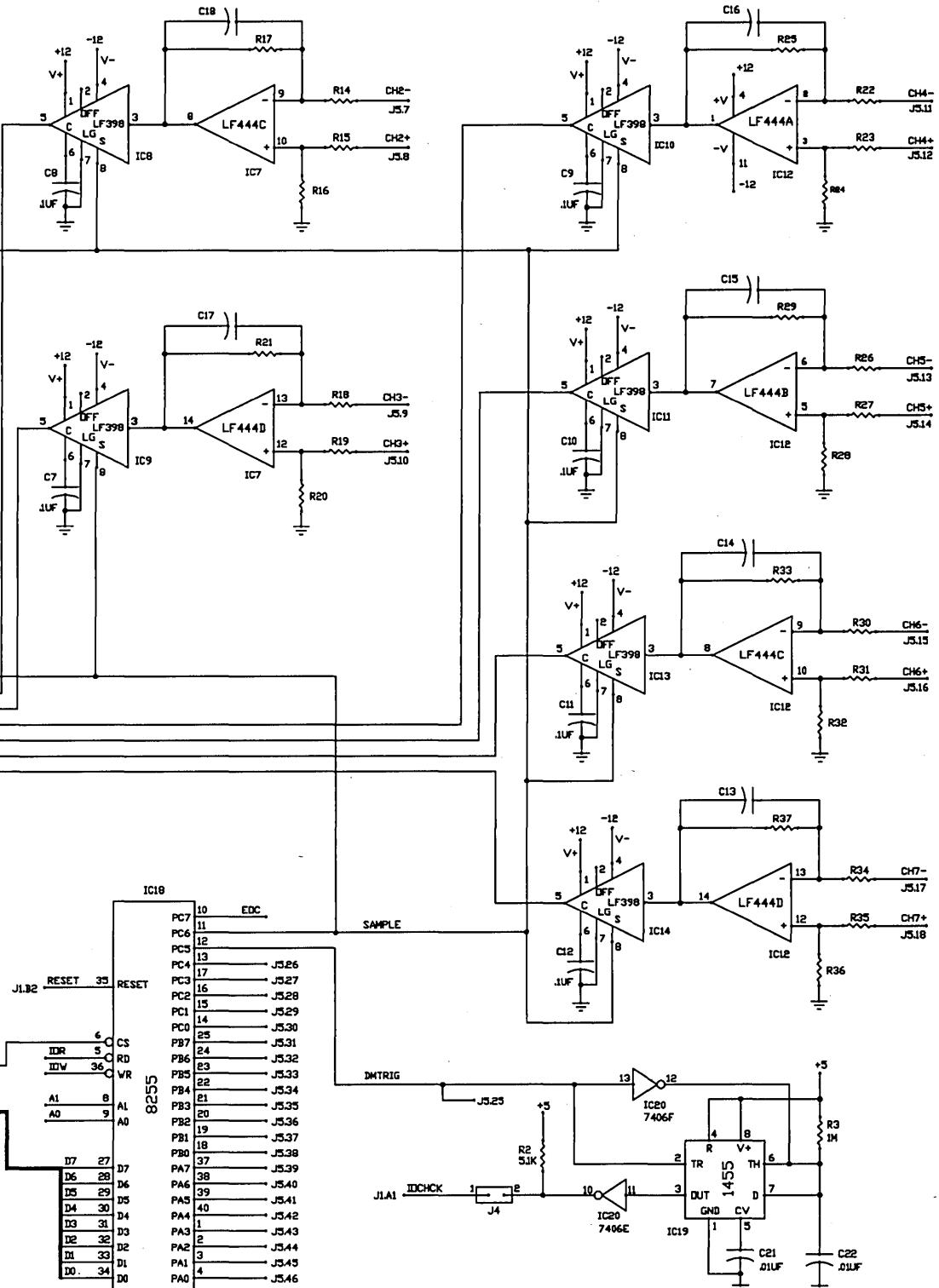
Figure 1—Motherboard I/O Map

I/O DECODED	I/O USED	FUNCTION
0000h - 001Fh	16	DMA controller
0020h - 003Fh	2	Interrupt controller
0040h - 005Fh	4	Timer counter
0060h - 007Fh	4	PPI (8255) chip
0080h - 009Fh	4	DMA page registers
00A0h - 00BFh	1	NMI mask bit

♦ ♦ ♦

Figure 2—PC Card with a Dead-man Timer





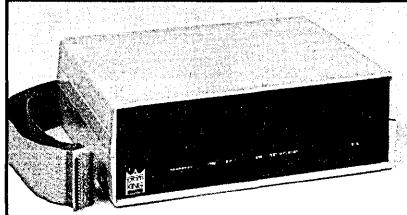
wonderful place to develop and debug programs.

Along with the memory, other onboard resources include four DMA channels, one dedicated to DRAM refresh. There are three counter/timer channels—one dedicated to requesting refresh cycles, one to interrupting the processor (used in the PC for time of day with a link interrupt), and one to work the speaker.

There is also an eight-level priority interrupt controller. Level 0 (highest priority) connects to the counter/timer channel 0, and channel 1 connects to the keyboard adapter circuits. The other six interrupts are bussed to the adapter cards.

The keyboard adapter circuits provide an externally clocked serial input to the motherboard, with an interrupt generated at the conclusion of a received byte. There is also an eight-position DIP switch mapped into the I/O space to allow for static input to a program; normally the board reads this switch at power up. Figure 1 presents the I/O map of the devices on the motherboard.

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 a real EPROM
- Battery backup

\$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

Figure 3—Function Offsets (from BASE)

OFFSET (HEX)	READ FUNCTION	WRITE FUNCTION
0	CH0 CONVERSION DATA	SET A/D CH0 FOR INPUT
1	CH1 CONVERSION DATA	SET A/D CH1 FOR INPUT
2	CH2 CONVERSION DATA	SET A/D CH2 FOR INPUT
3	CH3 CONVERSION DATA	SET A/D CH3 FOR INPUT
4	CH4 CONVERSION DATA	SET A/D CH4 FOR INPUT
5	CH5 CONVERSION DATA	SET A/D CH5 FOR INPUT
6	CH6 CONVERSION DATA	SET A/D CH6 FOR INPUT
7	CH7 CONVERSION DATA	SET A/D CH7 FOR INPUT
8	NO FUNCTION	LOWER 8 BITS D/A 0 VALUE
9	NO FUNCTION	UPPER 4 BITS D/A 0 VALUE
A	NO FUNCTION	LOWER 8 BITS D/A 1 VALUE
B	NO FUNCTION	UPPER 4 BITS D/A 1 VALUE
C	8255 A PORT READ DATA	8255 A PORT WRITE DATA
D	8255 B PORT READ DATA	8255 B PORT WRITE DATA
E	8255 C PORT READ DATA	8255 C PORT WRITE DATA
F	NO FUNCTION	8255 CONTROL REGISTER

♦ ♦ ♦

Figure 4—C Code to Monitor EOC

```
get_channel(chan)

char chan;

{
    int eoc;
    int result;

    /* start the conversion process */
    outp(BASE+chan,0); /* select the channel, start */

    /* monitor EOC */
    eoc=0;
    while(!eoc) /* loop executes until EOC is high */
        eoc=inp(BASE+0x0e) & 0x80; /* get the EOC value */
    result=inp(BASE+chan); /* get the converted value */
    return(result);
}
```

♦ ♦ ♦

C-THRU-ROM

C-THRU-ROM lets you write C code, compile it with Microsoft or Turbo, dump it to the target XT via a serial port, and debug it (remotely). The only thing you can't do is use DOS calls. Microsoft tells you which of its library routines use DOS calls in the "Writing Programs for Read-Only Memory" section of its manual.

After the usual compile and link, C-THRU-ROM starts to work. The LOC command in C-THRU then locates the resulting program.

Located? Besides the program code and data, an EXE file contains information on program memory requirements, segment modifications, and the start address. DOS resides in low memory and loads programs right above itself. Programs start at the first byte of code and end with the last byte of data.

In our embedded system, we (usually) don't have things so well lined up. ROM gets mapped in high memory space so it can pick up the 8088's power-on jump vector (paragraph FFFFh). RAM, however, must start at paragraph 0000h to support the 256 four-byte interrupt vectors. Fortunately, C-THRU's LOC can handle a memory map with holes in it.

Once located, the program communicates via the serial port to the target system's kernel for debugging. Kernel? Serial port? This is the only hard part of the whole package: getting the C-THRU-ROM kernel configured for, and running on, the target system. C-THRU supplies the kernel source, but you must configure it for your own motherboard.

In an XT motherboard, the kernel must initialize all the ports, set up interrupt vectors in low RAM, and set up the DMA channel and timer for DRAM refresh. For all this, you should have a good idea what's going on inside the XT's hardware. Once you have the kernel running, however, the debugging environment is grand.

Even though the target system has, say, no drive, no monitor, no keyboard, no nothing (but a serial port), you can download programs from the host and debug them on the target.

So the program resides on the target system, and the C-THRU-ROM program RDEB runs on the host PC. You're stepping through C code (or assembly or both) one line at a time, watching variables change. You can also set breakpoints on C symbols. The environment is a lot like CodeView.

STAGE#1 - Action Diagram Editor

STAGE#1 is a full screen editor that automates the generation and editing of action diagrams, the most versatile diagramming method available today. Action diagrams can be used to define software structure from the top-most overview down to the lowest level pseudo code. No other diagramming method has that capability.

More than 70 commands provide the familiar word processor functions, plus a few unique features. For example:

CONVERTS ACTION DIAGRAMS;

```
*Print_Character_Set
  For line 0 to 7
    Put leading spaces in the output string
    Put the line number and ':' in the output string
    Calculate offset (line * 32)
    If line is not 0 or 4
      For count equals 0 to 31
        Character value is count plus offset
        Add character and space to the output string
      /* Endfor */
    /* Else */
    /* Add 'Control characters, can't print them!' to the */
    /* output string */
    /* Endifelse */
    /* Display the output string */
    /* Add CR and LF to the output string */
    /* Print the output string */
  /* Endfor */
  /* Print a formfeed */
```

INTO COMMENT LINES!

STAGE#1 also doubles as a normal text editor for entering program code. Direct access to DOS allows trial compilation or assembly without exiting the editor.

System requirements: IBM PC/XT/AT, PS/2 or compatible; PC or MS-DOS, 2.1 or higher; minimum of 256KB of memory; CGA, EGA, VGA or monochrome monitor; two disk drives; printer with either the full ASCII or the IBM graphics (line-draw) character set. A plotter is not required.

INTRODUCTORY PRICE \$159.00

Price: \$189.00 (US), includes shipping (UPS ground) within the continental US. Washington residents add 7.8% sales tax. Educational discounts and site licensing are available. NOT copy protected. Specify 3 1/2 or 5 1/4 inch disk.

Why settle for a program that will produce only flowcharts and organization charts when you can have much more?



SIMPLY POWERFUL
Mt. St. Helens Software
P.O. Box 3319
Pasco, Washington 99302-3319



For orders or
information:
(509) 547-2582
8 AM - 5 PM (PST)

Mt. St. Helens Software, Simply Powerful and STAGE#1 are trademarks of Mt. St. Helens Software. Other brand and product names are trademarks of their respective holders.

After you've finished debugging, you simply link the C program with C-THRU-ROM's startup code (which replaces the normal C startup code) and produce an Intel hex format file. Then send the file to your EPROM programmer. (My Taiwanese programmer loves Intel hex.)

There are some drawbacks to the package, the most limiting being C-THRU's inability to debug interrupt routines. So, we usually write polling loops for initial debugging and then add interrupts later.

Talking To The World

After building several embedded systems, I have discovered that microprocessor systems, used as controllers (read that as left alone in the real world), sometimes jump out of their programs into what is affectionately known as outer space.

Editor's note: Apparently, this is the part of outer space where daisies grow.

If it's really important to keep the system running, you should include a device known as a dead-man timer in the design. If a dead-man timer isn't serviced at a certain interval, it resets the processor.

The service to the dead-man must be from the main loop of the program, not from an interrupt routine. It's possible for the processor to be totally lost but still be capable of servicing interrupts.

Since I couldn't find any PC cards containing a dead-man, I built my own. Because it would be silly to put together a whole board with one or two tiny chips, I added a little something: 8 channels of 8-bit 10 KHz A/D conversion, 2 channels of 8- or 12-bit D/A, and 22 digital I/O lines. See Figure 2.

The heart of the A/D is the National Semiconductor ADC0809 eight channel A/D converter. This is a successive approximation eight-bit converter with an internal eight-channel multiplexer. In this design it will perform a conversion in about 100 microseconds (IC2).

The 20L10 PAL handles address decoding, supplying all the read and write strobes. The decode is for 16 contiguous addresses at the starting points indicated on the schematic. The D/A function (ICs 15 and 16) generates analog signals for the real world.

The schematic indicates the part number for the 12-bit converters, but the 8-bit parts (DAC830) are pin for pin compatible. The 8255 programmable peripheral

Figure 5—Assembly Routine to Collect A/D Data From Channel 5

```

DATA_SEG      SEGMENT AT 0040H
    RESULT     DB 2
DATA_SEG      ENDS

STACK_SEG     SEGMENT PARA STACK 'STACK' AT 0800H
    DB 100 DUP(?)
STACK_SEG     ENDS

CODE_SEG      SEGMENT AT 0F600H
    ASSUME CS:CODE_SEG

MOV AX,DATA_SEG ;INITIALIZE THE DATA SEG
MOV DS,AX

MOV AX,STACK_SEG ;INITIALIZE THE STACK
MOV SS,AX
MOV SP,0H

;INITIALIZE EXTRA IF USED
ASSUME DS:DATA_SEG,SS:STACK_SEG

START:
MOV AL,0
MOV DX,BASE+5 ;CONVERT CH 5
OUT DX,AL ;START
MOV DX,BASE+0FH ;POINT TO C PORT
LOOP0: IN AL,DX ;GET THE CONDITION
AND 10000000B ;RESET UNUSED BITS
JZ LOOP0 ;LOOP UNTIL EOC IS HIGH

MOV DX,BASE+5 ;POINT TO THE DATA
XOR AX,AX ;CLEAR
IN AL,DX ;GET THE DATA
MOV AX,DS:RESULT ;STORE IT

```

♦ ♦ ♦

interface chip (IC18) supplies the 24 I/O lines. I use two lines on the board so there are 22 available through the I/O connector.

The dead-man timer (IC19) is a CMOS 555 timer configured as a resetable monostable multivibrator. When it times out, it makes the I/O channel check bus line active. I installed a jumper on the line to disable this function during debugging.

All real world inputs and outputs enter the card through the 50 pin dual row header. I use FET input differential op amps to buffer all the analog inputs. Differential inputs are much more immune to noise than single-ended inputs.

I included capacitors in the amplifiers' feedback loops for filtering, if necessary. Following the op amps, I added sample and hold amplifiers so our slow A/D

converter won't be confused by rapidly changing signals. Don't let inputs to the ADC0809 exceed the 0-5 volt range.

A Cheap Alternative

The alternative to using C-THRU-ROM for this type of development is to do all the development and debugging on a PC. This is feasible and could be even more so.

Replace the standard BIOS ROM on the PC with a ROM program that initializes the motherboard and then jumps to the highest paragraph of the next lower ROM. This would allow development to start in the "crash and burn" type environment.

So how do we do it? Let's look at doing an A/D conversion using C for the C-THRU environment and assembly lan-

guage for the jump ROM environment. First, disable the dead-man timer on the card by removing the jumper J4 for development. The card has the following address offset mapping, the base address in the processor's I/O space being 02C0, 02D0, 03C0 or 03D0, depending on the jumper setting. (See Figure 3).

To start a conversion, in A/D channel 5, for instance, just write data to the address of channel 5 (the board address plus an offset of 5). It doesn't matter what you write.

After the write, the EOC (end of conversion) line will go low. (Monitor this line by reading the high bit (bit 7) of port C of the 8255.) My program monitors this line, waiting for it to go high, indicating the conversion's done.

A C program to do this might look like Figure 4. Note that BASE is the base address of the board.

While this code would work fine with the C-THRU package, you'd need a locator program to justify the compiler's segment references if you used this with a simple jump-start ROM.

To run in a jump-start type environment (without a locator program) would require that the C compiler handle the location, or you'd have to use assembly language. In assembly, of course, you have complete control of the segments and jump locations.

In Figure 5, you'll see my assembly language version of the C program in Figure 4. I wrote it for a PC motherboard with 64K of RAM starting at 00000h, the jump start ROM at FE000h, and the user ROM at F6000h.

The last thing the jump start ROM will do after initializing the peripherals, the RAM refresh, and the interrupt vector table is to far jump to FDFF0h, the last paragraph of the user ROM. Here you must place a far jump to your code's entry point, *with interrupts disabled*.

The Offerings

IDEDEC, Inc., offers the following packages to support the development of embedded systems using the PC-XT motherboard as the platform for stand-alone applications:

C-THRU-ROM by DATALITE

List \$500—IDEC \$400

Includes fully developed kernel for standalone PC

A/D+D/A+I/O+DEADMAN ADAPTER

A&T List \$199—IDEC \$139

Bare board + PAL

IDEC \$49

Both of the above

List \$699—IDEC \$500

Initialization + Jump ROM for PC

List \$49—IDEC \$25

for non C-THRU ROM development

◆ ◆ ◆



This add was prepared on a laser printer using an IDEC Supervision captured Tiff image, Xerox Ventura Publisher, ZSoft's PC Paintbrush IV, and Application Techniques' Pizazz Plus. Call:

VISA

215-249-0673

Master
Card

ASMFLOW \$99.95

AT LAST!

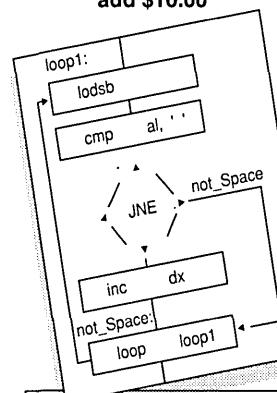
YOU CAN STEP BACK AND TAKE A LOOK AT YOUR CODE

FLOW CHART AND ANALYZE YOUR ASSEMBLY LANGUAGE SOURCE CODE

- Flow Charts
- Tree Diagrams
- Stack Sizing
- Register Analysis
- CPU Timing Analysis
- Procedural X-Reference
- 8088/87 to 80386/387
- Context-Sensitive Help
- Menu/Batch/Command line Operation
- MASM 5.1 Compatible

QUANTASM CORPORATION

19855 Stevens Creek Blvd, Suite 154
Cupertino, CA 95014
(408) 244-6826



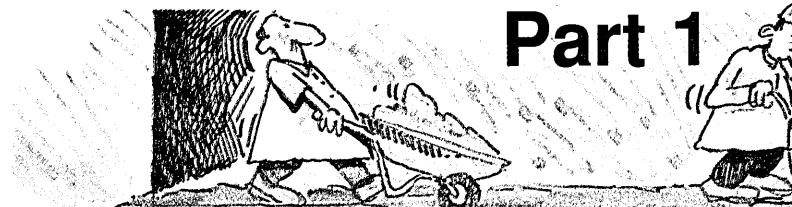
VISA • MC 30 - Day Money Back Guarantee

Reader Service Number 180

Reader Service Number 139

MICRO CORNUCOPIA, #51, Jan-Feb 1990 15

Implementing A Neural Network In C: Part 1



Russ and Roy were hits at SOG East. After their presentation, I found myself in a motel room cornered by these two neural addicts. I must say my mind is still working on the inputs—back propagation, I think it's called.

All seriousness aside, neural nets are very possibly the most important new technology I've seen. Once you're comfortable with the concepts, things really open up. In many ways this is everything that AI had promised, and more.

Neural networks are hot! For many good reasons. We now use applications which include embedded neural networks in manufacturing, teaching, aerospace engineering, financial advising, circuit board analysis, and more.

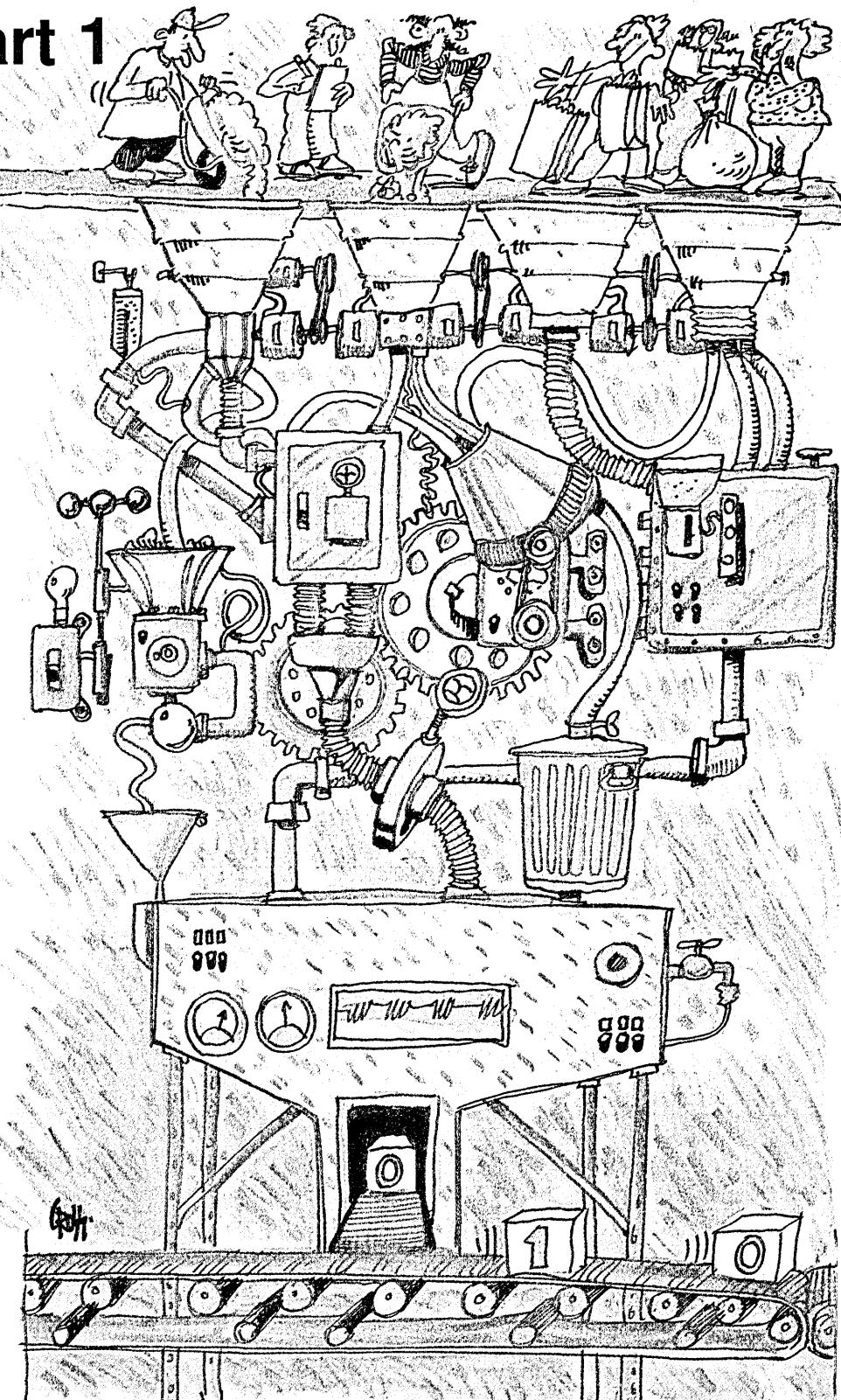
Contrary to popular opinion, you don't need a super parallel processing computer, a Sun workstation, or a Ph.D. in neurobiology to use a neural network. A PC and the C code for the neural network tool (NNT) we develop in this article will get you going.

This issue we'll briefly introduce NNTs, explain how they work, and show you all the equations and code you need to create one. In Part 2, we'll show you how to train and use our NNT. Meantime, download our NNT (including C source code and .EXEs) from the Micro C BBS and tell us what you think about this fascinating technology.

Editor's note: We've excerpted Russ & Roy's article(s) from their forthcoming book on implementing and using neural networks, due in 1990 from Academic Press.

Intro

A neural network, in short, looks for patterns in a set of sample cases (a training set), learns from these samples, and



classifies new examples based on the patterns it's learned.

Neural networks are particularly useful for solving problems that use imprecise or fuzzy input patterns. They're particularly inappropriate for solving problems requiring precise calculations. You'll probably never successfully balance your checkbook with one. (Of course your checkbook might be unbalanceable, anyway.)

We broadly characterize NNTs using three criteria:

(1) Network architecture: in general, how slabs (made up of layers) are interconnected and how they receive input and output. More on slabs and layers later. For now, look over Figure 1: a typical NNT feedforward architecture. Feed-forward means that the information flow is always in one direction, from input to output, without any feedback.

(2) The type of transfer function we use for processing elements (PEs or nodes or neurodes) in the network. That is, what function describes the output of an element given its input?

(3) The type of learning paradigm we use to train the network.

Think of these three criteria (or categories) as the top level attributes of an NNT. As we'll show you, you can't always vary these attributes independently. Certain architectures, for example, preclude certain learning paradigms.

Neural biological structure (as we currently understand it) and the implementation or representation of this structure in NNTs differ significantly. Let's face it—the brain is much more complex than anything we can write (yet!). Loosely, we say that a processing element (PE) or unit in an NNT is roughly analogous to a biological neuron.

In a typical NNT, each processing element (PE) connects to other processing elements. Each of these connections can

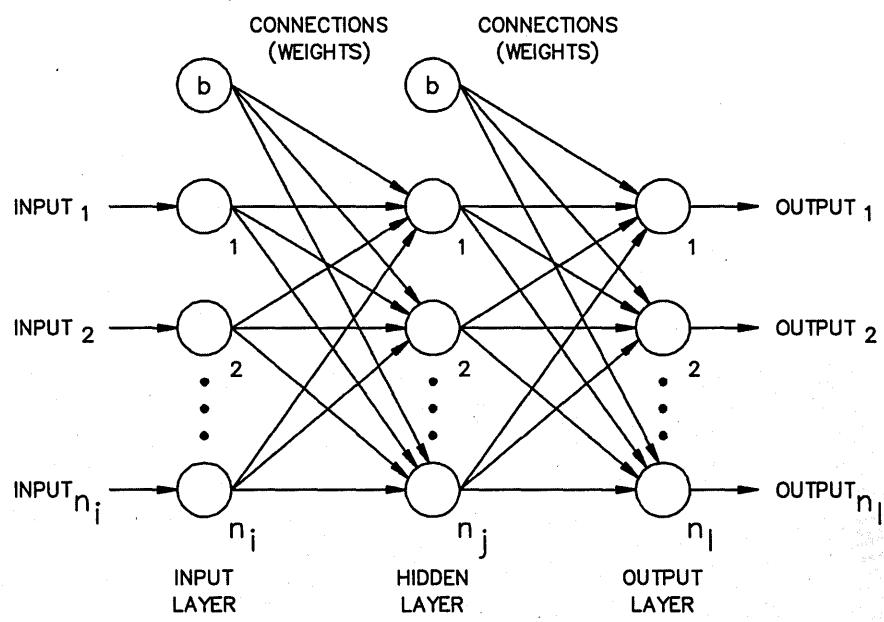
be positively or negatively weighted. Information about the state of a PE passes on to the PE(s) connected to it based on these weights and the network's transfer function.

Neurons in biological neural networks typically cycle in about 10-100 milliseconds. The clock frequency in an 80286 or 80386-based microcomputer is generally 10-30 MHz, which results in a basic cycle of 0.03-0.10 microseconds.

Even taking into account the number of multiply accumulate operations needed to calculate and propagate a new value for a PE (typically 10-100), the basic cycle time for an individual PE is still only about 1-10 microseconds, orders of magnitude faster than our brain! But speed is deceiving; our brain processes

Neural networks are particularly useful for solving problems that use imprecise or fuzzy input patterns.

Figure 1—Back-propagation Network Structure



many kinds of information faster.

Neural networks are hot, but they're far from perfected. At this juncture several commercial products show promise, but even the best of these will no doubt change (for the better) as we learn more about our brains and more about computers.

But enough general introduction. Let's create an NNT, beginning with a few definitions for the back-propagation model, the architecture we'll use.

The Back-Propagation Model

Although a back-propagation neural network model is generally considered a good one, it has no "standard" definition. Neural network experts agree mostly on the algorithms they use to describe network training and operation, but don't agree how these algorithms should be implemented.

We'll try to simplify our implementation by describing the back-propagation (BP) model (as we understand it) in some detail. We'll examine each of its elements and describe how these elements combine to form the BP topology.

Figure 1 (remember you looked before) shows a simple, three-layer, BP model.

- Each circle represents a processing element (PE).
- Each arrow represents an interconnection (synapse) and its associated weight.
- The PEs with the letter "b" inside are called bias nodes.

In this article (and our book), subscripted lower case letters represent the attributes of individual PEs (nodes) and of individual connections.

- the letter "i" represents an input,
- "o" represents an output,
- "w" represents a connection weight,
- "n" represents the number of nodes in a layer.
- The subscripts "i,j,l" refer to the input, hidden, and output layers, respectively.

(If you use more than one hidden layer, the subscript "k" represents it.)

For example, " i_i " is the input to an input layer PE, " o_j " is the output of a hidden layer PE, and " n_l " is the number of PEs in the output layer.

Bold lower case letters represent vectors. For example, " \mathbf{i}_i " represents the input vector to the input layer, made up of all the individual inputs. " \mathbf{o}_j " repre-

sents the output vector of the output layer.

We often work with a combination of an input vector and its associated output vector. This combination of an input and its associated output comprises a "pattern vector," represented by a " \mathbf{p} ".

We list the input part first, then the output. Typically we divide all our patterns into two categories or sets: a training set and a testing set. The subscripts "r,s" are associated with training and testing, respectively.

N

eural networks
are hot, but they're
far from perfected ...
even the best of
these will no doubt
change ... as we
learn more about
our brains and more
about computers.

So, for example, " \mathbf{p}_r " is a training pattern and " \mathbf{p}_s " is a testing pattern. In both cases, in the representation of the vector (for example, in the pattern files for the NNT), the output components follow the input components.

Connection weights require two subscripts that represent the sending and receiving layers. For example, the weight of the connection from an input PE to a hidden PE is " w_{hi} ." Note that the receiving PE layer is the first subscript, and the sending PE layer is the second. While this may seem somewhat counter-intuitive to you, it's the generally accepted way to represent weights, and corresponds to the matrix notation which sometimes represent weights.

We represent matrices by bold capital letters. For example, " \mathbf{W}_{ji} " represents the matrix of connection weights to the hidden layer (from the input layer).

Don't despair (if you're despairing); we'll use vectors and matrix notation sparingly.

We'll also use three coefficients later:

- the learning coefficient, eta (η),
- the momentum factor, alpha (α),
- and the error term, delta (δ).

Later, we'll describe each of the network elements. In Part 2 we'll describe the operation and training of the BP network of Figure 1, step by step. Let's examine now how we present input to the network.

Network Input

On the left of Figure 1, note the inputs entering the network via the input layer (a layer of processing nodes). These inputs can be a set of raw data, a set of parameters, or whatever we've chosen to represent as a pattern.

For our BP NNT, the value of each input can take on any value between 0 and 1. That is, the input values are analog and are normalized between the values 0 and 1. The fact that we can use analog inputs adds significant flexibility to our NNT.

Does the normalization between 0 and 1 constrain us in any way? Probably not, at least not usually. Whenever we deal with a real-life computer system that's receiving input, we're always limited to some extent by the size of the number we can put in. As long as the resolution of our input data doesn't get lost in the normalization process, we're all right.

In our implementation of the BP NNT, we use standard floating point variables, called "float" in C. Floats are 32 bits long, using 24 bits for a value, and 8 bits for an exponent.

We therefore have a resolution of about 1 part in 16 million, or, stated another way, resolution to 7 decimal places. If your data has 7 significant figures or less, you'll be okay. We haven't found this at all limiting. Even input data from a 16-bit analog to digital (A/D) converter requires a little less than 5 digits of resolution. Most of the applications we've seen require 3 to 5 digits of resolution.

Normalizing our input patterns can provide us with a tool for preprocessing our data in various ways. You can normalize the data by considering all the " n " inputs together, normalize each channel separately, or normalize groups of channels in some way that makes sense. In some cases, the way you choose to normalize the inputs can affect the performance of the NNT, so this is one place you can really experiment.

If your inputs all consist of raw data points, you'll probably normalize all the channels together. If the inputs consist of parameters, you may normalize each channel separately, or normalize channels that represent similar kinds of parameters together. For example, if some of your parameter inputs represent amplitudes and some represent time intervals, you might normalize the amplitude channels as a group and the time channels as a group.

Feedforward Calculations

Now (assuming we have the input pattern represented by our normalized set of inputs) what happens at the input layer? Given that the values are normalized, the input PEs simply split the signal into multiple paths to the hidden layer PEs.

In other words, the output of each input layer PE is exactly equal to the input, and is in the range of 0 to 1. (Another way of looking at the input layer is that it normalizes input. Most NNT implementations normalize inputs before they're presented to the network.)

The input signals are then sent to the PEs of the hidden layer via the connections from the input layer. A weight is associated with each connection. Note that each PE of the input layer connects to every PE of the hidden layer. Likewise, each PE of the hidden layer connects to every PE of the output layer.

Also note that each connection, and all data flow, goes from left to right in Figure 1. This one-way flow is called "feedforward." There are no feedback loops, even from a unit to itself, in a feed-forward network. All standard back propagation implementations, including our BP NNT, are feedforward.

The way the net input to a PE is calculated, and the way the PE calculates its output as a function of its net input, depends on the type of transfer function we're using in the NNT. Our implementation (and most BP NNTs today) uses an additive sigmoid PE.

We'll now show you the mathematical equations that describe the training and testing/running modes of a BP NNT. We won't derive or prove them (sparing you those mathematics), but you can find all the proofs you want in Rumelhart and McClelland (Vol. 1). Much of what you'll want to know is in Chapter 8, the chapter on internal representations. (See References.)

The signal presented to a hidden layer

PE in the network of Figure 1 due to one single connection is just the output value of the input node (the same as the input of the input node) times the value of the connection weight.

The net input to a hidden PE is the sum of the values for all connections coming into the PE, as described in Equation 1. Note that this includes the input from the node we call the "bias node," which we assume always has an output of 1, and which we treat otherwise as

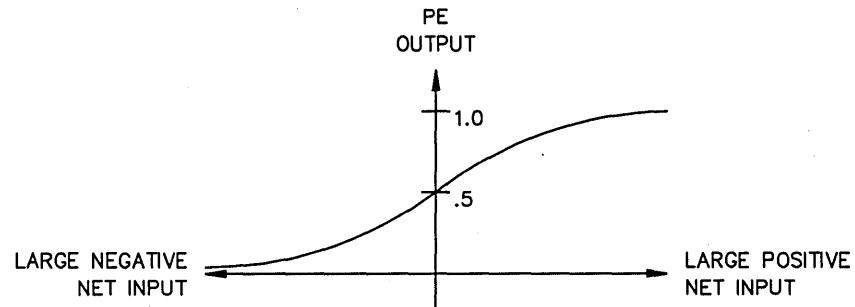
any other node. We'll say more about the bias node later.

$$i_j = \sum_i w_{ji} o_j \quad \text{Equation 1}$$

Equation 2 describes the output of a hidden node as a function of its net input. This is the sigmoid function to which we've been referring. The sigmoid function is illustrated in Figure 2.

Text Continued on page 24

Figure 2—Sigmoid Transfer Function Used in Back-propagation NNT



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 NEW! 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. \$195



Need Special Effects, but caught in a GRASP?

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! \$99



Blazing Graphics Text

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. \$149

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.

No Royalties! 30-day Money Back Guarantee.
VISA/MC/AMEX/COD/PO accepted.



MICRO PROGRAMMING

11315 Meadow Lake • Houston, Texas 77077 • (713) 870-0737

1-800-227-0918

Figure 3—BATCHNET.C

```

/*
Generic back-propagation neural network
(c) 1988, 1989 R.W.Dobbins and R.C.Eberhart
All Rights Reserved
$Revision: 1.1 $ $Date: 21 Sep 1989 11:35:06 $
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <conio.h>
#include <cctype.h>
#include <string.h>

#define ESC 27
#define ERRORLEVEL 0.04
#define ITEMS 8

/* typedefs & prototypes for dynamic storage ...*/
typedef float *PFLOAT; /*... of arrays */
typedef PFLOAT VECTOR;
typedef PFLOAT *MATRIX;

void VectorAllocate(VECTOR *vector, int nCols);
void AllocateCols(PFLOAT matrix[], int nRows,
                  int nCols);
void MatrixAllocate(MATRIX *pmatrix, int nRows,
                    int nCols);
void MatrixFree(MATRIX matrix, int nRows);

/* define storage for net layers */
/* Arrays for inputs, outputs, deltas, weights &
   targets */

MATRIX out0; /* input layer */
MATRIX out1; /* hidden layer */
MATRIX delta1; /* delta at hidden layer */
MATRIX delw1; /*change in weights input:hidden*/
MATRIX w1; /* weights input:hidden */
MATRIX out2; /* output layer */
MATRIX delta2; /* delta at output layer */
MATRIX delw2; /*change in weights hidden:output*/
MATRIX w2; /* weights hidden:output */
MATRIX target; /* target output */
VECTOR PatternID; /* ID for each pattern */

void main(int argc, char *argv[])
{
    float eta = 0.15, /* default learning rate */
          alpha=0.075; /*default momentum factor*/
    int nReportErrors=100; /*err report frequency*/
    float ErrorLevel=ERRORLEVEL; /*OK error level*/
    char MonitorError=0; /*true=mon. err display*/
    float error; /* latest sum squared error val */
    register int h; /* index hidden layer */
    register int i; /* index input layer */
    register int j; /* index output layer */
}

```

```

int p, /* index pattern number */
q, /* index iteration number */
r, /* index run number */
nPatterns, /* # of patterns desired */
nInputNodes, /* # of input nodes */
nHiddenNodes, /* # of hidden nodes */
nOutputNodes, /* # of output nodes */
nIterations, /* # of iterations desired */
nRuns; /* # of runs (or input lines) */
FILE *fpRun, /* run file */
*fpPattern, /* source pattern input file */
*fpWeights, /* initial weight file */
*fpWeightsOut, /*final weight output file*/
*fpResults, /* results output file */
*fpError, /* error output file */
char szResults[66], /* various pathnames */
char szError[66];
char szPattern[66];
char szWeights[66];
char szWeightsOut[66];
char *progname = *argv; /* name of executable
                           DOS 3.x only */

/* read optional - arguments */
for (; argc > 1; argc--)
{
    char *arg = *++argv;

    if (*arg != '-')
        break;
    switch (*++arg)
    {
        case 'e':
            sscanf(++arg, "%d", &nReportErrors);
            break;
        case 'd':
            sscanf(++arg, "%f", &ErrorLevel);
            break;
        default: break;
    }
}
if (argc < 2)
{
    fprintf(stderr, "Usage: %s {-en -df}\n"
              "runfilename\n", progname);
    fprintf(stderr, " -en => report\n"
              " error every n iterations\n");
    fprintf(stderr, " -df => done if\n"
              " sum squared error < f\n");
    exit(1);
}
/* Open run file for reading */
if ((fpRun = fopen(*argv, "r")) == NULL)
{
    fprintf(stderr, "%s: can't open file %s\n",
            progname, *argv);
    exit(1);
}
/*Read line 1: (lines to read' from run file)*/

```

```

fscanf(fpRun, "%d", &nRuns);

/* beginning of work loop */
for (r = 0; r < nRuns; r++)
{
    /* read and parse run specification line; */
    fscanf(fpRun,
    "%s %s %s %s %d %d %d %d %f %f",
    szResults,      /* output results file */
    szError,        /* error output file */
    szPattern,      /* pattern input file */
    szWeights,      /* initial weights file */
    szWeightsOut,  /* final weights out file */
    &nPatterns,     /* # patterns to learn */
    &nIterations,   /* # iterations through data */
    &nInputNodes,   /* # input nodes */
    &nHiddenNodes,  /* # hidden nodes */
    &nOutputNodes,  /* # output nodes */
    &eta,           /* learning rate */
    &alpha);        /* momentum factor */
/* allocate dynamic storage for all data */
MatrixAllocate(&out0, nPatterns, nInputNodes);
MatrixAllocate(&out1, nPatterns, nHiddenNodes);
MatrixAllocate(&out2, nPatterns, nOutputNodes);
MatrixAllocate(&delta2, nPatterns, nOutputNodes);
MatrixAllocate(&delw2, nOutputNodes, nHiddenNodes + 1);
MatrixAllocate(&w2, nOutputNodes, nHiddenNodes + 1);
MatrixAllocate(&delta1, nPatterns, nHiddenNodes);
MatrixAllocate(&delw1, nHiddenNodes, nInputNodes + 1);
MatrixAllocate(&target, nPatterns, nOutputNodes);
VectorAllocate(&PatternID, nPatterns);
/* Read the initial weight matrices: */
if ((fpWeights=fopen(szWeights, "r"))==NULL)
{
    fprintf(stderr, "%s: can't open %s\n",
            progname, szWeights);
    exit(1);
}
/* read input:hidden weights */
for (h = 0; h < nHiddenNodes; h++)
    for (i = 0; i <= nInputNodes; i++)
    {
        fscanf(fpWeights, "%f", &wl[h][i]);
        delw1[h][i] = 0.0;
    }
/* read hidden:out weights */
for (j = 0; j < nOutputNodes; j++)
    for (h = 0; h <= nHiddenNodes; h++)
    {

```

Continued on page 22

The \$25 Network

Try the 1st truly low cost LAN

- Connect 2 or 3 PCs, XT's, AT's
- 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

Why waste money on simple file transfer systems?

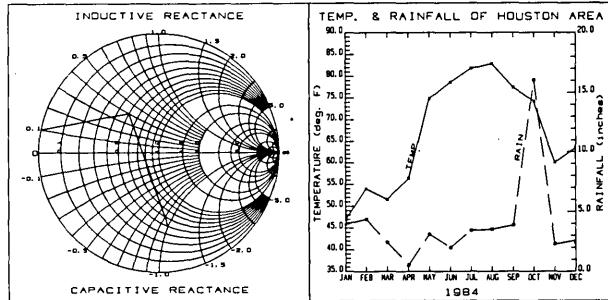
Reader Service Number 149

INGRAF 2.10

A multi-device graphics library for Scientific, Engineering, and Business users! Supports video, printer, and plotter graphics for personal computers.

Over 100 routines to create bar, pie, and smith charts; linear, log (semi-log and log-log), and polar plots; axes and grids with tick marks and labels, markers, line types, curves, arcs, circles, ellipses, and more. (Full source code; no royalties or run-time fees!)

The window function allows you to use up to 32 windows. You can set individual characteristics (such as sizing, scaling, labeling, etc.) for each window or identical characteristics for all windows.



INGRAF is available for C, FORTRAN-77, and QuickBASIC (Pascal available soon); support for numerous compilers.

We also have GRAFLIB 4.10 for video and printer graphics; PLOTLIB 4.10 for pen plotter graphics, and FORTLIB 4.05, a FORTRAN enhancement library.

Sutrasoft
P.O. Box 1733 • Sugar Land, TX • 77487
Order Line : 1-800-888-8460
All other calls : (713) 491-2088

Reader Service Number 173

```

fscanf(fpWeights, "%f", &w2[j][h]);
delw2[j][h] = 0.0;
}
fclose(fpWeights);
/* Read in all patterns to be learned: */
if ((fpPattern=fopen(szPattern, "r"))==NULL)
{
    fprintf(stderr, "%s: can't open %s\n",
            progname, szPattern);
    exit(1);
}
for (p = 0; p < nPatterns; p++)
{
    for (i = 0; i < nInputNodes; i++)
        if (fscanf(fpPattern, "%f",
                   &out0[p][i]) != 1)
            goto ALLPATTERNSREAD;
/*read target output for input patterns*/
for (j = 0; j < nOutputNodes; j++)
    fscanf(fpPattern, "%f", &target[p][j]);
/* read in identifier for each pattern */
fscanf(fpPattern, "%f ", &PatternID[p]);
}

ALLPATTERNSREAD:
fclose(fpPattern);
if (p < nPatterns)
{
    fprintf(stderr, "%s:%d out of %d patterns\
                read\n", progname, p, nPatterns);
    nPatterns = p;
}
/* open error output file */
if ((fpError = fopen(szError, "w")) == NULL)
{
    fprintf(stderr, "%s:can't open file %s\n",
            progname, szError);
    exit(1);
}
fprintf(stderr, nIterations>1?"Training..\n"
        :"Testing\n");
/* begin iteration loop */
for (q = 0; q < nIterations; q++)
{
    for (p = 0; p < nPatterns; p++)
    {
        /* hidden layer - Sum input to hidden
           layer over all input-weight combnatns*/
        for (h = 0; h < nHiddenNodes; h++)
        {
            /* begin with bias */
            float sum = w1[h][nInputNodes];
            for (i = 0; i < nInputNodes; i++)
                sum += w1[h][i] * out0[p][i];
            /* Compute output (use sigmoid) */
            out1[p][h] = 1.0/(1.0 + exp(-sum));
        }
        /* output layer */
        for (j = 0; j < nOutputNodes; j++)
        {
            float sum = w2[j][nHiddenNodes];
            for (h = 0; h < nHiddenNodes; h++)
                sum += w2[j][h] * out1[p][h];
            out2[p][j] = 1.0/(1.0 + exp(-sum));
        }
        /*delta output-Compute deltas for each
           output unit for a given pattern */
        for (j = 0; j < nOutputNodes; j++)
            delta2[p][j] = (target[p][j] -
                           out2[p][j]) * out2[p][j] * (1.0 -
                           out2[p][j]);
        /* delta hidden */
        for (h = 0; h < nHiddenNodes; h++)
        {
            float sum = 0.0;
            for (j = 0; j < nOutputNodes; j++)
                sum += delta2[p][j] * w2[j][h];
            delta1[p][h] = sum * out1[p][h] *
                           (1.0 - out1[p][h]);
        }
        /* adapt weights hidden:output */
        for (j = 0; j < nOutputNodes; j++)
        {
            float dw;          /* delta weight */
            float sum = 0.0;
            /* grand sum of deltas for each output
               node for 1 epoch*/
            for (p = 0; p < nPatterns; p++)
                sum += delta2[p][j];
            /*find new bias for each output unit*/
            dw = eta * sum+alpha *
                 delw2[j][nHiddenNodes];
            w2[j][nHiddenNodes] += dw;
            delw2[j][nHiddenNodes] = dw;
            /* Calculate new weights */
            for (h = 0; h < nHiddenNodes; h++)
            {
                float sum = 0.0;
                for (p = 0; p < nPatterns; p++)
                    sum += delta2[p][j]*out1[p][h];
                dw = eta * sum+alpha * delw2[j][h];
                w2[j][h] += dw;
                delw2[j][h] = dw;
            }
        }
        /* adapt weights input:hidden */
        for (h = 0; h < nHiddenNodes; h++)
        {
            float dw;          /* delta weight */
            float sum = 0.0;
            for (p = 0; p < nPatterns; p++)
                sum += delta1[p][h];
            /*find bias weight for hidden units*/
            dw = eta * sum + alpha *
                 delw1[h][nInputNodes];
            w1[h][nInputNodes] += dw;
            delw1[h][nInputNodes] = dw;
        }
    }
}

```

```

w1[h][nInputNodes] += dw;
delw1[h][nInputNodes] = dw;
/* Calculate new weights */
for (i = 0; i < nInputNodes; i++)
{
    float sum = 0.0;
    for (p = 0; p < nPatterns; p++)
        sum += delta1[p][h]*out0[p][i];
    dw = eta * sum + alpha*delw1[h][i];
    w1[h][i] += dw;
    delw1[h][i] = dw;
}
/* monitor keyboard requests */
if (kbhit())
{
    int c = getch();
    if ((c == toupper(c)) == 'E')
        MonitorError++;
    else if (c == ESC)
        break; /* End gracefully */
}
/* Sum Squared Error */
if (MonitorError || (q&nReportErrors==0))
{
    for (p=0, error=0.0; p<nPatterns; p++)
    {
        for (j = 0; j < nOutputNodes; j++)
        {
            float temp = target[p][j] -
                        out2[p][j];
            error += temp * temp;
        }
    }
    /* Average error over all patterns */
    error /= nPatterns;
    /* Print it number and err val */
    fprintf(stderr, "Iteration %5d/%-5d \
                  Error %f\r", q, nIterations,
                  error); /* to console */
    MonitorError = 0;
    if (q & nReportErrors == 0)
        fprintf(fpError, "%d %f\n", q,
                error); /* to file */
    /* End when error satisfactory */
    if (error < ErrorLevel)
        break;
}
/* end of iteration loop */
for (p = 0, error = 0.0; p < nPatterns; p++)
{
    for (j = 0; j < nOutputNodes; j++)
    {
        float temp=target[p][j] - out2[p][j];
        error += temp * temp;
    }
}
/* Average error over all patterns */
error /= nPatterns;
/* Print final it number and error value */
fprintf(stderr, "Iteration %5d/%-5d Error \
               %f\n", q, nIterations, error); /* CON */
fprintf(fpError, "\n%d %f\n", q, error); /* fp */
fclose(fpError);
/* print final weights */
if ((fpWeightsOut = fopen(szWeightsOut, "w")) == NULL)
{
    fprintf(stderr, "%s: can't write %s\n",
            progname, szWeightsOut);
    exit(1);
}
for (h = 0; h < nHiddenNodes; h++)
    for (i = 0; i <= nInputNodes; i++)
        fprintf(fpWeightsOut, "%g%c", w1[h][i],
                i%ITEMS==ITEMS-1 ? '\n' : ' ');
for (j = 0; j < nOutputNodes; j++)
    for (h = 0; h <= nHiddenNodes; h++)
        fprintf(fpWeightsOut, "%g%c", w2[j][h],
                j%ITEMS==ITEMS-1 ? '\n' : ' ');
fclose(fpWeightsOut);
/* Print final activation values */
if ((fpResults=fopen(szResults, "w"))==NULL)
{
    fprintf(stderr, "%s: can't write %s\n",
            progname, szResults);
    fpResults = stderr;
}
/* Print final output vector */
for (p = 0; p < nPatterns; p++)
{
    fprintf(fpResults, "%d ", p);
    for (j = 0; j < nOutputNodes; j++)
        fprintf(fpResults, " %f", out2[p][j]);
    fprintf(fpResults, " %-6.0f\n",
            PatternID[p]);
}
fclose(fpResults);

/* free dynamic storage for data */
MatrixFree(out0, nPatterns);
MatrixFree(out1, nPatterns);
MatrixFree(delta1, nPatterns);
MatrixFree(delw1, nHiddenNodes);
MatrixFree(w1, nHiddenNodes);
MatrixFree(out2, nPatterns);
MatrixFree(delta2, nPatterns);
MatrixFree(delw2, nOutputNodes);
MatrixFree(w2, nOutputNodes);
MatrixFree(target, nPatterns);
free(PatternID);
}
fclose(fpRun); /* close run file */
}

/* Array storage allocation routines */
/* Allocate space for vector of float cells for

```

Continued on page 24

$$o_j = \frac{1}{1 + \exp(-i_j)}$$

Equation 2

As you can see in Figure 2, the output, after going through the sigmoid function (also called a squashing function), is limited to values between 0 and 1. At a net input of 0 to the PE, the output is 0.5. For large negative net input values, the output of the PE approaches 0. For large positive values, it approaches 1.

The nonlinear nature of this sigmoid transfer function plays an important role in the performance of the neural network. We can use other transfer functions, as long as they're continuous and possess a derivative at all points.

Functions such as the trigonometric sine and the hyperbolic tangent have been used, but the exploration of these and other transfer functions is beyond the scope of this article. For more information, refer to Rumelhart and McClelland, and McClelland and Rumelhart.

Once Equation 2 calculates the output of each hidden layer PE, the net input to each output layer PE is calculated in a manner analogous to that used for calculating the net input for each hidden PE, as described by Equation 3. Likewise, the output of each output layer PE is calculated in the same way as for the outputs of the hidden layer PEs, as described by Equation 4.

$$i_1 = \sum_j w_{1j} o_j \quad \text{Equation 3}$$

$$o_1 = \frac{1}{1 + \exp(-i_1)} \quad \text{Equation 4}$$

This set of calculations, which results in the output state of the network (or simply the set of the output states of all the output PEs), is carried out the same way during the training phase as during the testing/running phase. The test/run operational mode just involves presenting an input set to the input nodes and calculating the resulting output state.

To summarize, during the feedforward calculations, two math operations are performed by PE(s). The output state, or activation, is obtained as a result. The first is a summation of previous layer PE outputs times interconnecting weights, and the second is the squashing function.

We can view the squashing function (illustrated in Figure 2) as a function similar to an analog electronics amplifier. The gain, or amplification, of the amplifier is analogous to the slope of the line,

Continued from page 23

```

one dimensional dynamic vector[cols] */
void VectorAllocate(VECTOR *vector, int nCols)
{
    if ((*vector = (VECTOR) calloc(nCols,
        sizeof(float))) == NULL)
    {
        fprintf(stderr, "Not enough memory\n");
        exit(1);
    }
}

/* Allocate space for columns (float cells) for
   dynamic two dimensional matrix[rows][cols] */
void AllocateCols(PFLOAT matrix[], int nRows, int nCols)
{
    int i;

    for (i = 0; i < nRows; i++)
        VectorAllocate(&matrix[i], nCols);
}

/* Allocate space for 2D dynamic matrix */
void MatrixAllocate(MATRIX *pmatrix, int nRows,
                   int nCols)
{
    if ((*pmatrix = (MATRIX) calloc(nRows,
        sizeof(PFLOAT)) ) == NULL)
    {
        fprintf(stderr, "Not enough memory\n");
        exit(1);
    }
    AllocateCols(*pmatrix, nRows, nCols);
}

/* free space for two dimensional dynamic array */
void MatrixFree(MATRIX matrix, int nRows)
{
    int i;

    for (i = 0; i < nRows; i++)
        free(matrix[i]);
    free(matrix);
}

```

♦ ♦ ♦

or the ratio of the change in output for a given change in input.

As you can see, the slope of the function (gain of the amplifier) is greatest for net inputs near zero. This serves to mitigate problems caused by noise, and by the possible dominating effects of large input signals.

The code in Figure 3, Batchnet.c completely implements our NNT.

Part 2

Next issue (Part 2) we'll train the network (by error back propagation) and show you how to use it. Meanwhile,

download the complete system from the Micro C BBS (or order the issue #51 disk for \$6, add \$2 postage for foreign orders) and start exploring this amazing technology.

If you want to really get into neural networks, a great place to start is Vol. I of *Parallel Distributed Processing* by Rumelhart and McClelland. You'll probably come away with a good understanding of back propagation networks if you read Chapters 1 through 4 and Chapter 8. Be prepared to read Chapter 8 at least twice.

Also consider attending conferences or symposia on neural networks. The

International Joint Conferences on Neural Networks (IJCNN), sponsored jointly by the Institute of Electrical & Electronics Engineers (IEEE), Neural Network Committee (NNC), and the International Neural Network Society (INNS). The societies alternate in terms of which one is lead society. For the time being, the INNS is lead in the Winter, IEEE NNC in summer.

The Winter 1990 IJCNN will be held in Washington, D. C., January 15-19, 1990. For more information, contact Deverman & Assoc., 4233 Spring Street #99, La Mesa, CA 92041, phone (619) 462-6800.

A smaller conference, but one of very high quality, is the Neural Information Processing Systems (NIPS) meeting, always held in Denver. This year, it will be held November 28 - December 2 at the Sheraton Denver Tech Center.

For more information, contact Kathy Hibbard, NIPS89 Local Committee, Univ. of Colorado, Engineering Center, Campus Box 425, Boulder, CO 80309-0425, phone (303) 492-4720. To give you an idea of the selectivity of this conference, this year over 500 papers were submitted; fewer than 100 will be accepted for oral or poster presentation.

References

Anderson, J. A. and E. Rosenfeld, Eds., *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, Mass., 1988, ISBN 0-262-01097-6, \$39.95.

McClelland, J. L. and D. E. Rumelhart, *Parallel Distributed Processing*, Vol. 2. MIT Press, Cambridge, Mass., 1986, ISBN 0-262-13218-4, \$16.75 paperback.

McClelland, J. L. and D. E. Rumelhart, *Explorations in Parallel Distributed Processing*. MIT Press, Cambridge, Mass., 1988, ISBN 0-262-63113-X, \$27.00 paperback (with software diskettes).

Rumelhart, D. E. and J. L. McClelland, *Parallel Distributed Processing*, Vol. 1. MIT Press, Cambridge, Mass., 1986, ISBN 0-262-18120-7, \$16.95 paperback.

Stubbs, D. "Neurocomputers." *M.D. Computing*, 5(3):14-24 (1988).

◆ ◆ ◆

REFURBISHED SEAGATE HARD DRIVES

ST-125	20 Meg	MFM	28MS	\$175.00
ST-125N	20 Meg	SCSI	28MS	\$230.00
ST-138	30 Meg	MFM	28MS	\$215.00
ST-138R	32 Meg	RLL	40MS	\$195.00
ST-138N	32 Meg	SCSI	40MS	\$235.00
ST-151	40 Meg	MFM	40MS	\$315.00
ST-157N	49 Meg	SCSI	40MS	\$275.00
ST-157R	50 Meg	RLL	40MS	\$240.00
ST-225	20 Meg	MFM	65MS	\$160.00
ST-225N	21 Meg	SCSI	65MS	\$200.00
ST-225R	20 Meg	RLL	65MS	\$150.00
ST-238R	30 Meg	RLL	65MS	\$165.00
ST-250R	40 Meg	RLL	70MS	\$190.00
ST-251-0	40 Meg	MFM	40MS	\$250.00
ST-251-1	40 Meg	MFM	28MS	\$270.00
ST-251N	40 Meg	SCSI	40MS	\$290.00
ST-277R	65 Meg	RLL	40MS	\$280.00
ST-277N	65 Meg	SCSI	40MS	\$325.00
ST-277R-1	65 Meg	RLL	28MS	\$320.00
ST-296N	85 Meg	SCSI	28MS	\$380.00
ST-4053	40 Meg	MFM	28MS	\$310.00
ST-4096	80 Meg	MFM	28MS	\$440.00
ST-4144R	120 Meg	RLL	28MS	\$510.00

Listed above are refurbished SEAGATE hard drives. Warranty on these units is 90 DAYS or the remainder of the factory warranty, whichever is greater. Most drives have six (6) months plus remaining warranty.

FOR ONE YEAR WARRANTY ADD \$45.00 EACH

THE ABOVE DRIVES SUBJECT TO STOCK

ALL SALES FINAL ON REFURBISHED DRIVES

Controllers and cables 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

3D-Surface Generation

An In-Depth Look At Graphics, Part 2

Okay, now that we have the theory, it's time to dissect the code. This time Greg explains the gnits and the grits.

I mentioned last issue that the z values for a 3D plot must be stored in a matrix declared as float **z. alloc_2d_array() handles the dynamic allocation used in both surface() and in the demo program. The method permits the dynamic allocation of very large arrays, of any type, not limited to what will fit in a 64K segment. *Editor's note: Greg's code was too big to include this time. Take a look for it on the Micro C BBS or the Issue #51 listings disk. (If you got last issue's listings, you already have the complete 3D package.)*

Since the tighter the grid mesh, the nicer the result, I recommend this method for allocating z. For smaller arrays there are better methods. Reference 1 contains an excellent discussion of these topics.

surface() also expects you to supply a line function. Now that I have Turbo C 2.0 instead of 1.0, the new line function lacks the "color" argument. It uses a separate function to set the line color. You could kludge your own function to pass to surface(), using Turbo's line() and setcolor(), but it would be better to modify the printer graphics modules and threed module to make my line function compatible with Turbo's.

The hmax parameter passed to surface() must always be MAXROWS-1 or 1599 (as currently configured) whether you use Portrait or Landscape mode (for printers). For CRT plots, it should be the maximum y value for the current graphics mode. You can obtain this value via a call to Turbo's getmaxy().

Finally, graphics programs tend to use a lot of memory, so I used the large code model for all the source code.

On To Printer Graphics

Module grafprt contains an assortment of functions to perform high-resolution plotting on IBM Graphics Printers and compatibles such as Epson FX and LX printers. The basic idea is to use the ESC L command to turn on low-speed double density graphics mode for an 8" line with 960 dots. You can obtain a vertical resolution of 1600 dots by using the ESC 3 command to select $\frac{1}{2}16$ line spacing.

The bit map is stored in two 2-dimensional arrays dynamically allocated using alloc_2d_array() with MAXCOLS (1600) columns and MAXPRINTROWS (100) rows of type unsigned char **.

The two arrays, Evenrow and Oddrow, interlace such that, when printing the bit map, one even row prints, the line spacing sets to $\frac{1}{2}16$ ", an odd row prints, the line spacing sets to $\frac{2}{2}16$ ", then the next even row prints.

prntgr() prints by looping through the entire process MAXPRINTROWS times, yielding 1600 vertical dots in 10 $\frac{5}{8}$ ". Reference 2 outlines this interlacing method. It is also where I obtained Professor Rasala's Bresenham line implementation. This achieves a 120 dots per inch density across the width of the page, and 160 dots per inch down the length of the page.

I also derived the point setting and blanking routines from Reference 2, so far as concerns the interlacing.

There is no need to discuss the rest of the code, as it is sufficiently documented internally. A brief description of the functions in module grafprt should suffice.

portrait() – Sets a global variable to indicate Portrait mode plotting.

landscape() – Sets a global variable to indicate Landscape mode plotting.

initgrafprt() – Allocates arrays for the bit map from the heap and initializes some global variables.

dismem() – De-allocates bit map array space.

pset() – Plots a point in the bit map.

plank() – Blanks point in the bit map.

prntgr() – Sends the graphics page (bit map) to the printer.

iprthn() – Draws a line in device integer coordinates.

frame() – Draws a boundary around the maximum (8" by 10 $\frac{5}{8}$ ") plotting region.

framxy() – Draws a boundary around the plot window defined by defreg().

transf() – Transforms real (x,y) to integer world (ix, iy) coordinates.

itransf() – Transforms integer world (ix, iy) to plot device integer (dx, dy) coordinates.

defreg() – Defines the correspondence between integer and real worlds.

prtlin() – Draws a line specified in real world coordinates.

Printer Graphics String Routines

Just to provide a complete set of tools, module grafstr contains routines that work with module grafprt to let you print scalable, rotatable, justifiable (left, right, and center) graphics strings.

Again, I think the code is adequately documented internally, so I will just list the functions in module grafstr.

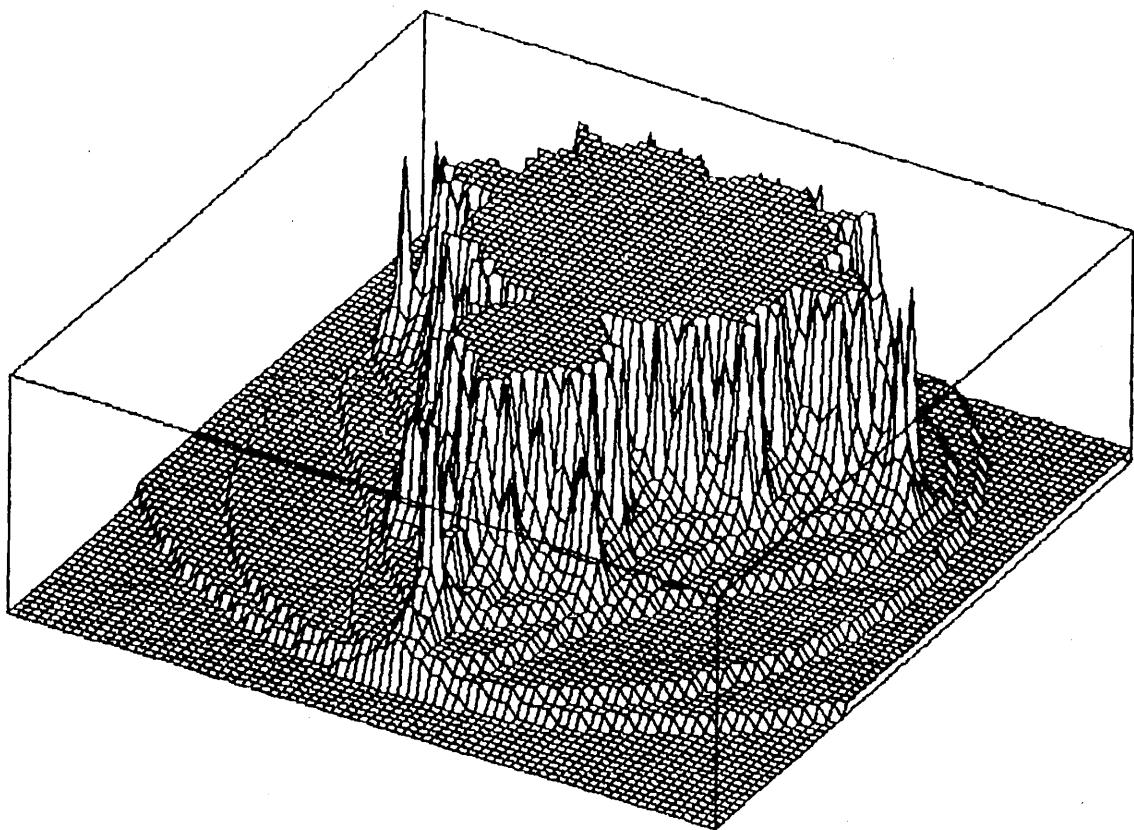
aspect_and_rotate() – Performs aspect ratio correction and rotations for strings. Used only by intprntstr().

swapmode() – Swaps between portrait and landscape modes by exchanging global variables declared in module grafprt. Used by intprntstr() and prtstr().

intprntstr() – Prints a string at (ix, iy) in integer world coordinates.

prtstr() – Prints a string at (x, y) in real world coordinates.

Module grafstr contains the character font description, and the comments include a complete description of how it works.



Now For The Fun Part

A demo program, test3d, demonstrates the use of the above described modules. Since we all like the Mandelbrot set so much, it plots surfaces produced by the Mandelbrot generator published on page 8 of *Micro Cornucopia* #39, Jan.-Feb. 1988. It has been modified to be consistent with the requirements of surface().

One important modification is scaling in the z direction. I didn't go through a detailed description of Cartesian geometry just for fun, you know. Cartesian implies the scales are the same in the x, y, and z directions. Before you pass a surface to surface(), you should scale the z values.

Scaling according to the larger of the two distances $x_{\max} - x_{\min}$ and $y_{\max} - y_{\min}$ usually works well. For the Mandelbrot generator, imagine how silly your surface would look if your x range was 0.02, your y range 0.25, and your z range was 500.

test3d allows plotting on the CRT or

on the printer (in both Portrait and Landscape modes). This page shows an example of the printer output.

Generating printer graphics on a tight grid, say $nx = 100$ and $ny = 100$, takes a lot of time. But you can get away with a small number of iterations and still investigate a lot of interesting terrain. Use the CRT mode to look around at likely regions of the set in coarse resolution, then use the printer mode to make pretty pictures.

I developed this software on a Leading Edge Model D running at 4.77 MHz. I find the time performance acceptable at this lowest possible denominator in the IBM PC world. At least, I keep telling myself it's acceptable. A spare 4 or 5 grand would change my mind real fast.

On my clunky Leading Edge, I can produce a 100 by 100 resolution surface, including the generation of the Mandelbrot values and the printout, in less than an hour. The surface generation into a printer bit map takes about 16 minutes at that resolution. On a CRT display the

surface is produced in real time (after the Mandelbrot values are produced).

Another Idea

Those of you with high-resolution, many colored CRT displays could try modifying the surface program so it fills the quadrangles with colors that correspond to the average height of the quadrangle.

But then enough's enough. It's definitely time to scout out some more beer.

References

- (1) *Advanced C Tips and Techniques*, Paul Anderson and Gail Anderson, Howard W. Sams & Co., 1988.
- (2) "High-Resolution Printer Graphics," Mark Bridger and Mark Goresky, *BYTE*, November, 1985, pp. 219-232.
- (3) "The Painter's Algorithm," Richard Chandler and Gary Faulkner, *PC Tech Journal*, November, 1985, pp. 181-187.



The Poet And The Computer

Stirring A Little Conscience Into The Data

In our quest for ever greater quantities of information and ever more sophisticated methods of manipulating, verifying, and displaying it, we risk forgetting something very important.

A poet, said Aristotle, has the advantage of expressing the universal; the technician or specialist expresses only the particular. The poet moreover, can remind us that man's greatest energy comes not from his dynamos but from his dreams. The notion of where a man ought to be instead of where he is; the liberation from cramped prospects; the intimations of immortality through art—all these proceed naturally out of dreams. But the quality of man's dreams can only be a reflection of his subconscious. What he puts into his subconscious, therefore, is quite literally the most important nourishment in the world.

Nothing really happens to a man except as it is registered in the subconscious. This is where event and feeling become memory and where the proof of life is stored. The poet—I use the term to include all those who have respect for and speak to the human spirit—can help supply the subconscious with material to enhance its sensitivity, thus safeguarding it. The poet, too, can help keep man from making himself over in the image of his electronic marvels. The danger is not so much that man will be controlled by the computer as that he may imitate it.

There once was a time, in the history of this society, when the ability of people to convey meaning was enriched by their knowledge of and access to the work of creative minds from across the centuries. No more. Conversation and letters today, like education, have become enfeebled by emphasis on the functional and the

purely contemporary. The result is a mechanization not just of the way we live but of the way we think, and of the human spirit itself.

The delegates to the United States Constitutional Convention were able to undergird their arguments with allusions to historical situations and to the ideas of philosophers, essayists, and dramatists. Names such as Thucydides, Aristotle, Herodotus, Plutarch, or Seneca were commonly cited to support their positions. They alluded to fictional characters from Aristophanes, Marlowe, or Shakespeare to lend color to the exploration of ideas. Analytical essays by Hamilton, Madison, and Jay that appeared in *The Federalist Papers* were an excursion into the remote corners of history.

Men such as Jefferson, Adams, Franklin, and Rush could summon pertinent quotations from Suetonius or Machiavelli or Montaigne to illustrate a principle. If they referred to Bacon's opinion of Aristotle, they didn't have to cite particulars; they assumed such details were common knowledge. Their allusions were not the product of intellectual ostentation or ornamentation but the natural condiments of discourse, bringing out the full flavor of the cultivated intelligence.

The same was true of correspondence. People regarded letters as an art form and a highly satisfying way of engaging in civilized exchange. The correspondence of Jefferson and Adams and Priestley was not so much a display of personal matters as a review of the human condition. It was not unusual for the writers to range across the entire arena of human thought as a way of sharing perceptions. Allusion was common currency. Today, we rarely turn to letters as a way of embarking on voyages of intellectual discovery.

The essential problem of man in a computerized age remains the same as it

has always been. That problem is not solely how to be more productive, more comfortable, more content, but how to be more sensitive, more sensible, more proportionate, more alive. The computer makes possible a phenomenal leap in human proficiency; it demolishes the fences around the practical and even the theoretical intelligence. But the question persists and indeed grows whether the computer makes it easier or harder for human beings to know who they really are, to identify their real problem, to respond more fully to beauty, to place adequate value on life, and to make their world safer than it now is.

Electronic brains can reduce the profusion of dead ends involved in vital research. But they can't eliminate the foolishness and decay that come from the unexamined life. Nor do they connect a man to the things he has to be connected to: the reality of pain in others; the possibilities of creative growth in himself; the memory of the race; and the rights of the next generation.

The reason these matters are important in a computerized age is that there may be a tendency to mistake data for wisdom, just as there has always been a tendency to confuse logic with values, and intelligence with insight. Unobstructed access to facts can produce unlimited good only if it is matched by the desire and ability to find out what they mean and where they would lead.

Facts are terrible things if left sprawling and unattended. They are too easily regarded as evaluated certainties rather than as the rawest of raw materials crying to be processed into the texture of logic. It requires a very unusual mind, Whitehead said, to undertake the analysis of a fact. The computer can provide a correct number, but it may be an irrelevant number until judgment is pronounced.

By Norman Cousins

Dean's Office - School of Medicine
12-138 CHS - U.C.L.A.
Los Angeles, CA 90024
(213) 825-8281

To the extent, then, that man fails to make the distinction between the intermediate operations of electronic intelligence and the ultimate responsibilities of human decision and conscience, the computer could obscure man's awareness of the need to come to terms with himself. It may foster the illusion that he is asking fundamental questions when actually he is asking only functional ones. It may be regarded as a substitute for intelligence instead of an extension of it. It may promote undue confidence in concrete answers. "If we begin with certainties," Bacon said, "we shall end in doubts; but if we begin with doubts, and we are patient with them, we shall end in certainties."

The computer knows how to vanquish error, but before we lose ourselves in celebration of victory, we might reflect on the great advances in the human situation that have come about because men were challenged by error and would not stop thinking and probing until they found better approaches for dealing with it. "Give me a good fruitful error, full of seeds, bursting with its own corrections," Ferris Greeslet wrote. "You can keep your sterile truth for yourself."

Without taking anything away from the technicians, it might be fruitful to effect some sort of junction between the computer technologist and the poet. A genuine purpose may be served by turning loose the wonders of the creative imagination on the kinds of problems being put to electronic tubes and transistors. The company of poets may enable the men who tend the machines to see a larger panorama of possibilities than technology alone may inspire.

Poets remind men of their uniqueness. It is not necessary to possess the ultimate definition of this uniqueness. Even to speculate on it is a gain.

♦ ♦ ♦



The question persists and indeed grows whether the computer makes it easier or harder for human beings to know who they really are, to identify their real problems, to respond more fully to beauty, to place adequate value on life, and to make their world safer than it now is.

LIMBO, Part Four

LIMBO Gets A Body

This time, ladies and gents, you'll watch as LIMBO gets a body (weird but true, serial fans). This is all in preparation for next time, when Igor (I mean LIMBO) gets a brain. Stay tuned, this melodrama only gets better.

To describe the superstructure, I'll start at the bottom and work my way up. At the bottom of any good robot are the batteries. Mounting batteries on a robot requires a little forethought. First, you must consider the weight distribution.

Batteries are typically the heaviest part of any mobile 'bot. Place the mass too high, and it'll be top heavy; place it too far from center line, and your unit wobbles and doesn't have much traction. Try to keep two-thirds of the robot mass concentrated in the lower third of the machine, distributed as closely and evenly about the center line as possible.

Also think about the battery's environment. No matter what the manufacturer says, you can usually expect a small amount of leakage from gelled-electrolyte cells, especially if they're not vertical. Robotics slang for this leakage is "The Ooze" or "Battery Ick." Robots tend to work batteries hard, too, causing the batteries to heat up—more ooze.

So leave some space around the batteries and, because of the hydrogen gas, make sure the space is well vented. Cool batteries last longer.

LIMBO will run on either one or two of the Panasonic batteries mentioned in the second article. To keep the center of gravity low, both batteries lie flat in the bottom of the superstructure.

Wooden cleats that hold the superstructure together provide $\frac{1}{2}$ " of air above and below the batteries. These spaces are great for routing wires from



the base to the stepper driver boards or the controller cards.

The card cage provides slots for five STD bus cards. The STD standard specifies a minimum card-to-card spacing of 0.5", but I chose 1.2" so I could use wire-wrapped boards. Above the card cage is a $\frac{1}{2}$ " airspace between the top cross panel and the middle disk. Use this space (it provides clearance for the STD card ejectors) to route wires between LIMBO's front and rear sections.

Why STD? Indeed, why a bus at all? For those of you who experienced the early closed architecture "Data Appliance" days of the Macintosh, the answer should be clear: a bus is the best way to overcome the designer's lack of clairvoyance. Where technology marches, LIMBO will follow.

STD is the quiet bus. Not flashy like VME, or superfast like Futurebus, STD is nonetheless the bus of choice for industrial control applications. The small form factor (4"x6.5") means ruggedness

and low cost. STD is traditionally an 8-bit bus, but a compatible high performance 32-bit standard has just been introduced, so there's plenty of room to grow.

Above the card cage the middle disk, X-dividers, and top disk make up the optical sensor platform. The X-dividers divide the platform into four sensor bays—facing front, rear, left, and right. Each bay shields its optical emitter/sensor arrays from ambient light and from adjacent emitter/sensor arrays.

Making The Superstructure

Tools: Machinist compass, straight-edge, try square, band saw or coping saw, brad point drill bits, spade bits, countersink, jack plane, marking gage.

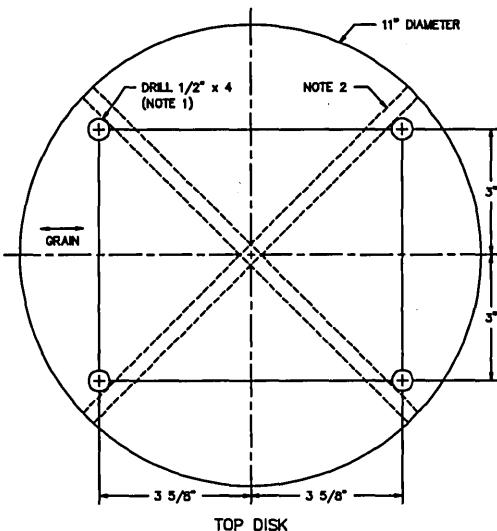
Materials:

2 pieces plywood $\frac{1}{4}$ "x24"x24" (both sides good),
1 piece clear hemlock moulding $\frac{1}{2}$ "x $\frac{3}{8}$ "x72",
4 plastic Trapez fittings,
4 1" steel right angle braces,
10 nylon card guides,
32 #6x $\frac{5}{8}$ " flathead phillips woodscrews,
32 #6-32x $\frac{3}{8}$ " flathead phillips machine screws,
4 #10-32x $\frac{3}{4}$ " flathead machine screws,
8 #6-32x $\frac{3}{8}$ " aluminum standoffs,
4 #10-32x $\frac{1}{4}$ " Tee-nuts,
24 #6-32x $\frac{1}{4}$ " Tee-nuts,
1 oz. 5-minute epoxy

Step 1: Fabricating The Pieces

With the machinist compass, lay out three 11" diameter disks on the better of the two squares of plywood. (See Figure 1.) Try to get plywood squares with as little warp as possible. Noting the grain directions on the disk layout diagram, lay out perpendicular diameters and label bottom, middle, and top disks as shown.

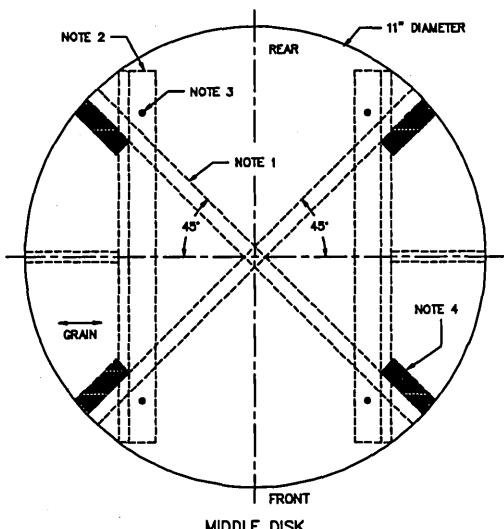
Lay out all straight lines by lightly (and precisely) scoring the plywood surfaces with a hobby knife. Sometimes lines



TOP DISK

NOTE 1: These holes should line up with Trapez fitting screws on the middle disk. The holes allow screwdriver access.

NOTE 2: Dashed lines show placement of X-divider panels on the underside of the top disk.



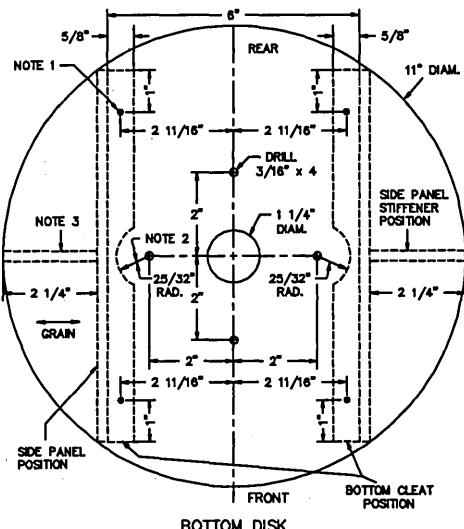
MIDDLE DISK

NOTE 1: 45° dashed lines show placement of X-divider panels on the top side of the disk.

NOTE 2: Dashed lines show placement of cleats & panels on the bottom side of the disk. See Bottom Disk layout for dimensions.

NOTE 3: Drill & countersink from the top side of the disk. See placement dimensions from the Bottom Disk layout.

NOTE 4: Hatched areas show placement of Trapez fittings. See text for mounting details.



BOTTOM DISK

NOTE 1: Drill & countersink from the bottom side of the disk. See text for exact drilling sequence.

NOTE 2: Clearance arc for 10-32 wing nut.

NOTE 3: Dashed lines show placement of cleats and panels on the top side of the disk.

NOTE 4: Material is 1/4" plywood.

Figure 1—Plywood Disk Layouts

with the grain tend to disappear, so mark them with a pencil, too. Don't be afraid to put too many labels on the wood; you're building a robot, not a piece of furniture. Cut out the disks and sand to final dimension.

Transfer the diameter lines to the opposite side of each disk using the try square. Each disk will have some warp in it, generally at right angles to the outer ply grain. Place the bottom disk crown up, the middle disk crown down, and top disk crown up; label the sides facing up "Top."

With middle and bottom disks oriented this way, their warps will cancel out, bringing the whole structure into proper alignment; the X-dividers will take care of the warp in the top disk.

Complete the layout lines for cleats, panels, and hole locations for each disk on their top and bottom surfaces. Drill

$3\frac{1}{16}$ " dia. mounting holes and the $1\frac{1}{4}$ " dia. wire access hole in the bottom disk. Drill the screwdriver access holes in the top disk. Use a brad-point drill for the smaller holes to get clean cuts.

On the remaining plywood square, lay out the side panels, stiffeners, and cross panels, again orienting the grain as shown. (See Figures 2 and 3.) These rectangular parts are best cut out on a table saw (for the precision). For those of you without access to a table saw (like me), cut out the pieces using a band saw, sabre saw, circular saw, or what have you, leaving $\frac{1}{32}$ " or so to sand or plane down to the line. I prefer to use a jack plane because I can shoot an edge straighter and truer with it than with a table saw.

Mark the line you'll plane down to on both sides of the piece. Tape the ends with masking tape just below the line to

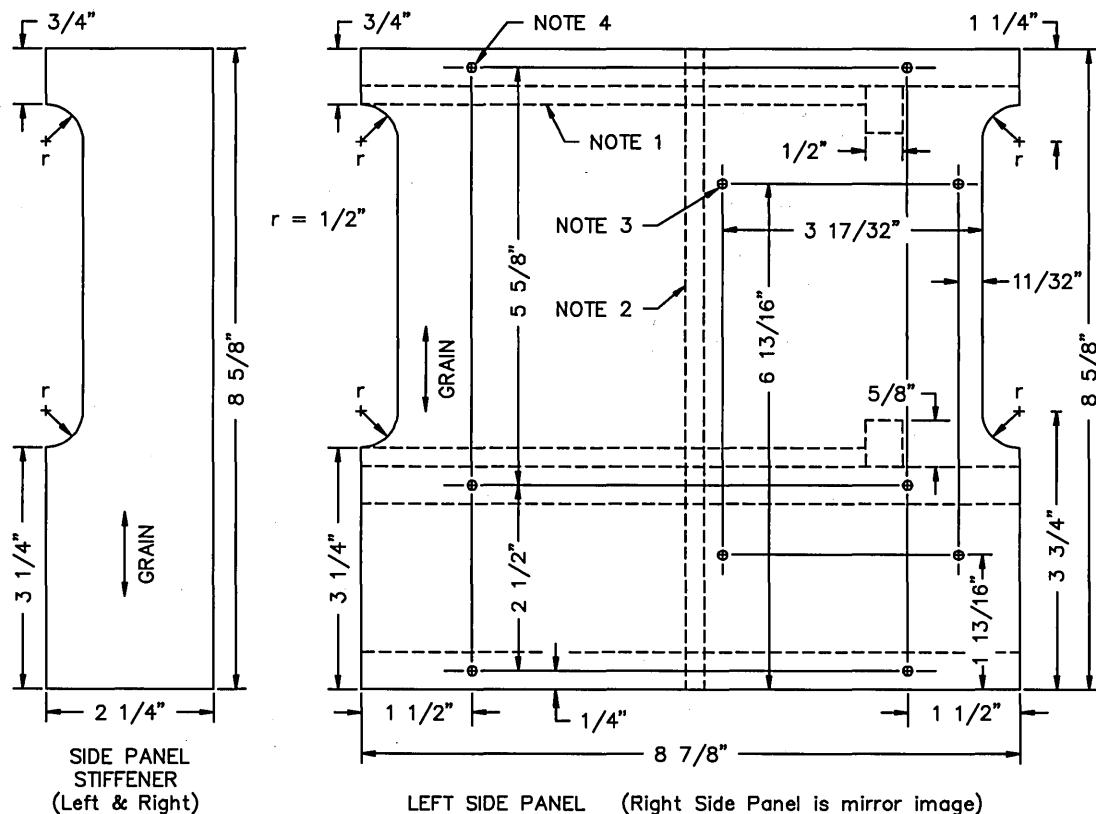
help prevent corner splitting. Precutting the corners with a hobby knife helps, too. Take an equal number of strokes from each direction, letting the plane do the work. You should need fewer than ten strokes to make the edge true and square.

Take the time to resharpen the iron after you're done. The adhesives used in making plywood are hard on the edge, and few tools are more disappointing to use than a dull plane.

After the side and stiffener panels are cut and planed to size, cut out the curved wiring recesses and sand smooth.

The X-divider panels require special attention. Once you've established the $3\frac{1}{2} \times 11\frac{1}{2}$ " outer dimensions, you must cut the intersecting grooves. The key to a tight fitting joint is to use the actual thickness of each panel at the middle, not the nominal $\frac{1}{4}$ " thickness.

Figure 2—Side Panels and Stiffeners



NOTE 1: Dashed lines show positions of cleats & cross panels on the inner side.

NOTE 2: Vertical dashed lines show placement of stiffener on outer side.

NOTE 3: LIMBO Stepper PCB mount holes (4). Drill & countersink from inner side of panel.

NOTE 4: Cleat mounting holes (6). Drill & countersink from outer side of panel.

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 3½" 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

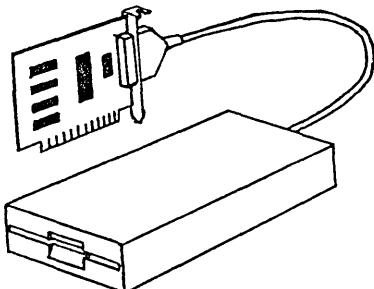
Two drive version of the CompatiCard, for the XT or AT. Same drive support as the CompatiCard I except no 8" or single density.

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 3½" 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.

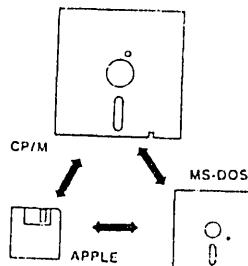
Megamate	\$ 329.95
----------------	-----------

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 5½" high density, 96TPI, 3½" (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 3½" 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.

Reader Service Number 10

Copy II PC by Central Point Software

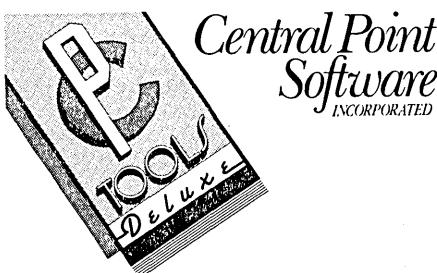
Don't let a damaged copy protected diskette stop you cold. Copy II PC lets you back up your master disks so you can keep going even when your key disk can't.

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 3½" 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
-----------------------------------	-----------

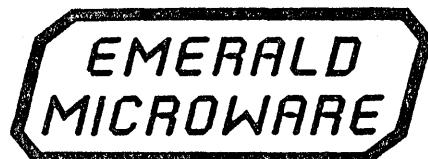


PC Tools Deluxe V5.5

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.

To do this, first find the vertical center line of both divider panels. Scribe a line offset $\frac{1}{8}$ " from the center line of one panel with the try square as a guide, then place the edge of the other panel along the scribed line so you can scribe its thickness onto the first panel.

When you've marked both panels, mark the end of the slots at the midpoint

of the width. Cut the slots to inside the lines thus scribed. Trial fit, sand, trial fit, etc., until the panels slide snugly together. Check for square often during this process. There should be little or no play and the top panel should seat on the same plane as the bottom panel when you are through.

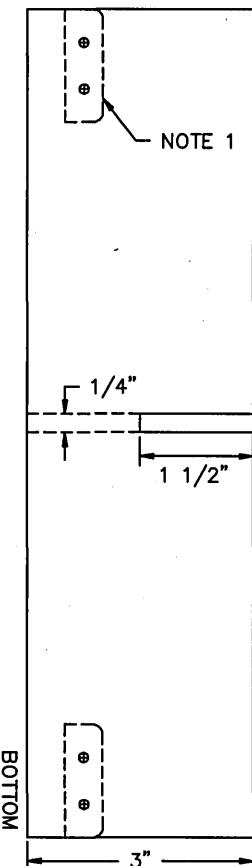
Next come the cleats. The moulding

stock you're likely to find at the lumber yard will probably not have a $\frac{1}{2}'' \times \frac{5}{8}$ " cross section. Get $\frac{3}{4}$ " square moulding and plane to final dimensions. This stock will be less than the nominal $\frac{3}{4}$ " square (more like $\frac{5}{8}$ " square).

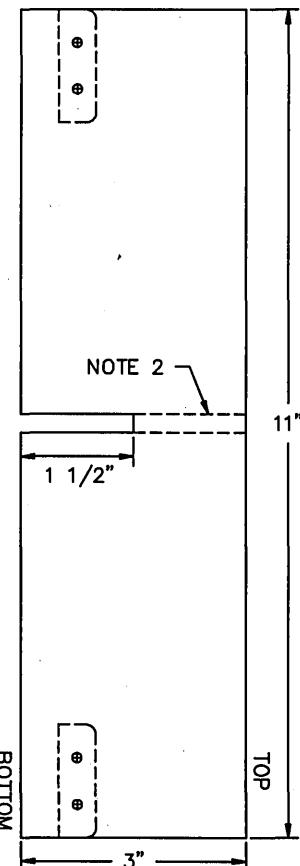
It's much easier to do the planing before you cut the cleats to length. Mark the whole length of the stock on two par-

Figure 3—X-dividers, Trapeze Fittings, & Cross Panels

RIGHT FRONT



RIGHT REAR



LEFT FRONT

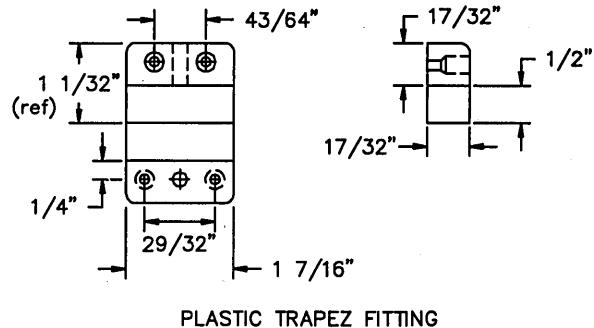
BOTTOM

LEFT REAR

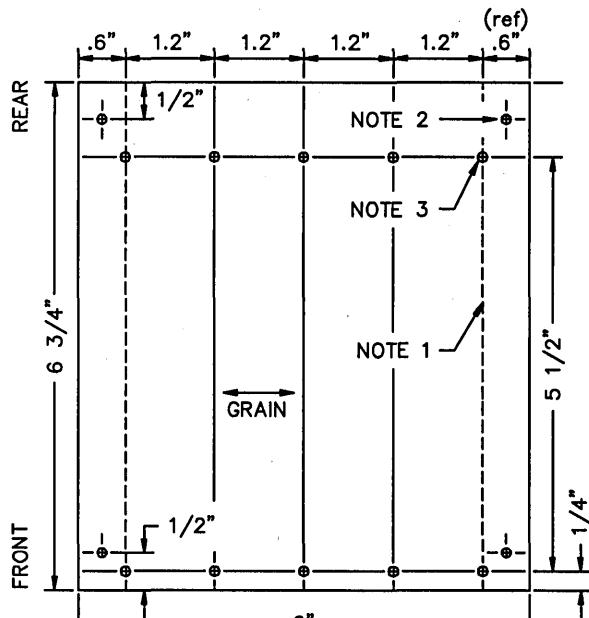
X-DIVIDERS

NOTE 1: Dashed lines show location of Trapez fitting (upper half). See text for mounting details.

NOTE 2: X-divider join line.



PLASTIC TRAPEZ FITTING



TOP & BOTTOM CROSS PANELS

NOTE 1: Dashed lines show placement of cleats on bottom of bottom cross panel and top of top cross panel.

NOTE 2: Drill & countersink from panel side, centered on cleat. (Cleat mounting holes).

NOTE 3: Card guide mounting holes (6). See text on mounting card guides.

C CODE FOR THE PC

source code, of course

<i>NEW!</i>	MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers)	\$500
	CSource Application Program Generator by MBAC (includes all source code; generator & libraries)	\$500
	dB2c (dBase-to-C translator; includes db.Files for C and db.Tools for C)	\$325
	CQL Query System (SQL retrievals on B-trees plus windows)	\$325
	GraphiC 5.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	\$325
<i>NEW!</i>	Drasch Clisp with Cruces (Lisp library and programming environment with rule processing capability; natural language example)	\$300
	Greenleaf Data Windows (windows, menus, data entry, interactive form design; specify compiler)	\$300
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	\$290
<i>NEW!</i>	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
	PC PostScript (complete PostScript interpreter (ROM Version 47.0A), 80286/386 only, many device drivers, optimized graphics, fast)	\$250
	db.File & db.Retrieve (B-tree and network database with SQL query and report writer)	\$245
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF; specify compiler)	\$225
<i>NEW!</i>	Booter Toolkit (floppy disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	\$210
<i>NEW!</i>	CDirect (multi-user hashed file manager; variable length fields, binary or ASCII data, alternate keys)	\$210
<i>NEW!</i>	BCPL Compiler (this is not C source but BCPL source; BCPL is the mother of C)	\$195
	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, Quattro, & Clipper)	\$155
	OS/88 (industrial-strength U*x-like operating system, many tools, cross-development from MS-DOS)	\$150
<i>NEW!</i>	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
<i>NEW!</i>	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
<i>NEW!</i>	Installation Toolkit 3.0 (250-page User's Guide, interactive script development program; many special cases covered)	\$120
<i>NEW!</i>	B-Strings (dynamic string handling; cut, copy, paste, search, user input, etc.; non-fragmenting memory management)	\$105
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	\$105
<i>Updated!</i>	Minix Operating System (Version 1.3; U*x-like operating system, includes manual)	\$105
	PC/IP (CMU/MIT TCP/IP for PCs; Ethernet, Appletalk & NETBIOS drivers, RVD, update by Dan Lanciani)	\$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	\$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
	Outer 1.0 (beautiful theorem-prover by Bill McCune; includes manual & two books by Wos; complete starter kit)	\$80
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	\$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	\$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	\$80
	Make (macros, all languages, built-in rules)	\$75
	eval() (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 XTs)	\$75
	Professional C Windows (lean & mean window and keyboard handler)	\$70
	ScreenLib (simple screen definitions, windows, pop-up menus, context-sensitive help)	\$65
	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
	Asyn-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
<i>NEW!</i>	REGX Plus (search and replace string manipulation routines based on regular expressions)	\$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
<i>NEW!</i>	LaplaceB (LaPlace polynomials, real and complex)	\$50
<i>Updated!</i>	CLIPS (rule-based expert system generator, Version 4.3; advanced manuals available)	\$50
<i>NEW!</i>	Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	\$45
<i>Updated!</i>	Fortran-to-C Translator by Polyglot	\$40
<i>NEW!</i>	DES Encryption & Decryption (2500 bits/second on 4.77 MHz PC for on-the-fly encryption at 2400 baud; U.S. only)	\$40
<i>NEW!</i>	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 & PREP (YACC workalike parser generator & attribute grammar preprocessor; now includes documentation)	\$35
	PC-XINU (Comer's XINU operating system for PC)	\$35
<i>NEW!</i>	RXC & EGREP (Regular Expression Compiler and Pattern Matching; RXC makes finite state machine from regular expression)	\$35
<i>NEW!</i>	CCALC (handy extended-precision calculator; real and complex models; many built-in functions)	\$30
<i>NEW!</i>	GNU Awk & Diff for PC (both programs in one package)	\$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	\$30
	Crunch Pack (14 file compression & expansion programs)	\$30
	Pascal P-Code Compiler & Interpreter or Pascal-to-C Translator (Wirth standard Pascal)	\$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
	A68 (68000 cross-assembler)	\$20
	XLT Macro Processor (general purpose text translator)	\$20
	Data	
<i>NEW!</i>	Moby Pronunciator (150,000 words & phrases encoded with full IPA pronunciation & emphasis points; 900 distinguished by part-of-speech)	\$160
<i>NEW!</i>	Moby Part-of-Speech (200,000 words and phrases described by prioritized part(s)-of-speech)	\$105
<i>NEW!</i>	Moby Hyphenator (150,000 words fully hyphenated/syllabified)	\$120
	Smithsonian Astronomical Observatory Subset (right ascension, declination, & magnitude of 258,997 stars)	\$60
	Moby Words (500,000 words & phrases, 9,000 stars, 15,000 names)	\$65
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	\$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	\$30
	KST Fonts (13,200 characters in 139 mixed fonts; specify TeX or bitmap format)	\$30
	USNO Interactive Computer Ephemeris (high-precision moon, sun, planet & star positions)	\$30
	U. S. Map (15,701 points of state boundaries)	\$15
	The Austin Code Works	
	11100 Leafwood Lane	Voice: (512) 258-0785
	Austin, Texas 78750-3409 USA	BBS: (512) 258-8891
		FAX: (512) 258-1342

Free surface shipping for cash in advance For delivery in Texas add 7%

MasterCard/VISA

Reader Service Number 4

MICRO CORNUCOPIA, #51, Jan-Feb, 1990 35

allel sides with a carpenter's marking gage set to $\frac{1}{2}$ " depth, plane to the lines, then mark the other two sides to $\frac{5}{8}$ " and plane to those lines. Keep it square.

Once you've planed the stock, cut six $8\frac{7}{8}$ " lengths, two 6" lengths, and two 2" lengths. Cut them slightly long, then sand to exact length. Select two of the long cleats and mark the $25\frac{3}{32}$ " radius clearance at their midpoints as the bottom disk diagram shows. Cut the arcs out and sand smooth. These arcs will provide clearance for wing nuts to hold the superstructure to the base.

Step 2: Preparing The Panels

Woodscrews hold the plywood panels and cleats together. Each screw requires a pilot hole and must be countersunk.

First clamp a cleat to the bottom disk, carefully lining up the edges of the cleat with the lines scribed earlier before tightening the C-clamps. If the disk is warped, the clamps and cleat will hold it flat.

Drill the holes through the disk into the cleat with a #44 bit. The depth to drill is $\frac{1}{16}$ " less than the combined thickness of the panel/cleat combination, or about $1\frac{11}{16}$ ". Number the cleat and the disk

before unclamping so you know which cleat goes where. Repeat for the top disk and the top and bottom cross panels (i.e., the top and bottom of the card cage).

Treat the side panels in a similar fashion. Orient the side panel so it bows away from the cleat. Be sure to orient the numbered cleat properly before clamping and drilling as above. The drilling depth this time should be about $1\frac{3}{16}$ ". Repeat for the rest of the long cleats.

Once you've drilled all the panel/cleat pilot holes, set aside the cleats and ream out all the panel holes with a $\frac{9}{64}$ " bit. Chuck up a 60 degree countersink and countersink one of the holes just enough so that a #6 flathead wood screw seats flush.

With the drillpress power off, bring the countersink bit gently down into countersunk hole until you have firm contact. Temporarily lock the spindle in place if your drillpress has this feature, otherwise hold the spindle in place while you set the depth stop to the current depth.

If your drillpress does not have a depth stop, then mark the position on the depth indicator (I've yet to see a drillpress without one or the other). Countersink the rest of the holes in all the other panels and disks, taking care to countersink from the correct side (i.e., the side you drilled the pilot holes from).

Drill the LIMBO Stepper PCB mounting holes ($\frac{9}{64}$ ") for left and right side panels. Countersink these holes, too, but from the opposite side of the panels (the crown side). Drill the $\frac{1}{8}$ " card guide mounting holes in both the top and bottom cross panels. Do not countersink these holes.

To finish with the panels, epoxy the stiffeners to the side panels, then epoxy and screw the long cleats to the bottom disk and top and bottom cross panels. Epoxy the 2" backstop cleats to the middle of the side panels in the battery compartment: Place them level with their ends butting against (but not epoxied to) both the bottom disk cleats and bottom cross panel cleats. The middle disk will eventually be attached to the same cleats as the top cross panel, but do not epoxy the cleats to the middle disk at this time. Instead, use screws alone to hold this disk to the cleats so the disk can be removed later.

Step 3: The Card Cage

The card cage must provide mechanical support for the backplane. I've al-

lowed for either a PCB or wire-wrapped backplane (for those of you who want to do more work and spend more money). The backplane PCB screws directly onto the two remaining 6" cross cleats (thought I'd forgotten 'em, eh?).

The DC-DC power convertor board mounts on the other side of these cross cleats in the rear of the superstructure. Don't worry, I'll talk about both the power convertor and backplane PC boards in detail next time. For now, it's sufficient just to point out where they go.

Butt each cross cleat against the rear of its cross panel and clamp to the long cleats. In the case of the top cross cleat, the long cleats have already been attached to the middle disk. Do the drill, ream, and countersink routine, this time with a drill depth of $\frac{7}{8}$ ".

Secure each cross cleat with epoxy and a #6x $\frac{7}{8}$ " wood screw on each end. If you like, you can also put a line of epoxy on the butt joint between cross panel and cross cleat. To finish the cross panels, epoxy the card guides in place.

Step 4: Constructing The Sensor Platform

One problem I had with the sensor platform was shielding the sensors from stray infrared while leaving easy access for maintenance. The solution needed to be strong enough to allow carrying LIMBO by two grab handles on the top disk, too.

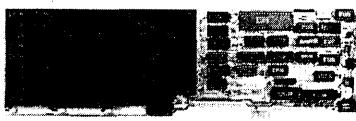
I must have thought up a dozen unsatisfactory methods, all too heavy, complicated, or expensive. Time to visit the local hardware store. The hardware store didn't have the answer, either, so I went to a wood working store. Furniture makers face this problem daily: how do you make a piece of furniture strong yet easy to break down into smaller, easy-to-carry sections?

The answer: "Knock Down" hardware. I chose a variety called the Trapez fitting, two mating plastic blocks with a single screw to hold them together. The Trapez is designed to join a vertical panel to a horizontal panel. The top block has sideways screw holes for mounting on the vertical panel; the bottom block has screw holes for mounting on the horizontal. (See Figure 3.)

When you remove the holding screw, the two panels come apart. Each fitting weighs a fraction of an ounce, so four of them are no problem, one for each corner of the X-dividers.

Angle brackets and epoxy per-

You've Seen Your Computer Run, **Now Watch It Fly!**



IBM-PC, XT, AT, '386 Blue Flame II SemiDisk Solid State Disk Emulator

Featuring:

- PC-DOS, MSDOS, and Concurrent DOS Compatible
- Very Fast Access: 6.4 Mbits/sec
- High Capacity: Up to 8 MB Per Board
- Expandable to 32 MB
- Battery Backup Option
- Hardware Parity Checking
- No Mechanical Wear
- No Special Interfacing
- Prices Start Under \$600.

SemiDisk
Systems, Inc.
11080 S.W. Allen #400
Beaverton, OR 97005
(503) 626-3104
FAX (503) 643-0625

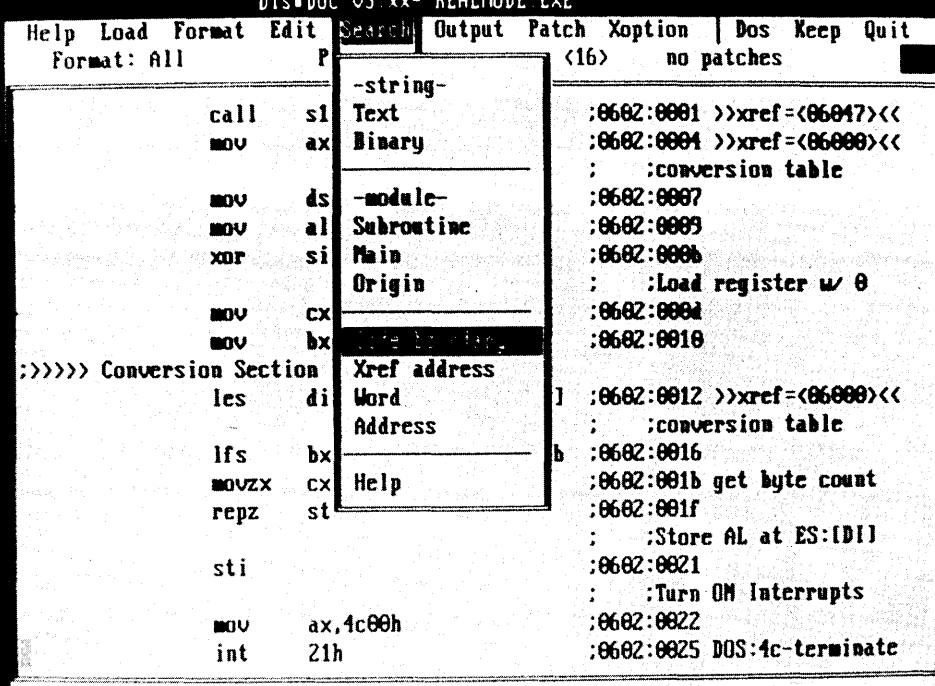
Reader Service Number 162

KNOWLEDGE=POWER

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

manently secure the top disk to the X-dividers, and Trapez fittings hold this whole assembly to the middle disk. Screwdriver access holes in the top disk aligned with the holding screws below complete the arrangements. And they only cost 70¢ each. Elegant, strong, cheap.

First mount the Trapez blocks onto the X-divider panels. Remove the screws from the middle disk and set the cleats and top cross panel aside. With the X-divider panel upright on the top of the middle disk, position the Trapez fitting squarely against both the vertical X-divider panel and the horizontal disk, then mark the hole locations of the top block with a pencil.

Use this same procedure to locate the holes for the angle brackets on the top

edge of each panel. Once you've marked all the hole locations, drill them out to $1\frac{1}{64}$ " dia. Lightly tap 6-32 Tee-nuts into each hole, then mount the Trapez fittings and angle brackets with 6-32x $\frac{3}{8}$ " flat-head machine screws. The brackets go on the opposite sides from the Trapez fittings; Trapez fittings go in the left and right bays, brackets in front and rear bays.

Assemble the dividers to check that the Trapez blocks sit flush with the bottom edges, and that the angle brackets are flush with the top. Epoxy the X-divider panels together, using a try square and masking tape to hold the panels square until the epoxy sets.

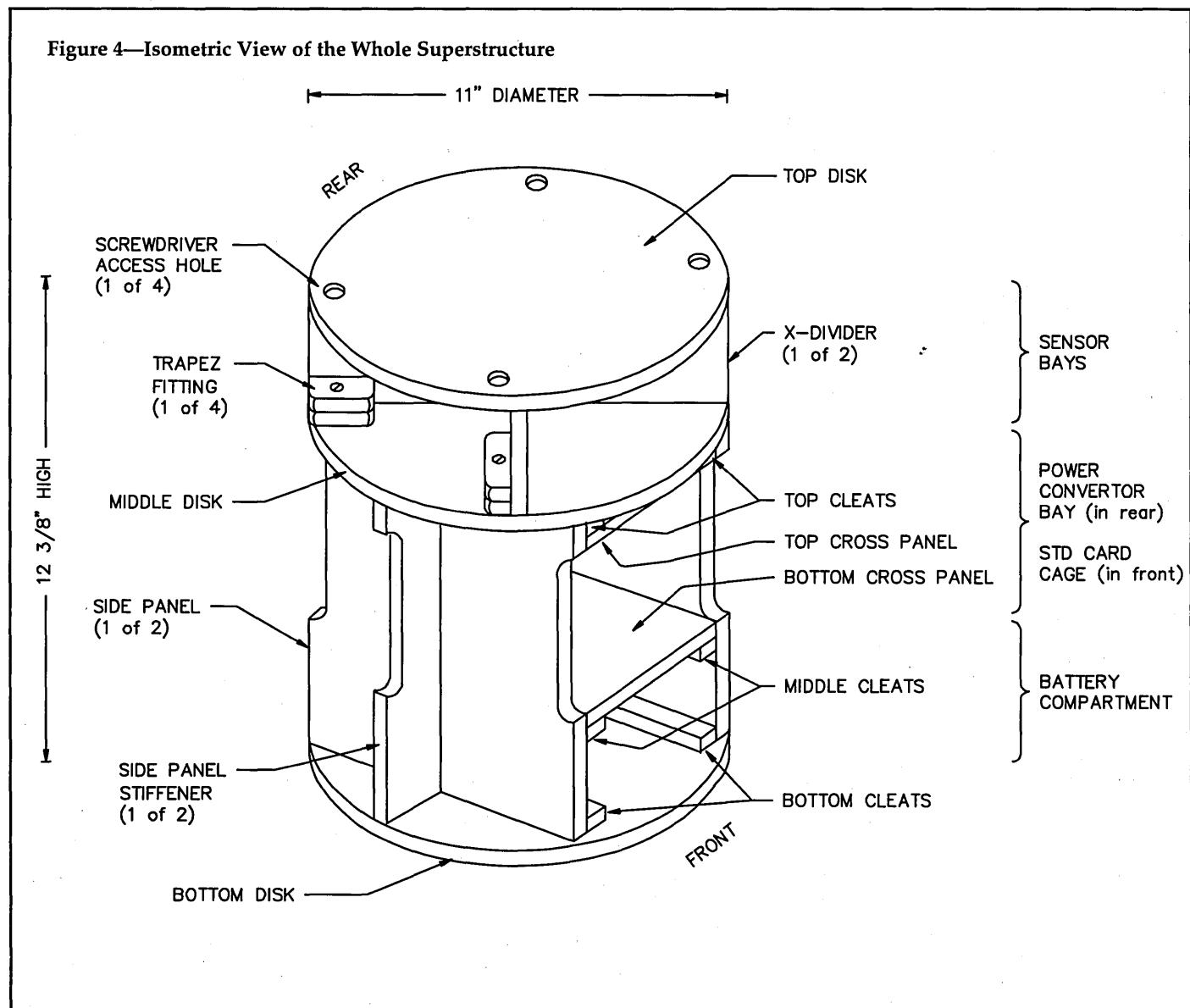
After the epoxy sets, position the X-dividers on the top surface of the middle disk. Carefully trace around the lower

blocks of each Trapez fitting, then remove the holding screws and use the lower blocks to mark the hole locations on the disk. Drill the mounting holes and mount the blocks with Tee-nuts and screws as above.

Try the fit between the X-divider and disk Trapez blocks. If it takes considerable force to mate all the blocks, you will have to enlarge one or more of the lower block mounting holes to allow the block(s) to move enough for easy mating.

Repeat this process for mounting the brackets to the top disk. Don't forget the screwdriver holes must be aligned with the holding screws. Check that this is right before you epoxy the X-dividers to the top disk. Mate the two halves of the sensor platform together and secure the holding screws.

Figure 4—Isometric View of the Whole Superstructure



Step 5: Finishing The Superstructure

Assemble the side panels to the top and bottom cross panels and the bottom disk. Use screws only for now. Sit the sensor platform on top. (See Figure 4.)

Notice that, with all the Tee-nuts protruding from the bottom of the middle disk, the middle disk no longer sits flush on its cleats. Use a Dremel tool or a chisel to cut shallow recesses into the side panels and cleats to accommodate the Tee-nuts. With some trial and error, you should get a good snug fit, especially when you tighten down the screws holding the middle disk to its cleats.

Mate the X-divider to the top disk and stand back to admire the completed superstructure. Well, almost complete. You still need to go back and epoxy the side panels and the middle disk to the cleats, but wait until after you install the STD backplane.

The last task is to mate the superstructure to the mobility base. Last time we drilled eight holes in the top (5 3/4" dia.) disk of the base, but used only four. Use the remaining holes to mount threaded studs that will protrude up through the bottom disk of the superstructure. Wing nuts hold the superstructure to the base.

The studs are merely 10-32x3/4" flat-head machine screws held captive to the base top disk by 10-32x1/4" Tee-nuts. The holes must be countersunk on the bottom side so that the screws are flush.

It is also necessary to counter-bore the other four Tee-nut holes on the top side of the base top disk as well as the mounting holes on the underside of the superstructure bottom disk with a 3/4" spade bit. The depth should be no more than the thickness of the flange of a Tee-nut. All this assures that the base top disk is in full contact with the superstructure bottom disk.

Making The Robot Move

Tools: 30 W soldering iron, VOM or DVM, needle-nose pliers, CMOS logic probe and/or oscilloscope, 5V and 6V power supplies, screwdriver.

Materials: Asst. electronic components (see Figure 5), silicone heat sink compound, masking tape, solder flux paste.

You can amuse your family only so long by pulling LIMBO around with a leash and making race car sound effects. Eventually they'll demand to see the robot move under its own steam. In steps LIMBO Stepper.

The LIMBO Stepper board is not the most up-to-date stepper motor controller, but it is cheap, rugged, and reliable, great features for any robot subsystem. (See Figure 6.) Karl Lunt has a fancier (and more expensive) stepper controller based on the 68HC705 for those of you who like smarter circuitry.

Three inputs control LIMBO Stepper's four output phases: STEP*, DIR*, and ENABLE. The asterisk in STEP* and DIR*

Figure 5—LIMBO Stepper Board Parts List

C1	100 mfd/50 V Electrolytic
C2	1 mfd/35 V tantalum
R1-R4	1 ohm, 1/4 W, 5%
R5	0.47 ohm, 2 W, wirewound
IC1	CD4070 Quad XOR
IC2	CD4027 Dual JK flip-flop
IC3	CD4093 Quad NAND, Schmitt trigger inputs
IC4	CD4584 Hex inverter/buffer, Schmitt trigger inputs
D1-D4	1N4005 600 V PIV, 1 A Silicon rectifier
Q1-Q4	IRF530 Power MOSFET
J101	10-pin male right-angle header, dual row
J102	8-position wire terminal block, 0.2" centers
J103	2-position wire terminal block, 0.2" centers
Misc:	Printed circuit board
	4 #6-32x3/8" machine screws
	4 #6-32 nuts
	4 heatsinks (AAVID #5751B)
	1 16-pin dip socket
	3 14-pin dip sockets
	Assorted wire jumpers
	♦ ♦ ♦

Figure 6—LIMBO Stepper Board Schematic

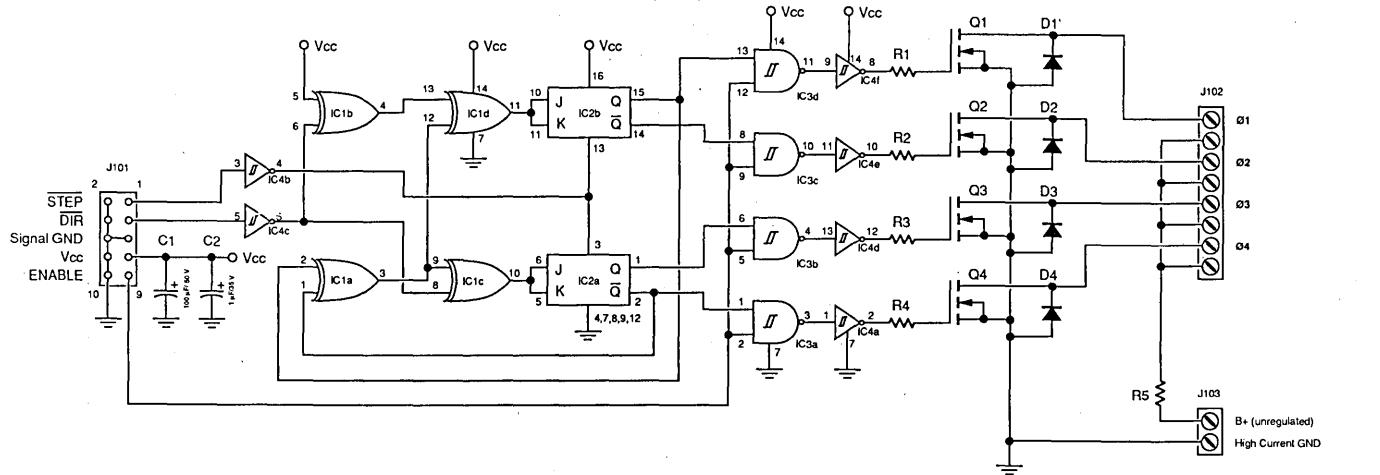
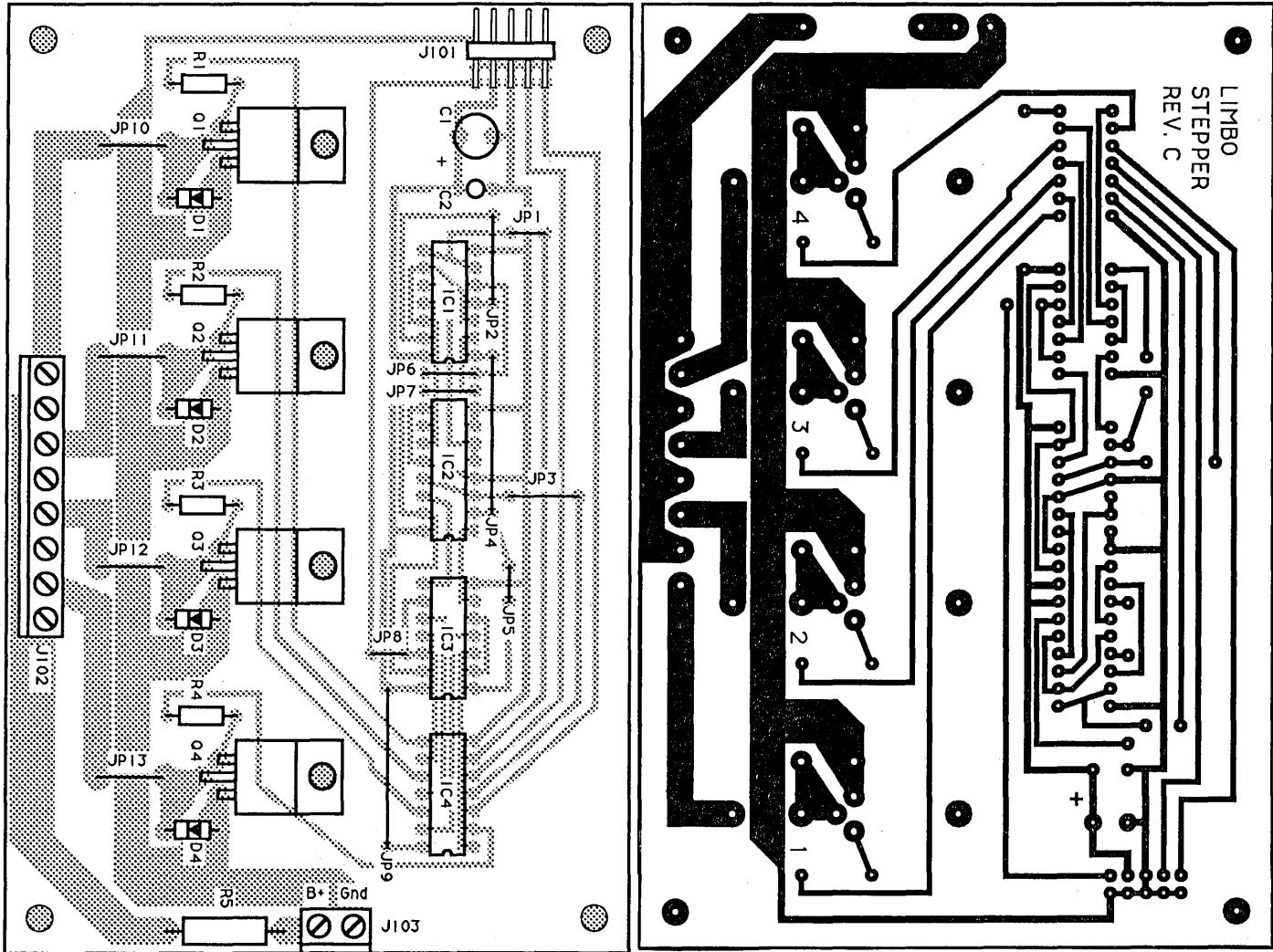


Figure 7—Stepper PCB Layout



signifies that these are active low signals. This is the convention used by the *STD Bus Specification and Practice Handbook*, so I'll use it also.

For each STEP* pulse, the stepper motor will rotate one step in the direction specified by DIR*. If STEP* is held steady, two phases of the motor will energize, locking the rotor. To conserve power, bringing ENABLE low allows all phases to deenergize, regardless of the states of DIR* and STEP*.

A dual JK flip-flop (in conjunction with a few XOR gates) provides all the sequencing. The NAND gates perform the ENABLE function, and four of the inverter/buffers drive the power MOSFET gates (gives you better rise times).

The 1 ohm resistors in the gate circuits prevent high frequency oscillation (ferrite cores are the ideal solution, but resistors

are cheaper). Power MOSFETs have extremely high current gains and can oscillate at several hundred MHz, which spells Smokey Trouble if not properly attenuated.

Use the other two invertors for their Schmitt trigger inputs to clean up the DIR* and STEP* signals should they pick up any electrical noise. The NAND gates also have Schmitt trigger inputs, so ENABLE gets cleaned up, too.

Overkill? Maybe. Noise pickup is surely the toughest problem to solve once you have it. Best to avoid it in the first place. Along the same line, I also used separate paths for signal ground and high current ground. In any system mixing high current circuits and logic circuits, it is essential to use a Single Point Ground (SPG), which means physically separating the current paths.

The traces and transistors on the board are sized to accommodate larger motors than those currently used in LIMBO, up to about 10 A per phase (with a higher wattage ballast resistor). With the 1 A per phase draw of the IMC stepper motor, the MOSFETs stay cool to the touch. Heatsinks are not required, but it's easier to put them in while building the boards than later when you want to use bigger motors.

I recommend using printed circuit boards for LIMBO Stepper. Point-to-point wiring works, but life is too short to continue using such time-consuming methods. I don't recommend wire-wrapping these boards because wire wrap connections aren't meant for high current.

For those of you who've never tried it, I encourage you to etch your own boards

from my layouts. (See Figure 7.) It's not that difficult, especially for a single sided board like this. Plus, you'll learn something.

An excellent tutorial on prototype PCB fabrication and design is David Kasten's *Electronic Prototype Construction* (H. W. Sams, 1987, ISBN 0-672-21895-X). The book covers everything from laying out boards to fabricating enclosures, and even such exotica as electroplating gold on edge connectors. This book belongs on every roboticist's bookshelf (or better, on his workbench). For you diehards, there are also etched and drilled boards available from Robotic Systems.

Stuffing the boards begins with the jumpers. I like to use the precut jumpers sold for solderless breadboards, but any 24 gauge solid wire will do. Next come the sockets. Simply hold the DIP sockets in place with a bit of masking tape while you solder. Solder the corner pins first, then work around to the other pins in a circular rotation. This will help you avoid overheating and delaminating any of the pads, and it helps prevent solder bridges.

Resistors and diodes come next. Watch the polarity of the diodes. Now install the MOSFETs and their heatsinks. Dab a bit of silicone heat sink compound on the back of each MOSFET before snapping its heat sink into place. Bend the leads with needle-nosed pliers, making sure that the tab mounting holes line up with the holes on the board when you insert the pins. Secure the heatsinked MOSFETs with 6-32x $\frac{3}{8}$ " machine screws and nuts, and solder the pins.

Finally, install the connectors and capacitors. These are taller components than the rest, so I save them for last.

With all the components soldered in, but with empty IC sockets, use a DVM to check continuity from the Signal Ground pin of J101 to the ground connections on the IC sockets. Check the socket contacts to make sure the solder joints are sound. Repeat this process with the +5V V_{cc} pin and the V_{cc} connections of the ICs.

Next check the Ohms reading between V_{cc} and Ground. If there is a low reading here, start looking for a solder bridge or a backwards diode. If you can't find it now, the thin curl of blue smoke when you first power up the board will point it out.

When the boards pass all these tests, then—and only then—install the ICs. Use the normal precautions for handling static sensitive components.

Editor's note: I know some of you will

wonder what "normal precautions" means, so: (1) don't shuffle your feet on the rug; (2) connect a piece of hookup wire between your watch and the board's ground; (3) don't hand an IC to another person without touching that person, skin to skin, first; (4) leave ICs in their tubes or foam until you're ready to install them; (5) don't slide ICs across the table top; and (6) don't give me static about not telling you.

To test the boards with the stepper motors, you'll need an adjustable squarewave generator that can output 5V CMOS levels from 0 to 1000 Hz. A bench function generator is ideal, but you can jerry rig a 555 timer to do the job. If you haven't the faintest idea what I'm talking about, I suggest you obtain a copy of Forrest Mims' *555 Timer Circuits, Engineer's Mini Notebook* at Radio Shack.

Connect the eight wires of the stepper motor to the screw terminals of J102. With power off, connect a 6V supply capable of sourcing at least 2 Amps to the B+ and High Current Ground terminals of J103. On J101 jumper ENABLE to V_{cc}, jumper DIR* to ground, connect V_{cc} and Signal Ground to a 5V power supply, then connect the squarewave output of your function generator (power still off) to STEP*. Now comes the big moment.

First, turn on the logic power supply alone, then turn on the function generator. Monitor the STEP* input with an oscilloscope or a logic probe to verify that STEP* is pulsing, then monitor the outputs of IC2, pins 1, 2, 14, and 15. On each, you should see a squarewave exactly 4x the period of the STEP* input.

If you have a two channel scope, you can verify that pin 2 is an inverted version of pin 1, as pin 14 is an inversion of pin 15. Pins 15 and 14 should be 90 degrees out of phase with pins 1 and 2, respectively. Now follow these signals on the schematic to the outputs of the NAND gates, where you will see inverted versions of the previous signals.

If you see no signals at these pins, check that ENABLE is tied to V_{cc}. If all is still well, follow farther to the right on the schematic. The signals are inverted once more by the inverter/buffers.

Once this all checks out, it's time to turn on the 6V power supply. The stepper motor should begin to rotate at once. Jumpering DIR* to V_{cc} should change the direction of rotation. If the motor just sits there and hums, try reducing the frequency input to STEP*. If the motor still doesn't rotate, or if it shudders and

turns erratically, check for cold solder joints. Also check that all the terminals are making connection to the stepper motor wires and that the wires are in the proper sequence on the J102.

If the motor appears completely dead, first disconnect the function generator output from STEP*, then jumper STEP* to ground. Try turning the shaft by hand with and without 6V applied to B+ of J103. With 6V applied, there should be firm resistance to turning; without 6V, there should be almost none. If there is no difference, check with a volt meter that 6V is making it past J103.

Check that 6V is also present on all the even terminals of J102. If everything checks out so far, measure Ohms from drain to source of each MOSFET with B+ disconnected. For those MOSFETs with 5V on their gate inputs, the On resistance should be about 0.18 ohm; for those with 0V on their gates, the reading should be several mega-ohms. Replace any MOSFETs that don't pass these tests.

If the board has been perverse enough to work perfectly right off, then you'll have missed all the above troubleshooting. Before you mount the boards on the superstructure, I advise you to go through all the tests, anyway. Troubleshooting electronic gear is good for your soul, and half of successful robotics is repairing the creatures.

Next time, we endow LIMBO with more sensors, more intelligence, and a DC-DC convertor.

Parts Sources

Nylon card guide—Part #770-4457
Concord
30 Great Jones St.
New York, NY 10012
(212) 777-6571

Trapez fitting—Part #03911
The Woodworker's Store
21801 Industrial Blvd.
Rogers, MN 55374
(612) 428-2199



Capturing & Graphing A Voice In Real-Time: Part 2

The Digital Half Of The Problem

Last issue Bruce gave us the analog portion of this often-discussed problem. This time he converts the analog signal to digital data and displays it.

While in graduate school, I took several classes in digital signal processing, digital filtering, and digital control systems. The classes always began with a professor or technical assistant saying, "sample at regular intervals."

While researching and writing this article, I naively assumed that any A/D board which contained a timer/counter could use that timer to start the conversion of the A/D converter. The timer makes sure the samples occur at precise intervals, something your PC can't guarantee (your program competes with the RAM refresh and clock interrupts).

However, a timer isn't always useful. My favorite board from Issue #49, the Real-Time Devices AD1000, has a real-time clock chip; unfortunately it can't be used to start A/D conversions. The Advantech board, sold by Rapid Systems and Halted Specialties, seems to be triggerable with a "pacer" clock. (This portion is so poorly documented it's not clear to me exactly how to do it, especially without a schematic.)

George Dooley at RTD explained that the industry-standard 12-bit A/D chip they used was designed in the days of the 8080 CPU. It uses two lines instead of one to start a conversion. One of these lines is an address from the CPU.

They've given long thought to running the converter with the timer/counter chip but haven't yet worked it out. They're designing a new board which uses a FORTH CPU and its own memory to store samples (and thus will not be subject to the vagaries of the

PC's interrupts). But it will cost three times more than the AD1000.

The upshot of all this is:

(1) Even boards which seem like they ought to be almost the same (i.e., the Advantech board and the RTD board) can function very differently;

(2) People buying the RTD AD1000 are using it for measurement, not digital signal processing or control;

(3) I'm doing my sampling in software, without the aid of the counter/timer.

George said he knew a fellow who used the AD1000 to sample music; apparently the sampling rate was high enough that the rattle in the sampling period (inconsistencies due to other demands on the system) didn't affect it. I've always heard the sampling period must be constant, though I'm not sure how constant.

My computer, a Standard 286 from PC Source, has two speeds: 6 and 10 MHz. Normally, I can't use 10 MHz because they used the wrong interrupt controller on the motherboard—it locks up during disk writes, tape backups, modem communications, etc. (PC Source was quite rude when I wanted it fixed, by the way, so I've been stuck in the slow lane ever since.)

Since I didn't use interrupts in this project, I was able to switch between 6 and 10 MHz and see a dramatic difference. In this case, *faster is better*; we can catch higher frequencies without aliasing.

Testing EOC

To sample, you start the conversion and then begin polling the end-of-conversion (EOC) line on the A/D converter. Once the conversion is complete you can read the value and start a new conversion.

To monitor end-of-conversion, strap the EOC line to the input of one of the

8255 parallel I/O ports. This is easy on the AD1000 board; just solder a small wire between two holes to select bit 7 of port A, B, or C.

I used port B since it isn't brought out to the external connector on the board, so none of those lines has been used. The 8255 defaults to input mode so you don't even have to configure the chip; just start reading it.

Code

I've separated the code according to function:

- Figures 1 and 2 (CAPTURE.H and CAPTURE.C) show how I tell the AD1000 hardware to capture a set of samples.
- Figures 3 and 4 (DISPLAY.H and DISPLAY.C) show how I translate data into screen graphics.
- Figure 5 (VOICE.C) declares the CAPTURE and DISPLAY functions by including their header files, and then uses the functions to capture your voice pattern and display it.

Figure 6 is the MAKEFILE which tells MAKE.EXE (which comes with Turbo C) how to assemble the project (look up "make" in the reference manual to understand Figure 6). By typing MAKE, you'll build VOICE.EXE.

I wrote all the code in Turbo C, but you shouldn't have too much trouble translating it to something else.

In CAPTURE.H, you'll see a set of C preprocessor macros for controlling the AD1000 board. I chose preprocessor macros over C functions because the macros put the code *in-line* when they're called.

The compiler gurus at Borland realized that if I/O takes too long, you may have to resort to assembly language. To solve this problem, they altered the way the compiler handles `inport()`, `inportb()`, `outport()`, and `outportb()` function calls. If

By Bruce Eckel

Revolution2
308 Meredith St.
Kennett Square, PA 19348

Figure 1—CAPTURE.H

```
/* CAPTURE.H by Bruce Eckel Revolution2 Real-Time
Consulting. Preprocessor macros & function
declarations for the RTD AD1000 board. Note that
for this code to work, the end-of-conversion (EOC)
line of the A/D converter must be strapped to the
parallel port B line 7 (PB7). This is accomplished
by soldering a short wire from the EOC line to the
PB7 line on connector P8 on the AD1000 board (this
connector is clearly marked by silkscreen info on
the board). This wire allows the code to monitor
status of the A/D converter by looking at PB7. */

#include <dos.h> /* So you get Turbo C import
and outport macros (fast) instead of functions. */

/* Macros in this file generate no code unless
they are called (thus you can put them in the
header file without causing multiple-definition
errors at link time). Since the compiler does all
the work of expanding the macro at compile-time,
you get readability and ease-of-use of function
calls, but the run-time speed of in-line code.
Because of the code generated by the compiler for
a function call, small macros don't necessarily
create more code than an equivalent function call.
Note the macros don't have semicolons after them;
this forces user to add semicolon so a macro call
has the same syntax as a function call. */

/* The following address is set by a bird jumper
clip on the AD1000 board. If you change the clip,
simply change the following (hex) number: */
#define base 0x200
/* Note that the compiler performs the addition of
this number in expressions like: base + 0xd so no
run-time overhead is incurred. */

/* This macro sets analog multiplexer address, so
you can select which input the A/D will read: */
#define MUX(channel) outportb(base + channel, 0)
/* (note that channels must be counted 0-7) */
/* Data written to the address is unimportant */
/* This macro starts a 12-bit A/D conversion.
```

```
Again, data written to the addr is unimportant: */
#define START_AD() outport(base + 9, 0)

/* This tests PB7. Is A/D conversion complete? */
#define COMPLETE() (inportb(base + 0xd) & 0x80)

/* This macro produces the value of the A/D
conversion. Backslash at end of the line allows
you to continue the macro on the next line. */
#define READ_AD(result) {result=(inportb(base+8) \
    << 4); result+=(inportb(base+9) >> 4);}

/* Note result is left-justified in the 16 bits of
the A/D registers. */

/* This macro starts a reading, waits till the A/D
completes the conversion, and reads the result: */
#define GET_VALUE(result) { \
    START_AD(); \
    while(!COMPLETE()) \
        ; \
    READ_AD(result); }

/* Here are declarations for functions defined in
CAPTURE.C. By including this header in another
file, you automatically declare these functions.
Thus, you can use the functions by including the
header and linking CAPTURE.OBJ at link time (see
the MAKEFILE for details). */
/* Display 16 bits of ones and zeroes: */
void print_binary(unsigned int);

/* Display value as a voltage between -5 and +5:*/
void print_value(unsigned int reading);

/* This function waits for A/D input to exceed
"threshold" and then fills "buf" with "bufsize"
samples, taken at software-controlled sampling
"rate." Note that the effect of "rate" will vary
depending on type and speed of your machine: */
void capture(unsigned * buf, int bufsize,
            unsigned threshold, int rate);

♦ ♦ ♦
```

you include the DOS.H file (as in CAPTURE.H), the compiler will replace any of those function calls with in-line code. Thus, the C code should be fairly optimal.

Study the code (and my comments there) for the details of what's going on.

A/D Converter Format

The 12-bit data must be read from the converter (a Harris HI-574A) in two bytes. The high byte, which you get by reading the board's base address + 8 (base+8 in the code), contains the high 8 bits. The low byte (base+9) contains the low four bits, shifted all the way to the left.

Thus the bottom four bits of the conversion are always zero. Notice that the preprocessor macro READ_AD() in CAPTURE.H fixes this by shifting both bytes to the right by 4.

The A/D converter could have read from 0 to +10 volts. RTD decided to have it read -5 volts to +5 volts. Thus, when the converter sees -5 volts, it outputs a digital 0; when it sees +5 volts, it outputs all ones.

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

Figure 2—CAPTURE.C

```
/* Uses the RTD AD1000 board to capture a set of samples. */
#include "capture.h"
#include <stdio.h>

/* Uncomment the following or define it on the compiler command line
   (see makefile) with -DTEST_BOARD to generate test code for a newly-
   installed AD1000 board: */
/* #define TEST_BOARD */

/* Print a value as 1s and 0s so you can see the bit patterns from
   the A/D converter: */
void print_binary(unsigned int binary) {
    int i;
    for(i = 15; i >= 0; i--)
        putchar((binary & (1 < i)) ? '1' : '0');
}

/* Print the AD1000 value as a voltage, so you can compare it with a
   voltmeter reading: */
void print_value(unsigned int reading) {
    if(reading & 0x0800) {
        putchar('+');
        reading &= 0x07ff; /* strip off "plus sign" bit */
    } else {
        putchar('-');
        /* Flip bits and mask them so a negative number near 0 becomes a
           small negative value: */
        reading = (~reading) & 0x07ff;
    }
    /* now scale and print the value */
    printf("%1.3f", 5.0 * ((float)reading) / ((float)0x07ff));
}

/* Capture a block of samples: */
void capture(unsigned * buf,int bufsize,unsigned threshold,int rate) {
    unsigned int sample;
    int i, j;
    /* First, wait until the threshold is exceeded: */
    while(1) {
        GET_VALUE(sample);
        /* High bit added to threshold to make it a positive A/D value: */
        if(sample > (threshold | 0x800))
            break; /* out of while(1) loop */
    }
    /* Then capture a block of samples: */
    for(i = 0; i < bufsize; i++) {
        GET_VALUE(buf[i]);
        for(j = 0; j < rate; j++)
            ; /* delay by a factor of "rate" between samples */
    }
}

#endif TEST_BOARD
/* The following program creates a digital voltmeter out of a selected
   channel of the AD1000. You should run this program and test a 1.5 volt
   battery, or a potentiometer connected to a DC power source, to insure
   that your AD1000 is installed and working properly. */
main() {
    const av = 10000; /* Averaging factor to remove noise */
    /* av=10000 gives incredibly accurate results (in complete agreement
```

```

with my voltmeter). Try smaller av to see rattle in your system. */
unsigned int sample;
long sum;
int i;
MUX(0); /* select channel */
while(1) {
    sum = 0;
    for(i = 1; i < av; i++) {
        GET_VALUE(sample);
        sum += sample; /* Add up a bunch of samples */
    }
    sum /= av; /* calculate the average */
    putchar('\r');
    print_binary(sum);
    putchar(' ');
    print_value(sum);
    if(kbhit()) exit(1);
}
#endif /* TEST_BOARD */

```

♦ ♦ ♦

Figure 3—DISPLAY.H

```

/* Declarations for functions to support graphic screen display of
voice patterns. */
#include <graphics.h> /* The Turbo C BGI header file */

/* "printf"-like function for printing messages while using the BGI in
graphics mode. */
void gprintf(char * format, ...);
/* Display the graphics "viewport" values (for debugging) */
void display_viewsettings(struct viewporttype view);
/* Types of scaling for display_series: */
typedef enum { quarter, half, full, twice, quad } scale_factor;
/* Display a series of numbers as dots on the screen: */
void display_series(unsigned * series, int series_size, int offset,
                    scale_factor vertical_scale_factor,
                    scale_factor horizontal_scale_factor, int color);

```

♦ ♦ ♦

This means that 0 volts produces a reading with only the high bit (bit 11, counting from zero) set. When the high bit is set, it's like turning on the plus sign. This, of course, is quite different from normal 2's compliment arithmetic. Notice the code in the function print_value() to account for the sign.

Capturing Data Points

The "magic numbers," which are the I/O port locations used in CAPTURE.H, come from the documentation for the AD1000. Notice that all the numbers are relative to a base address called (curiously) base. This address is set by a jumper on the board. If you change the

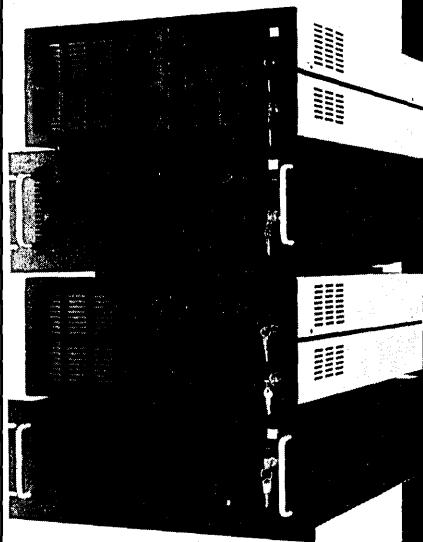
jumper, change the base. (Otherwise you won't get to first base.)

CAPTURE.C defines print_binary(), which prints a 16-bit value as a binary number (this often detects problems) and print_value(), which displays the value from the AD1000 as a voltage. By defining the name TEST_BOARD (see the MAKEFILE to do this on the command line; just type "make meter"), you can create a standalone program which pretends it's a digital voltmeter. Compare the output of the program with your standard voltmeter for verification.

CAPTURE.C also contains capture(), the function used in VOICE.C to capture a set of voice samples. Notice that you

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

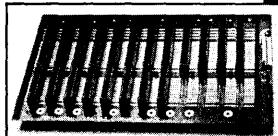
Designed to meet FCC

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

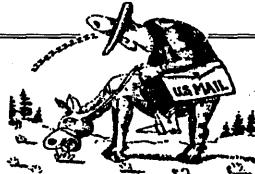
must pass the address and size of the buffer where you want the samples stored, as well as the threshold and the sampling rate. The threshold lets you cut off background noise (i.e., when you aren't speaking into the microphone).

Displaying Data Points

Figure 3 contains the declarations for the functions which display the data points. I used `gprintf()` and `display_view-settings()` for debugging and left them in for your enjoyment.

The important function is `display_series()`, which takes any series of points (not necessarily collected from the AD1000—you can create your own series). It then displays it after moving the whole group up or down by amount "offset," and scaling the output using `vertical_scale_factor` and `horizontal_scale_factor`.

The definition for `display_series()` is in Figure 4. I chose to make a local copy of the series before performing the scaling so the original series isn't touched. The call to the ANSI C library function `calloc()` allocates the space on the heap (which is later de-allocated with `free()`).



U.S. Postal Service Statement of Ownership, Management and Circulation (Required by 39 U.S.C. 3685) 1.A.)Title of Publication: MICRO CORNUCOPIA 1B.)Publication Number: 0747-587X 2.)Date of Filing: 9-29-89 3.)Frequency of Issue: Bi-monthly 3A.)Number of Issues Published Annually: 6 3B.)Annual Subscription Price: \$18.00 4.)Location of Known Office: 155 NW Hawthorne, Bend, Oregon 97701-2917 5.)Location of the Headquarters or General Business Offices of the Publishers: 155 NW Hawthorne, Bend, Oregon 97701-2917 6.)Name and Complete Address of the Publisher, Editor, and Managing Editor: Publisher: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917; Editor: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917; Managing Editor: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917 7.)Owner: Micro Cornucopia, Inc. PO Box 223, Bend, Oregon 97709-0223; David J. Thompson 1259 NW Iowa, Bend, Oregon 97701-1001; Sandra S. Thompson 1259 NW Iowa, Bend, Oregon 97701-1001 8.)Known Bondholders, Mortgagors, and Other Security Holders Owning or Holding 1% or More of Total Amount of Bonds, Mortgages or Other Securities: None 10.)Extent and Nature of Circulation: Average Number of Copies Each Issue During Preceding 12 Months A.)Total Number of Copies Printed: 29,104 B.)Paid Circulation: 1)Sales Through Dealers and Carriers, Street Vendors, and Counter Sales: 12,059 2)Mail Subscription: 9,555 C.)Total Paid Circulation: 21,614 D.)Free Distribution by Mail, Carrier, or Other Means, Samples, Complimentary, and Other Free Copies: 507 E.)Total Distribution: 22,121 F.)Copies not Distributed: 1)Office Use, Left Over, Unaccounted, Spoiled after Printing: 893 2)Returns from News Agents: 6,090 G.)Total (Sum of E, F1 and 2)-Should Equal Net Press Run Shown in A: 29,104; Actual Number of Copies of Single Issue Published Nearest to Filing Date: A.)Total Number of Copies Printed: 28,000 B.)Paid Circulation: 1)Sales Through Dealers and Carriers, Street Vendors, and Counter Sales: 17,396 2)Mail Subscription: 9,178 C.)Total Paid Circulation: 26,574 D.)Free Distribution by Mail, Carrier, or Other Means, Samples, Complimentary, and Other Free Copies: 274 E.)Total Distribution: 26,848 F.)Copies not Distributed: 1)Office Use, Left Over, Unaccounted, Spoiled after Printing: 1,125 2)Returns from News Agents: 27 G.)Total (Sum of E, F1 and 2)-Should Equal Net Press Run Shown in A: 28,000

Figure 4—DISPLAY.C

```
/* Functions to support displaying a voice pattern on the screen using
the Borland BGI graphics functions. */
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h> /* va_start(), etc. */
#include <string.h>
#include <math.h>
#include "capture.h"
#include "display.h"

/* This is a "printf"-like function for printing messages while using
the BGI in graphics mode. It is limited because it always starts at
the left and doesn't wrap at the bottom of the screen, but it's useful
for debugging. */
void gprintf(char * format, ...) { /* graphics printf for BGI */
    static int textline = 0;
    char textbuf[80];
    va_list argptr;
    va_start(argptr, format);
    vsprintf(textbuf, format, argptr);
    va_end(argptr);
    outtextxy(0, textline, textbuf);
    textline += textheight("x"); /* move to the next line */
}

void display_series(unsigned * series, int series_size,
                   int offset, scale_factor vertical_scale_factor,
                   scale_factor horizontal_scale_factor, int color) {
    int i, j, step;
    /* create a local copy of the array: */
    int * lseries = (int *) calloc(series_size, sizeof(int));
    memcpy(lseries, series, series_size * sizeof(int));
    /* First, perform vertical scaling of the local array: */
    switch(vertical_scale_factor) {
        case quarter: for(i = 0; i < series_size; i++)
            lseries[i] >>= 2; /* divide by 4 */
            break;
        case half: for(i = 0; i < series_size; i++)
            lseries[i] >>= 1; /* divide by 2 */
            break;
        case full: break; /* no change */
        case twice: for(i = 0; i < series_size; i++)
            lseries[i] <<= 1; /* multiply by 2 */
            break;
        case quad: for(i = 0; i < series_size; i++)
            lseries[i] <<= 2; /* multiply by 4 */
            break;
    }
    if(offset)
        for(i = 0; i < series_size; i++)
            lseries[i] += offset;
    /* Now display the array according to horizontal scale factor: */
    switch(horizontal_scale_factor) {
        /* display a part of the series in the whole space: */
        case quarter: for(i = 0; i < (series_size/4); i++)
            putpixel(i<<2, lseries[i], color);
            break;
        case half: for(i = 0; i < (series_size/2); i++)
            putpixel(i<<1, lseries[i], color);
            break;
    }
}
```

```

        putpixel(i<<1,lseries[i], color);
        break;
    /* display the whole series in the whole space: */
    case full:   for(i = 0; i < series_size; i++)
                  putpixel(i,lseries[i], color);
                  break;
    /* display the whole series in a smaller space: */
    case twice:  for(i = 0; i < (series_size/2); i++)
                  putpixel(i, lseries[i<<1], color);
                  break;
    case quad:   for(i = 0; i < (series_size/4); i++)
                  putpixel(i,lseries[i<<2], color);
                  break;
}
free(lseries); /* release local copy of the array */
}

void display_viewsettings(struct viewporttype view) {
/* The following macro was used during debugging to display all the
viewport settings. Notice the use of the new ANSI C preprocessor
directives: the "stringize" directive (#) which takes the argument
and makes it into a string, and the "paste" directive (##) which
takes two names and creates a variable name from them. Notice also
that in:
    #arg " = %d"
the preprocessor concatenates the two strings together. */

#define P(arg) gprintf(#arg " = %d", view.#arg)
P(left); P(right); P(top); P(bottom); P(clip);
getch();
}

```

♦ ♦ ♦

Figure 5—VOICE.C

```

/* Display a voice pattern using Borland BGI graphics functions. */
#include <conio.h> /* kbhit() */
#include <stdlib.h> /* calloc() */
#include "capture.h"
#include "display.h" /* includes graphics.h */

main() {
    int graphdriver = DETECT; /* will request autodetection */
    struct viewporttype view;
    int i, graphmode, color;
    unsigned int * points;
    /* Initialize graphics. The string in the third argument tells the
    function where to look for the graphics drivers (you may need to
    change it for your own environment): */
    initgraph(&graphdriver, &graphmode, "c:\\turboc");
    color = getmaxcolor(); /* works with all machines */
    getviewsettings(&view);
    /* display_viewsettings(view); */ /* for debugging */
    /* create an array according to size in pixels of the monitor: */
    points = (unsigned int *)calloc(view.right, sizeof(int));
    MUX(0); /* select the channel to read */
    while(!kbhit()) {

```

Continued on page 48

étude™

25 MHz 8-bit ANALOG-TO-DIGITAL CONVERTER

Based on the TRW THC1068-1 hybrid flash converter, its high signal-to-noise ratio yields excellent accuracy at the Nyquist limit.

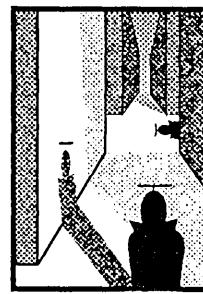
- 4 KB of cache SRAM or to host as converted at DMA speed
- I/O or DMA data transfer
- 10 MHz full-power bandwidth
- 3.92 mV resolution
- 16 jumper selectable base addresses
- External clock and trigger inputs TTL compatible
- Software source code included

\$495~~99~~

ALSO AVAILABLE AS A KIT FOR \$99, INCLUDING:

- Printed Circuit board
- Software
- Manual & assembly instructions

Requires: PC compatible 1/2 length 8-bit expansion slot. DOS 2.11 or greater. EGA, VGA or Hercules display needed for graphic representation of data.



Silicon Alley

P.O. BOX 59593
RENTON, WA 98058
206.255.7410

©1989 Silicon Alley Inc. étude is a trademark of Silicon Alley Inc. Other brand or product names are trademarks or registered trademarks of their respective holders. Prices and specifications subject to change.

Reader Service Number 177

The ANSI C library function `memcpy()` copies the series into the new memory. Use this process whenever you want to make a local copy of an array. Notice you must pass the size of the array into the function. Once inside the function, the array simply looks like an address, so there's no way to figure out its size.

Although `calloc()` and `memcpy()` are quite efficient, making a local copy of the array slows things down a bit, as does the scaling and offset process. I reasoned that data acquisition is the critical process, and display can be done at leisure. You can of course hard-code the display process, and make it much smaller and faster.

ANSI Preprocessor Tricks

The function `display_viewsettings()` in Figure 4 tells you how big the screen is, in pixels, using the `gprintf()` function (discussed later). The BGI figures out how big the screen is at run time, after it loads the graphics driver from disk when you call the BGI function `initgraph()`. The graphics drivers must be available; the program won't run without them.

You can make a viewport of any size, but the system defaults to full screen. Thus, by calling the BGI function `getviewsettings()` (as in Figure 5), you load the argument with the current screen parameters. `display_viewsettings()` will print this information.

I retain `display_viewsettings()` in order to use the new ANSI preprocessor functions. I find these tremendously helpful when debugging or doing any repetitive production of code. Notice the macro:

```
#define P(arg) gprintf(#arg " = %d", \
    view.##arg)
```

The ANSI C preprocessor will take `#arg` and "stringize" it; that is, it will take whatever argument is substituted for `arg` and turn it into a string. Now, since there is no punctuation between `#arg` and `= %d`, the preprocessor will perform string concatenation and turn the two strings into a single string.

Finally, the `##` is the token pasting operator. It will take the argument to the left and "paste" it to the argument on the right to create a new token. (Note: *not* a string, but something the compiler will treat as a name for a function, variable, `typedef`, `struct`, etc.) This means, for example, the macro call—

```
capture(points, view.right, 1000, 0);
for(i = 0; i < view.right; i++)
    points[i] >>= 3; /* do some rough scaling */
display_series(points, view.right, 20, half, full, color);
clearviewport();
}
closegraph();
}

◆◆◆
```

Figure 6—MAKEFILE for VOICE.EXE and METER.EXE

```
# Change these two paths for your particular installation:
# path for standard include files:
INCLUDE = \turboc
# path for libraries:
LIB = \turboc
# memory model:
MEM = s

# Following rule tells make how to create a .obj file from a .c file:
.c.obj:
    tcc -c -m$(MEM) -f -I$(INCLUDE) $*.c

# The main rule—this will be invoked if you simply type "make":
voice.exe: voice.obj capture.obj display.obj
    tcc -L$(LIB) -m$(MEM) voice.obj capture.obj display.obj \
        graphics.lib

# If you type "make meter.exe" the following rule will generate a
# program called CAPTURE.EXE which will test your AD1000 by turning it
# into a simple voltmeter, so you can compare the readings with a DVM.
meter.exe:
    tcc -c -m$(MEM) -f -I$(INCLUDE) -DTEST_BOARD capture.c
    tcc -L$(LIB) -m$(MEM) capture.obj
    ren capture.exe meter.exe
    del capture.obj

# These are the dependencies. When any of the files in the dependency
# list change, the .obj file is re-made according to ".c.obj" rule:
voice.obj : voice.c capture.h display.h
capture.obj : capture.c capture.h
display.obj : display.c display.h

◆◆◆
```

P(left);

will produce:

```
gprintf("left = %d", view.left);
```

As you can see, the additions to the preprocessor in ANSI C can be very useful.

Write Your Own printf()

The function `gprintf()` in Figure 4 is a

clone of `printf()`. At run-time, `printf()` analyzes its format string and goes through the arguments to print them according to the format. Normally, this would be a very unpleasant task to duplicate, and one I would never attempt. But the ANSI C library provides functions declared in `STDARG.H`, which make writing your own `printf()`-like function quite easy. `printf()` is one example.

The basic technique is this: define a `va_list` and pass it to `va_start()` along

with the address of the format string. Then you can either pick through the arguments yourself (see an ANSI C reference) or call the special functions `vprintf()`, `vfprintf()`, or `vsprintf()`, which are `printf()`, `fprintf()`, and `sprintf()` for variable argument lists.

I used `vsprintf()` to send the results to a buffer called `textbuf`. This is very powerful because it gives you all the output formatting abilities available in `printf()`.

After you're finished manipulating the variable argument list, you must call `va_end()` to clean everything up. After that, I call the BGI function `outtextxy()` to send `textbuf` to the screen in graphics mode. I keep track of the line I'm on with a static variable called `textline` (which keeps its value between function calls). This function could be cleaned up quite a bit for "real" use, but for debugging it's quite handy.

Let's Do It

Figure 5 is the `main()` function which uses the `CAPTURE` and `DISPLAY` functions to display your voice on the screen. The BGI function `initgraph()` initializes the graphics screen according to the argument `graphdriver`, which is set to auto-detect.

This means that `initgraph()` will determine which hardware you have and load the appropriate graphics driver from disk. It looks for the graphics driver in the path specified by the third argument.

To improve this program for general use, you may want to tell the user to set an environment variable to the path where the graphics drivers are, then read the environment and use that as the third argument for `initgraph()`. You can also fold the path in via a compiler command-line argument and change it in the makefile.

The BGI function `getmaxcolor()` returns the number corresponding to the "maximum" color available on that system. I've tried it on both my EGA (15) and Herc clone (1). Conveniently, the color always seems to be the equivalent of white, so it should work correctly with any PC adapter.

The BGI function `getviewsettings()` will place the parameters of the current viewport into the function argument (notice you're passing an address, so "view" is filled with the parameters). To find out the size of the screen, you just read `view.right` (for the width) and `view.bottom` (for the height). You can display all

these values with the `display_viewsettings()` function in Figure 4.

Because you never know until runtime how big your screen is going to be, you can't determine beforehand how big the buffer to hold your data points should be. Dynamic memory allocation comes to the rescue again—by using `malloc()` with the value of `view.right`, I get an array of ints which is the correct size for any screen.

The system spends most of its time in the while loop, capturing a series of points and displaying that series. When you press any key, the program cleans up (by calling the BGI function `closegraph()`) and quits. Try making the `display_series()` arguments adjustable from the keyboard.

Fast Fourier Transform

I considered doing an FFT on the waveform in order to see the frequencies in my voice, but I ran out of time. If you want to add an FFT to this system, an excellent reference is *Numerical Recipes in C*. (*Editor's note: Their recipe for digits in light oil (#2) is unreal.*) It includes the source code for an FFT in C plus a thorough explanation, so it shouldn't be too hard to adapt it to this project.

Help For Your Herc Clone

Three issues ago in the "parallel port file transfer" article, I noted that Herc clone cards have parallel ports which seem a bit tender (I blew mine out while trying to transfer a file). I couldn't find the chip which drove the port anywhere, and assumed it was custom (i.e., unavailable).

John Welch wrote to tell me he had also zapped his chip, a UMC 82C11, but that they could be obtained from:

John Kennaugh
5725 St. Charles Road, Suite 202
Berkeley, IL 60163
(312) 544-0120

Next Time

In the next issue (Object Oriented Programming), I'll be talking about Zortech's C++ version 2.0 and Borland's Turbo C++.



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++ & 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
308 Meredith Street
Kennett Square, PA 19348
(215) 444-0828
bix:Beckel cbs:72070,32256
net: 72070.3256@compuserve.com

Getting Started In Hardware

I've gotten a lot of requests lately for a real beginner's level hardware project. Okay, here it is, simple enough to warm anyone's irons.

If you work/play primarily with software, building electronic projects opens a new area of exploration. No libraries, no editors, no optimizing compilers...this is really low-level stuff. Here, you get your hands dirty playing with wires, resistors, and sockets. When I talk about tools, I mean tools.

Yet your entry into hardware construction need not be expensive. A couple of trips to the local Radio Shack and you could have your first project running for under \$20. That's probably less than you spent on your last "save the galaxy" game!

(Note: I will deal almost exclusively with Radio Shack in this article. I don't work for Radio Shack, in fact I don't know why anyone would want to work for Radio Shack, but they are everywhere. You can get your supplies a lot of other places; but if you ever get stuck on a Sunday afternoon and just have to have a 555 timer, I'll bet next week's lunch money you'll wind up at RS.)

First, Breadboard It

To get started in hardware, you need to build something. *Editor's note: this is a rite of passage.* So head on down to the Shack (or to your nearest hacker buddy with a well-stocked junk box) and get the following:

- 1 Modular IC breadboard socket (276-175 @ \$7.49 or 276-174 @ \$11.95)
- 1 555 timer IC (276-1723 @ \$1.19)
- 1 Pkg of jumbo red LEDs (276-041 @ 69¢)
- 1 Pkg of 10K Ohm 1/4-watt resistors (271-1335 @ 39¢)

1 Pkg of 270 Ohm 1/4-watt resistors (271-1314 @ 39¢)

1 22 µF electrolytic capacitor (272-1026 @ 69¢)

1 Pkg of 3 spools solid 22 Gauge hook-up wire (278-1306 @ \$3.49)

1 Pkg of 9v battery snap connectors (270-325 @ \$1.19)

1 9v battery

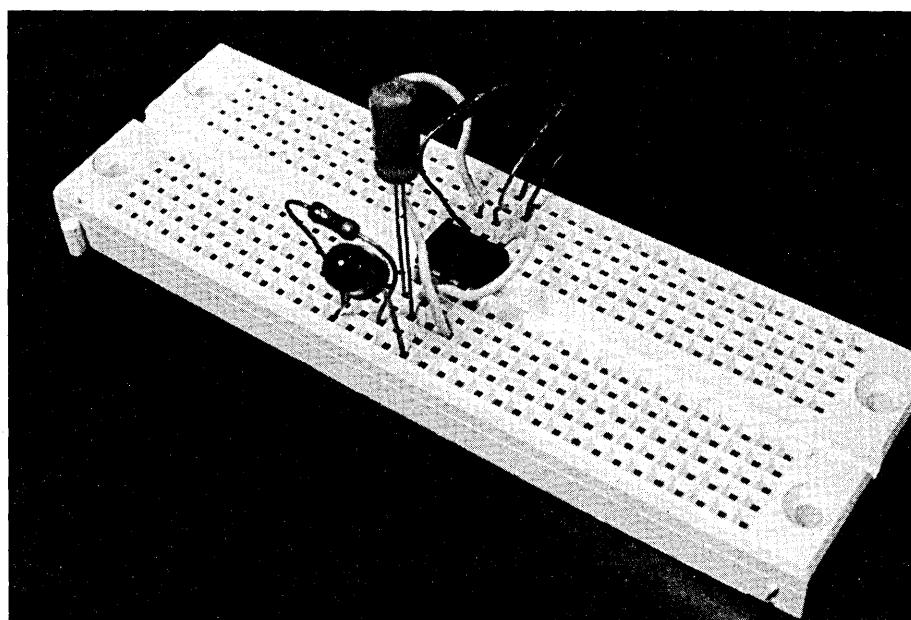
Your first project uses the 555 timer IC to blink a red light-emitting diode (LED). Building this circuit up on the breadboard lets you finish the thing quickly, rewire any errors with a minimum of fuss, and try changing components. It's a great way to start.

Begin by clearing off some table space, opening the breadboard and 555 timer IC packages, and taking a moment to look over the 555. Plug the eight metal pins of the IC into the breadboard so the legs straddle the wide gap down the center of the breadboard.

On the top of the 555, you'll find either an engraved dot next to one pin, or an engraved notch in the center of one edge. This dot (or notch) provides orientation information for most ICs; pin 1 is immediately next to the dot, and immediately counterclockwise of the notch (looking from the top of the chip). See Figure 1 for additional information.

Notice that with the 555 plugged into the breadboard, each pin occupies one hole out of a row of five holes. All five holes in a row are connected; anything plugged into an empty hole connects to anything else in that row. To connect a resistor or capacitor to one of the IC's pins, just plug one of the component's wire leads into a hole in the same row.

Refer to the schematic in Figure 2. First, plug one lead (either one will do) of a 10K resistor into a hole connected to pin 8 of the 555. Bend the leads of the resistor into a horseshoe shape and plug



The Author's Lash-up (er, Breadboard Version). Note the Careful Lead Dress to Reduce Spurious RF Emissions.

By Karl Lunt

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

the remaining lead into a hole connected to pin 7 of the 555. Now check the schematic to see how this resistor connection is pictorially represented; the sawtooth symbol drawn between pins 8 and 7 is the resistor you just plugged in.

Do the same with another 10K resistor, plugging it into the board so it runs from U1-7 to U1-6. (On the schematic, U1 means the 555, and the number after the dash refers to the pin number of the component.)

So cut a couple of 2" pieces of hook-up wire and strip $\frac{1}{4}$ " of insulation from each end. Connect one piece of wire from U1-2 to U1-6. The other wire runs from U1-4 to U1-8. While you're at it, notice how these wires appear on the schematic.

Now for the battery connector; make sure that both the red and black wires of the connector have at least $\frac{1}{4}$ " of bare wire showing. Do not connect the battery yet! Run the red wire to U1-8 and the black wire to U1-1. Notice in particular how the black (negative) lead of the

battery connector appears on the schematic; the downward-pointing triangle means ground. This is generally, though not always, the most negative point in the circuit. (I'm positive about this.)

Almost done! Now take a close look at the LED. Notice that the rim of the red plastic case has a flat spot next to one lead (this same lead may be the longer of the two, but no guarantee). The flat edge marks the cathode lead; this appears on the schematic as a flat bar. The other lead connects to the arrowhead.

Unlike wiring the resistors (which have no orientation), you must get the LED leads wired exactly as shown on the schematic. Plug the LED into the breadboard so the cathode lead (flat edge) fits into a totally empty row of pins. The other LED lead (arrow head) connects to U1-3.

Next, connect a 270 ohm resistor between the cathode lead of the LED and pin U1-1.

We have only one component left.

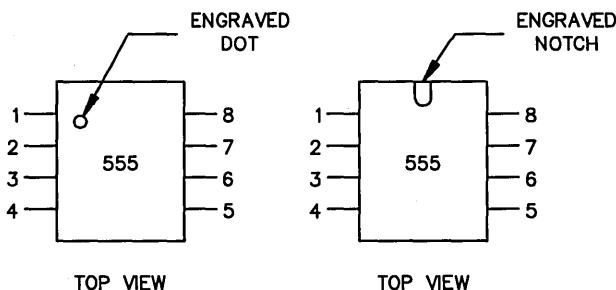
The 22 μ F electrolytic capacitor. Like the LED, it has two leads and those two are different. Wire it up backwards and it won't work (electrolytics can make strange noises and smoke and ooze if wired backwards).

The electrolytic cap has one lead marked with a string of minus signs and (generally) a stripe. You must connect this lead to the more negative circuit point of the two leads. The other lead of the cap matches the plus-sign shown on the schematic. Connect the negative lead of the cap to pin U1-1 and the positive lead to pin U1-2.

Take some time to check your wiring against the schematic and against the instructions. If anything looks wrong, it probably is.

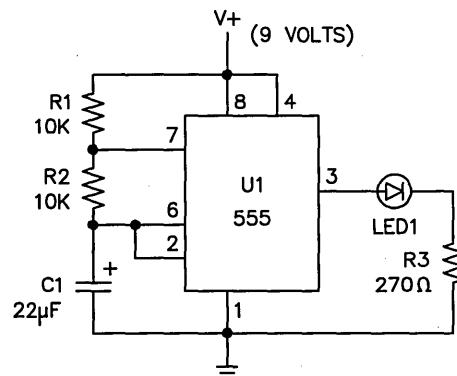
Okay, plug a 9-volt battery into the battery connector. The LED should blink about once per second. If so, congratulations! If not, disconnect the battery before something melts and go back over the circuit again. (Is your battery good?)

Figure 1—Pinout and Orientation of the 555 Timer



Using the dot or notch on an IC package to locate pin 1. This same identifying convention holds for all other DIP (dual-inline package) ICs.

Figure 2 —Schematic of a Basic 555 Oscillator



A simple LED blinder. R1, R2, and C1, as shown, will give a pattern of one second on, one second off. Change these values to see the results on rate blink and duty-cycle.

Once you get it working, think about the implications of what you just accomplished. No software, no compiler, no fancy debugger; yet the darn thing actually does something! I tell you, there are possibilities here. I'll bet someone could build a fairly complicated thingamajig that used no software at all!

But a breadboarded circuit has limited use; although you could put it into a small chassis and use it for whatever you need a blinking LED for, the circuit would not withstand much abuse. To do the job right, you have to do some soldering.

Your First Tool Kit

Just as with software, a set of well-designed tools makes any project easier. As a minimum, you'll need:

A small, low-power (10-25 watt) soldering pencil (or "iron")

A pair of needle-nosed pliers

A pair of wire cutters

A small spool of rosin-core solder

A pair of wire strippers (to strip insulation from hookup wire)

If you stay with hardware for long, you'll replace these first tools as you outgrow them. No big deal, you probably outgrew your first text editor, too.

For the soldering iron, try the Shack 64-2070 unit. Rated at 25 watts, it comes with a decent-sized tip (large enough for speaker wires, small enough for IC leads) and a little stand to hold the tip off the table. (The latter comes in handy when you're married.) If you stay with hardware long enough, you'll soon replace this open-top stand with a complete cage, but the stand will do for now. Price: \$6.29.

Be sure to pick up some rosin-core solder; 64-005 gets you 2.5 ounces for \$3.19. If you use up this much solder, you're either hooked on hardware or terminally clumsy. In the old days, you had to be careful about getting rosin-core versus plumbing solder; the latter contains an acid core that eventually destroys any connection you use it on. Today, you won't need to worry about this distinction if you buy from an electronics store.

The needle-nose pliers (64-1843 is a nice-looking pair at \$4.49) may be the most often-used item in your toolbox. Get a good pair. *Editor's note: These pliers can double as chop sticks when you're having canned Chinese food at home.*

The handle should fit comfortably in your hand as you open and close the pliers. If you have to rearrange your face while you pick up a wire, try another pair. Check that the tips of the pliers meet precisely.

Use the same care in selecting your wire cutters (also known as diagonal cutters; old-timers call them "dikes"). Shack's 64-1841 (\$4.49) matches the size of the needle-nose pliers above. Again, check the feel of the handle and make sure the cutting edges close completely when you clamp down. By the way, most dikes cut copper wire and other items equally soft. Use them on steel wire or nails and you will buy new ones.

Insulation strippers rank as a beginner's tool. Buy them to keep your frustration level down as you get started. If you stay with the hardware hobby, you will eventually use the dikes as insulation strippers. The Shack model 64-2129 (\$2.79) handles any wire from 10 to 24 gauge, an ample range for beginning projects.

What Were Those Terms?

When I started out in electronics, well-intentioned people used all kinds of terms I didn't know, taking my silence for understanding. Allow me to explain a few key words and phrases....

Insulation (don't laugh, I had to learn

this one the hard way!) is the plastic (or nylon, Teflon, or whatever) coating over a bare wire. The covering prevents that wire from contacting other conductors.

You can easily remove plastic insulation, by far the most common type, by using the insulation strippers mentioned above. Other types of insulations may require expensive heat strippers or similar tools. For now, stick to the plastic-covered hookup wire.

Gauge refers to the thickness of the bare wire, regardless the thickness of the insulation. This gauge number sometimes appears as AWG (for American Wire Gauge), as in 24 AWG. The larger the number, the thinner the wire.

For example, 30 gauge wire gets used often for wirewrapping digital circuitry. This wire measures 0.01" in diameter; its tiny diameter makes it easy to work with, though it cannot carry much current. A common size for two-wire power cords and speaker cable is 18 gauge wire; it measures 0.04" across and handles up to 16 amps.

Most simple hardware projects will use 22 or 24 gauge for "point-to-point" soldering, 18 or 20 gauge for power cables, and 30 gauge for wirewrapping.

Stranded wire refers to a single conducting wire made of many smaller bare wires, all covered together by a single layer of insulation. The power cords around the house use stranded wire to make them more flexible; without the flexibility, they would soon break. When you wire up a small project, you will normally use solid wire for the point-to-point connections and stranded wire for any cables that will be wiggled around often.

Gentlemen, Warm Up Your Irons

Soldering is like riding a bicycle—it's easy when you know how to do it.

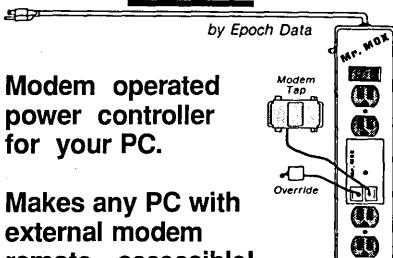
Start by measuring and cutting several pieces of solder exactly 1/4" long. Place one of these solder pieces on the table top in front of you. Clear your mind of distracting images and stare fixedly at this little bit of solder. Become one with the solder.

This represents the most solder you will likely need to make any single electronic connection.

Beginners cause more problems by using too much solder than by not using enough. Too much solder on a connection causes short circuits, poor connections (that don't hold as they should), and an ugly-looking product. Resolve

Mr. MOX™

\$99.95



Reader Service Number 135

now to be a craftsman in any hardware project you undertake; make each connection the best you know how to make.

Now plug the soldering iron in and let it heat up; this should take about five minutes. While you're waiting, get a small sponge (about 2" x 3" and about 1/4" thick), soak it in water and wring it out. If you can't find such a sponge, fold a clean cloth rag into a pad about 1/4" thick, then soak and wring that. In a pinch, you can also use a paper towel folded several times to get the right thickness.

Pick up the soldering iron as if it were a pencil, using whichever hand you use best. Carefully touch the tip to one of the small pieces of solder you cut earlier and watch the solder melt onto the tip of the iron. Then wipe the tip of the iron across the damp sponge to clean off most of the melted solder (this should only take one or two quick swipes).

Repeat the above steps two or three times for each side of the soldering iron's tip.

You have just cleaned and "tinned" your soldering iron tip, one of the most important steps toward making a good

solder connection. Get into the habit of tinning your iron after every two or three solder connections, or anytime it sits for more than a few minutes.

Editor's note: When I heat a new iron for the first time, I hold solder against the tip while it's heating. Heat causes oxidation on an untinned tip. Once it's oxidized, you have to file off the corrosion, perhaps damaging the iron tip coating. So, smother the tip in solder the instant it's hot enough. (Prevents oxidation.) Then wipe off most of the solder from the face of the iron you're going to use just as you're taking the iron to the work. That way the face will be bright and shiny clean. Some of the solder from the joint will cling to the iron, helping to keep it tinned.

Cut a three-foot length of solder off the spool and roll it into a convenient shape; I use a loose coil, like a coil of rope. Put the large spool of solder away; for now, you can use the short spool you just made.

Pick up both the soldering iron and the solder, then touch the solder to the tip of the iron. Allow only 1/4" of solder to melt onto the iron's tip. (You do remember what 1/4" of solder looked like after it flowed onto the tip, don't you?)

Clean the solder off the tip, using your damp cleaning pad.

Repeat these steps a few times, always trying to judge when you have used about 1/4" of solder. If necessary, cut off a 1/4" length of solder as before and melt it onto the tip again, to refresh your memory.

Now Add Some Wire

Use your wire-stripers to remove all the insulation from a 4" length of 22 gauge wire. Cut the wire off the spool at the insulation, so you end up with a length of bare wire.

Repeat this step with several 1" lengths of wire.

Now hold a 1" wire in your strength hand and the needle-nose pliers in your precision hand. Carefully bend the end of the wire into a small hook, just large enough to fit around the long, bare wire you cut previously. Loop the hook over the long wire (somewhere near the middle will do) and use the needle-nose pliers to clamp the hook closed.

If you do these steps properly, you should be able to pick up the 4" length of wire and the smaller wire will remain

QEdit™ ADVANCED

The fast, easy to use, fully featured text editor at an affordable price.



If you are looking for the right combination of price and value in a text editor, then give QEdit a try. At **ONLY \$54.95** and a money-back guarantee—you just can't go wrong.

QEdit is fast, easy to use, and simple to install. At the same time you get all of these features and more.

- Completely configurable, including keyboard and colors
- Edit as many files simultaneously as will fit in memory
- Open up to eight windows
- 99 scratch buffers for cut-and-

- paste or template operations
- Exit to DOS (or a DOS shell) from within QEdit
- "Pop-Down" menu system and customizable Help Screen
- Column Blocks
- Easy to use macro capability including keyboard recording
- Wordwrap and paragraph reformat capabilities
- Recover deleted text
- Automatic indentation for C programming
- Import files and export blocks
- Locate matching braces and parentheses
- Execute command line compilers from within QEdit



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

QEdit™ \$54.95
ADVANCED

ALSO AVAILABLE \$79.00
QEdit™ for OS/2

To order direct call
404-641-9002



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 and SemWare are trademarks of Applied Systems Technologies, Inc.
© 1989 Applied Systems Technologies, Inc.

firmly attached where you clamped it. If the smaller wire slips, spins, or falls off, redo the hook and reclamp the wire.

Practice this hook-and-clamp routine several times, using the 1" wires you had cut. Each time, make sure that the shorter wire clamps firmly onto the longer wire; if the connection is loose or shaky, redo it.

Now you get to try some real soldering. Clean and tin the tip of your iron. Pick up the solder in your strength hand and hold the iron in your precision hand. Select one of the clamped wire connections you just made as your first target. The next sequence of steps should all be done in about three seconds.

Firmly hold the iron's tip against the connection, touching both the long wire and the short wire simultaneously. Hold the tip on this connection for about one second.

Touch the solder to the wires at their junction and flow about 1/4" of solder into the connection. **DO NOT** touch the solder to the iron's tip! Remove the solder from the heated wires after you feel 1/4" of solder has melted into the connection.

Hold the tip against the soldered junction for an additional second, then remove the iron. **DO NOT** touch or move the wires, and do not blow on the heated connection. Wait about five seconds for the connection to cool. Clean your iron and set it back down on its stand.

Now take a moment to inspect your work. The tiny gaps in the hook should be filled with bright, shiny solder. There might even be a thin sheen of solder on some of the wires. You should clearly see the shape of the wires, as well as the hook in the connection.

If you cannot see the shape of the wires or of the hook, you used too much solder. Try another connection using less solder.

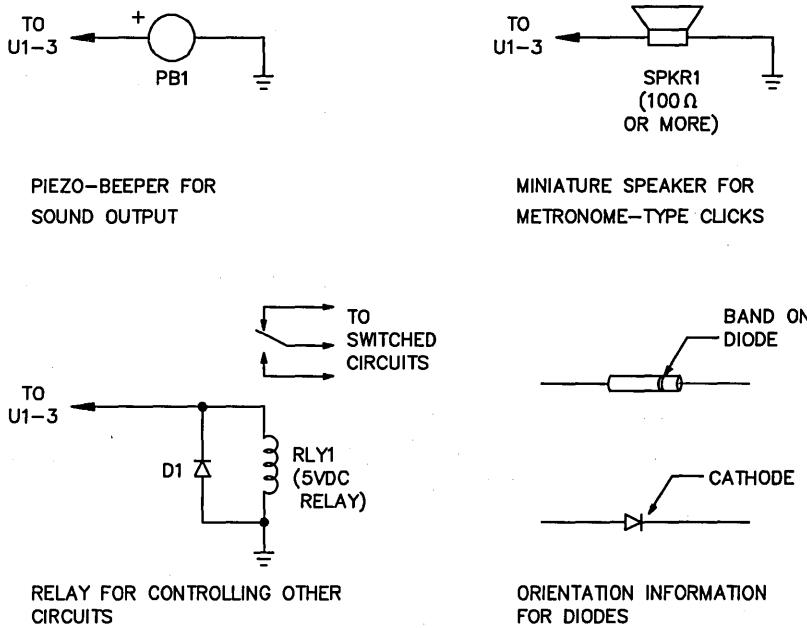
If you can see copper inside the hook, where the wires meet, you did not use enough solder, or you may not have left the iron on the connection long enough. Try another connection.

If the solder appears grainy or grey, either you moved the connection before it cooled or your iron's tip wasn't clean. Reclean and tin the tip, then try another connection. This time, let the wires rest a little longer before you touch them.

Caring For Your Soldering Iron

Be sure to clean and tin the tip of your iron regularly. Never use acid-core solder

Figure 3—Other Ideas for the Output Device



A short list of other output devices that can be controlled by the 555 circuit. The diode is intentionally connected backwards in the relay circuit: do not leave it out if you use the relay.

on your iron. Never use the iron's tip to pry a connection loose or press a connection closed, use your needle-nose pliers for that.

Always remember to unplug your iron when you finish your work; your iron's heating element (not to mention your house) will last longer. Always put the iron back onto its stand when you finish soldering.

Never, ever, use a file to clean the iron's tip. Except for a damp cloth, the only other cleaning tool you might possibly (I say, possibly) need would be a wire brush; and you would only need this if you got something really stubborn on the tip—say, human flesh.

Meanwhile, Back At The Blinker

Okay, now that you know how to solder, it's time to build a permanent version of your breadboard circuit. Start by picking up a couple more items from Radio Shack:

1 Pkg of 8-pin DIP solder-tail sockets (276-1995 @ 59¢)

1 Multipurpose predrilled project board (276-150 @ 99¢)

Some beginners look on sockets as an extravagance; after all, build the circuit

properly and you don't need sockets, right? But if you make a mistake....

The project board (and the other Radio Shack predrilled boards like it) may be the best experimenter's bargain around. In the old days, you had to bend leads and connect wires because the perf-board didn't have solder pads on each hole. These project boards, with pre-etched pads and channels, make wiring a snap.

Notice that the component side of the project board carries a silk-screened white copy of the copper pads that are etched on the solder side. You can use this layout guide to help decide where to mount components.

Start with the 8-pin DIP socket. Just like the IC that fits into it, this socket has a notch, dot, or slot for identifying pin 1. As you wire your circuit, treat this socket just as if it were the actual IC. Later, you'll plug the IC into the socket, using the same orientation, and your IC connections will be correct.

Working from the component (silk-screened) side of the project board, position the DIP socket so that the two rows of pins straddle the two long copper channels running the length of the pro-

ject board. Exactly where along the length of the board you place the socket isn't critical; near the middle will do. Push the socket flush against the board, hold it in place with one finger or a bit of masking tape, then turn the board over to view the solder side.

Notice that each pin of the socket connects to a three-hole row of solder pads, all joined with a run of copper foil. Looks something like the breadboard, doesn't it?

Using the needle-nose or a small-bladed screwdriver, carefully bend each of the socket's pins until it lies flat against the solder pad surrounding it. Make very sure that each pin touches only its own row of solder pads.

After bending all the pins into place, carefully solder each connection. Be sure you don't accidentally fill an adjoining hole with solder. Also, guard against putting too much solder on a connection and forming a solder bridge to a nearby pad. (Quiz: How much solder should you use for each connection?)

With the socket in place, add the resis-

If the solder appears grainy or grey, either you moved the connection before it cooled or your iron's tip wasn't clean.

tors, capacitor, and LED. Use the schematic and your breadboard to help you get the connections right. For each component, bend the leads until they fit properly into the holes (work from the component side). Cut the leads about $\frac{1}{8}$ " too long, then bend the leads over so

they contact their solder pads. Again, watch for shorts to nearby pads as you bend the leads. Solder the leads in place.

For the two connections that called for hook-up wire on the breadboard, you can run hook-up wire along the component side of the board to appropriate holes, then solder the wires. Or, you can run the hook-up wire along the solder side of the board, bending each end of the wire around the proper lead before soldering.

Double-check each connection, then again after you finish. If things look good, plug the 555 into the DIP socket, connect the battery, and watch the LED blink.

This circuit could become part of a robot, an alarm or signaling system, a model train layout, or any number of other projects.

Variations On A Theme

So you can blink an LED; now what? The 555 can supply up to 200 ma on pin 3. For some ideas, look at Figure 3.

The piezo-beeper (try Shack's 273-065)

It Isn't Just Another Fable

TurboFlow is serious flowcharting software at a price that won't cost you your kingdom. It's easy to use and more powerful than Merlin's magic.

TurboFlow runs on an IBM PC and features a complete set of ANSI symbols, is menu driven, interfaces with desktop publishing software, and supports a variety of printers and plotters.

So stop living in the dark ages. Call 1-800-882-5822 and order your copy, or ask for our free brochure.

TurboFlow & Logitech serial Mouse \$89
TurboFlow \$69

Legendary Software from Daytron Electronics, Inc. (214)669-2137
610 S. Sherman, Suite 104, Richardson, Tx 75081

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 Choisser 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.



Annabooks

12145 Alta Carmel Ct Suite 250-262
San Diego, California 92128

800-462-1042

In California 619-271-9526

Money-back guarantee

Reader Service Number 160

MICRO CORNUCOPIA, #51, Jan-Feb 1990 55

emits a 78 db squawk, enough to irritate most people. If you use a small speaker (about 100 ohms), you get a click, somewhat like a metronome. Or you can add a relay (such as 275-240 at \$1.99) to control devices needing more current or higher voltages than the 555 provides.

If you're adding a relay, be sure to include the diode (use Shack's 276-1101 at 49¢) as shown. It protects the 555 from the reverse EMF generated by the relay. Don't try to use this relay to control 120 VAC (house current) without some experienced help!

Figure 4 shows my working version of this circuit; the basic 555 schematic serves as a starting point. I use this version to help me with a simple home problem.

We have a freezer in the garage, connected to the same electrical circuit as the two upstairs bathrooms. Unfortunately, my wife's hair dryer occasionally trips the ground-fault circuit protector in the bathroom, shutting power off to the whole circuit. If we don't notice this in time (or if the breaker trips for any other reason), we stand to lose a freezer full of food.

Now I have a simple solution; I just leave the 555 timer circuit plugged into a bathroom outlet. If the power shuts down, the relay drops out, closing the connection on the 9-volt battery and starting the piezo-beeper. Resetting the GFCI in the bathroom or garage restores power, pulls the relay back in and shuts off the beeper. I just make sure the battery is fresh.

That's A Wrap

Now you know enough about hardware construction to do some simple circuits. If you like this type of hobby, start shopping for books and reading.

Radio Electronics magazine offers plenty of construction projects each issue. Although the circuits are more sophisticated than the 555 project just described, the magazine usually supplies a source for printed circuit boards and (occasionally) a complete kit of parts for each project. *RE* also runs columns on theory, practice, and tips for experimenters of all kinds.

Though catering to a very special audience, the American Radio Relay League (ARRL) handbook, published annually, is an excellent source of technical information. The book covers all types of electronics of interest to radio amateurs (hams), including antenna theory, trans-

mission/reception fundamentals, RF filtering, and (you guessed it) computers.

The 1989 edition contains many projects dealing with computers in the ham shack; stop by the local library and browse through a copy. Who knows, you might even decide to take up ham radio. Then you could finally find a use for all your computers.

Radio Shack (who else?) carries several little booklets written by Forrest Mims III. Forrest has written technical books of different sizes for years. These booklets run about \$1.49 and cover digital logic circuits, op-amps, communications, semiconductors....

He also has a 32-page booklet devoted to the 555. Each book carries one (or more) complete schematics per page, with parts lists and cautions where needed. For \$2.49 you can try his *Getting Started in Electronics*, a hands-on intro to hardware.

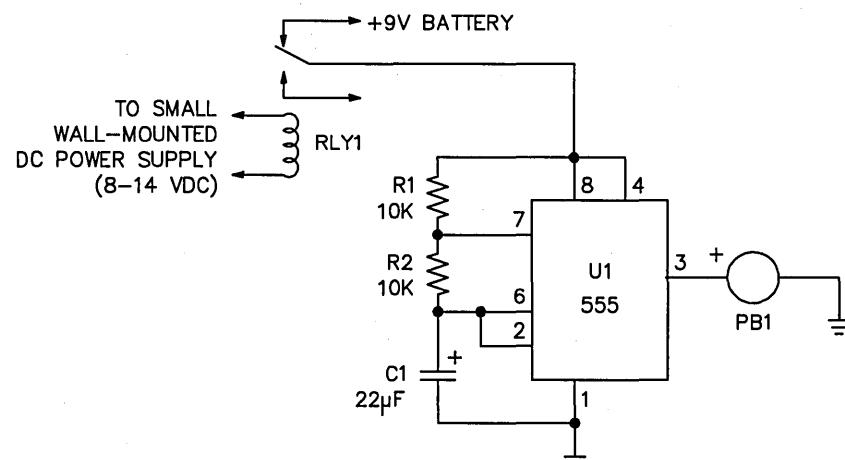
Without a doubt, the "most schematics in a single book" award has to go to *Modern Electronic Circuit Reference Manual*, by John Markus. Updated periodically, this monster manual contains hundreds of pages, each with several previously-published schematics.

Circuits come with a full parts list, a brief discussion of operation, and a reference to the magazine or book that originally published the project. Known simply as the Markus book, you can usually find it on the reference shelf of any large library. He's arranged the circuits by topic (burglar alarms, ultrasonic, power supply, audio amps, etc.) and the sheer volume of information will keep you busy for weeks.

Spend some time with hardware. After all, there was hardware long before there was software. You could end up with a new hobby, additional job skills, more respect for the abilities of the hardware designers, extra insight into the guts of your computer. Who knows, you might want to try building one of those babies yourself. It's really not that hard.



Figure 4—Practical Version of the Basic 555 Oscillator Circuit



This version uses a piezo-beeper, rather than an LED. The plug-in power supply can be any DC value, so long as it causes the relay to pull in when current is applied. Note that the 9v battery connects to the normally closed relay contact; this keeps the beeper quiet until the power into the wall-mount supply is shut off.



ZipGraph: Part 1

By Scott Robert Ladd

705 West Virginia
Gunnison, CO 81230
(303) 641-6438
BBS: (303) 641-5125

Scott begins his super-power, super-compatible graphics routines with this issue. He also gives us the latest on hot new packages from Zortech and Lattice.

It's fall in the Rockies as I write this. The summer residents have scurried back to warmer climes, and those of us remaining are bundling up. I'm sure you'll hear about Gunnison on your local news during the next four months; consistently, we have the coldest temperatures in the lower 48 states. Weathermen take a perverse pleasure from mentioning "and in Gunnison, Colorado, the low temperature today was a brisk 40° below zero!"

It doesn't feel that cold. There's no humidity. The sky is sunny 350 days out of the year here, so we have a constant supply of solar radiation. Since sunlight warms objects, it's quite possible to go ice fishing on Blue Mesa Reservoir in 10° below weather, without wearing a heavy coat. Strange, huh?

Planning for Rocky Mountain SOG II continues. Maria and I plan to hold RM-SOG II during the weekend of June 14-16 in 1990. We need to hammer out some exact details, but the conference is a go.

Maria and I are very grateful for the overwhelmingly positive comments we received from this year's attendees. Particular thanks go to Linda and Karl Lunt for their kind words in the last issue of *Micro C*. There's no way we could go through a summer without a SOG. You'll find SOG information in the sidebar.

Toolbox

As time goes on, things change. In our industry, the world changes on a daily basis, which can sometimes be disconcerting. Recently, a change in my programming has come to mean a change in this column. Yes, I have a confession to make: C is no longer my primary programming language.

I've found a new love, a programming lan-

guage that gives me everything C did—and more. My newfound favorite is C++, a language that has captured my heart and hard disk. At first, it was a tenuous relationship. I was hesitant to leave C, a language in which I had written so much. Yet C++ slowly won me over with promises of object-oriented programming and wonderful new capabilities. So, my friends, C++ will begin to creep into this column.

I firmly believe that C++ will replace C in most applications over the next several years. It offers so many advantages over C, and it currently has momentum. So, as time goes on, you'll see more C++ and less C. But don't worry—I'm not abandoning C, I'm just adding C++. As you'll see, this month's column discusses 850 lines of low-level code. Let's get to it....

C Explorations

As promised, this installment of the column begins the presentation of my auto-sensing graphics library for IBM PCs and compatibles. Called ZipGraph, it functions on several levels. We'll start with the low-level routines, which handle basic tasks such as determining the installed adapter type and the plotting of pixels.

Future segments will cover drawing primitives, clipping, region filling, and other basic tasks. In the highest level of this library, I'll present a series of C++ classes, built on the lower-level C code, which will handle advanced routines for ray tracing, 3D modelling, and animation.

An obvious question might be: why write a graphics module? Not only are there dozens of commercial graphics libraries for C, but most compiler vendors now include their own graphics routines. What does ZipGraph offer that others don't?

I had several goals in mind when building ZipGraph. To begin with, I wanted it to be fast. The current version is more than twice as fast as any other graphics library I've tried.

Surprisingly, my C-language version of the

low-level graphics routines was nearly as fast as the one I had built in assembly language. The advantages of having easily maintainable C source far outweighed the few percentage points loss in speed.

I also wanted to make ZipGraph portable. As time passes, more and more of my articles involve programs which do graphics. Alas, C compiler vendors do not have any interest in making their proprietary graphics libraries compatible.

If I write a program using the Borland Graphic Interface (BGI) included with Turbo C, that program won't compile with any other C compiler. Rather than shut people out, I decided to build a library which would compile under all the popular compilers.

The version of ZipGraph presented here compiles with Borland Turbo C 2.0, Lattice C 6.01, Microsoft C 5.10, QuickC 2.01, and Zortech C/C++ 2.01. In addition, the resulting object modules can link to Microsoft Fortran 5.0 and Stony Brook Modula-2 2.01.

Finally, I wrote ZipGraph because I find commercial libraries limiting. For example, most do not include printer routines, and some do not support certain graphics adapters. On top of that, they lack fundamental capabilities, such as a function to generate non-orthogonal ellipses.

No graphics library I know of completely supports ray tracing, animation, object rotation, and 3D plotting. Additionally, commercial libraries are written using C and assembly language only, without utilizing the object-oriented features of C++. As you'll see in future columns, C++ can do some fantastic things.

The ZG_LWLV module presented here is the base for all the other modules in the ZipGraph system. It uses

Rocky Mountain SOG II

June 14-16, 1990
Gunnison, Colorado

Last year's Rocky Mountain SOG was a great success. Maria and I, in a fit of unbridled optimism, have decided to hold a sequel. Yes, *Micro C* readers, there is a Rocky Mountain SOG II.

Thursday the 14th will begin with a choice of activities: a horseback ride into the rockies, or the traditional whitewater rafting. Thursday night, we'll have a barbecue at a national forest campground.

Friday and Saturday will again be filled with presentations. The focus this year will be on graphics, animation, and simulation. We already have talks on fractals, simulated ecosystems, and robotics. As always, we're looking for more speakers; please contact us.

If there's enough interest, we'll have a dealer's room this year. If you have something to sell, drop us a line. If you'd like to be part of a swap meet, ask us about that, too.

As always, this should be a fun time. SOG isn't just a computer conference; it's a place to meet friends old and new, enjoy the outdoors, and maybe even learn something.

The price schedule this year is:

Conference registration—

Before February 1st:	\$20
From Feb. 1 thru June 1:	\$30
After June 1:	\$40

THURSDAY—

Horseback riding trip:	\$25
All-Day Raft Trip:	\$60
Barbecue (adult)	\$10
Barbecue (under age 12)	\$6

SATURDAY Night Banquet:	\$15
RM SOG II Tee Shirt	\$10

Please register as early as possible! Some activities may fill early. Ask about family and group discounts! Speakers attend the conference and the Banquet free.

Remember: you only need to pay for those things you participate in. For example, if a family member wants to go to the other events but not the conference sessions on Friday and Saturday, they don't need to pay the conference fee.

Gunnison is located 200 miles southwest of Denver, and about 170 miles directly west of Pueblo on U.S. Highway 50. The Gunnison area is serviced by both United and Continental airlines. We also have bus service, numerous campgrounds, and many motels.

You can contact Rocky Mountain SOG II at:

705 West Virginia
Gunnison CO 81230
Voice: (303) 641-6438
BBS: (303) 641-5125

The BBS will contain the latest information. We hope you'll join us.



some of the more powerful features of C and provides the basic functions for detecting the type of graphics adapter. It also supports plotting and reading pixels on all common IBM adapters.

Editor's note: Scott's 850 lines of code just wouldn't fit in this issue. See "Around The Bend" for more on the problem of stuffing programming examples into Micro C.

In the meantime, the code is available on Micro C's BBS (503) 382-7643, and on the Issue #51 disk.

IBM PCs and their compatibles were originally designed to be modular. In spite of IBM's recent move toward building video hardware into the computer's motherboard, most vendors continue to let you install whatever video you want.

Currently, a PC will usually contain one of six standard video adapters: the Monochrome Display Adapter (MDA), the Color Graphics Adapter (CGA), the Hercules Graphics Card (HGC), the Enhanced Graphics Adapter (EGA), the Multi-Colored Graphics Array (MCGA), or the Video Graphics Array (VGA).

PC software authors have always faced the problem that any one of these video adapters can be installed in a PC. While the PC's BIOS supports most of the adapters, it does not support the Hercules card. I'm sure most of you have had the experience of finding a piece of graphics software which will not run on your computer.

Some graphics libraries allow you to detect which video adapter is installed. Borland's Graphic Interface (BGI) has this capability. However, the BGI uses external drivers, loaded at run time.

While you can convert the BGI drivers to object modules, you then have to go through a clumsy procedure to link them in and make the program aware of their resident status. Other auto-sensing packages fail to support certain adapters or are just plain clumsy.

I've always believed that computers should do work for you. ZipGraph not only detects the presence of all the above adapters, it also automatically sets up its routines so that you can write one program which works. The type of adapter you're working with is as transparent as possible.

Any source program using the ZG_LWLV.L module must include ZG_LWLV.H—the header file. ZG_LWLV.C contains the implementation of the module. Finally, ZGTEST.C

gives you an example of how to use the ZG_LWLV module.

We'll start with a quick synopsis of how to use the ZG_LWLV functions. First, you need to call ZG_Init to initialize the module. ZG_Init detects the adapter installed in the PC and saves its initial status. The public ZG_VideoInfo structure will return the information on the adapter.

You then need to set a graphics mode using the ZG_SetMode function. It sets the video mode you request, assigns the proper pixel plotting and reading function to the function pointers ZG_PlotPixel and ZG_ReadPixel, and returns information on the new graphics mode in ZG_VideoInfo.

You can then plot pixels by calling the function pointed to by ZG_PlotPixel and read pixels via the function pointer ZG_ReadPixel. When your program is done, it should call ZG_Done to restore the video adapter to its pre-programmed state. ZGTEST.C will show you the details of how this all fits together. Now, let's examine the above process in detail.

Identifying The Graphics Card

When called, ZG_Init attempts to identify the video adapter installed in your computer. There isn't any built-in way to determine the adapter type, but we can use the process of elimination.

ZG_Init begins by calling an MCGA and VGA BIOS function, which returns the adapter type. If an MCGA or VGA BIOS is installed, this call will tell us which one of those it is. If the adapter installed is not a VGA, the call to this BIOS function will fail.

Once we've eliminated the VGA, we call an EGA BIOS function. Again, if the EGA BIOS is not present, the function will not return expected values. If there's no EGA, we ask the BIOS for the hardware information word. We check the appropriate bits to see if we're dealing with a color or monochrome adapter. A color adapter will be a CGA at this point (since we have eliminated the other color adapters).

If a monochrome adapter is installed, our final task is to differentiate between an MDA and an HGC. We do this by monitoring the vertical sync bit of the monochrome card's status register; if the bit changes, we have a Hercules card.

Information on the adapter's type and its installed monitor is placed into

the ZG_VideoInfo structure. You can use constants for these values, defined in the ZG_LWLV.H file, to make your code a bit clearer. At this point, ZipGraph has only determined which video adapter you have.

I've tested the detection routine on several computers with a variety of adapters. So far, it has worked flawlessly with VGA, EGA, MDA, and Hercules adapters. I do not have a CGA or MCGA adapter, but I tested those routines on my Paradise 16-bit VGA card, which emulates the CGA and MCGA. I'd appreciate hearing from you if you have problems.

Selecting The Graphics Mode

To display graphics, you now need to set a graphics mode. Call the ZG_SetMode function and pass one of the ZG_MOD constants defined in ZG_LWLV.H as its first parameter. ZG_SetMode will return 1 if the requested mode is not valid for the adapter detected.

The public global variable ZG_VideoInfo will again return information on the mode. That information includes the x and y dimensions of the new graphics mode and the number of colors available. If all goes well, ZG_SetMode will return 0 to indicate success.

ZG_SetMode also does some automagical work for you. Two special graphics modes are ZG_MOD_BESTRES and ZG_MOD_MOSTCOLOR. Respectively, they represent—for the detected adapter—the best possible resolution, and the resolution providing the greatest selection of colors.

The global static array VideoTable contains the actual modes for both of these special modes, for each adapter. In addition, VideoTable contains a bit mask which indicates the valid modes for a given adapter. The requested graphics mode is compared against the ModeList value for a given adapter type to make sure it is valid. If it isn't, ZG_SetMode returns an error.

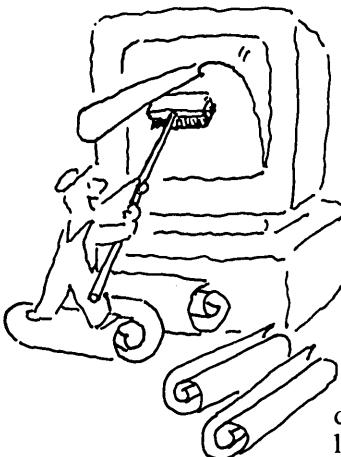
Once it has determined the validity of the requested graphics mode, ZG_SetMode uses the information stored in the global static array ModeData to do the mode set-up. ModeData contains the equivalent BIOS mode for the mode requested, the addresses of the appropriate pixel plotting and reading functions, and the dimensions and color counts for each mode.

When the mode is set, ZG_SetMode

Two great tools.

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



ORDER NOW. YOU RISK NOTHING.

Thousands of satisfied customers have already ordered from us. Why not visit your dealer or call toll-free, right now and put Saywhat and Topaz to the test yourself? They're fully guaranteed. You don't risk a penny.

Special limited-time offer! Save \$26. Buy Saywhat?! and Topaz together for just \$99 (plus \$5 shipping and handling, \$10 Canada, \$25 International, Calif. residents add 7%).

Visit your nearest dealer
or call toll-free:
800-468-9273

In California: 800-231-7849
International: 415-571-5019

Software Science, Inc.
100 Valley Drive, Brisbane, CA 94005

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

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

TOPAZ

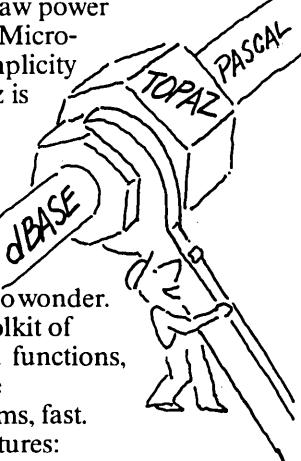
The breakthrough DBMS toolkit for Pascal

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 filenames 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!

SOFTWARE SCIENCE INC.

assigns the appropriate values in the ModeData table to the function pointers ZG_PlotPixel and ZG_ReadPixel.

Thus, for each graphics mode, the correct pixel plotting and reading routines are set—automatically.

Once that is done, ZG_SetMode assigns values from the ModeData table to the width, height, and color count values of the ZG_VideoInfo structure.

You should note here that ZG_LWLVLC handles the Hercules card in a special manner. Since the PC BIOS does not support the HGC in any way, its graphics and text modes must be set via special functions. The Zip-Graph functions make exceptions to their normal set-up by calling the special Hercules functions rather than the BIOS.

Plotting Pixels

Different video modes require different pixel plotting and reading routines, which is why I use pointers to the functions for these tasks.

You can use ZG_PlotPixel and ZG_ReadPixel exactly as if they were regular functions. However, what they do depends on the graphics adapter. Their value is based on the graphics mode you requested via ZG_SetMode. In a way, you can look upon this as a form of polymorphism.

This column has already run far longer than it should, so I won't go into the specifics of how the individual pixel plotting and reading functions work. If you're dying to know, pick up a copy of Richard Wilton's *Programmer's Guide to PC & PS/2 Video Systems* (ISBN 1-55615-103-9), by far the best reference ever published about the nuts and bolts of PC graphics programming.

I doubt ZG_LWLVLC will be static. For instance you could add support for the super VGA modes, such as 800x600. Additionally, you might like to try some of the non-documented VGA modes, such as 320x400 with 256 colors. But all in good time. I suspect this issue's source code is more than enough for most of you to chew on for a while.

News and Reviews

Everything comes in bunches. Two significant C-related products have arrived on the scene—Zortech C++ 2.01 and Lattice C 6.01.

Zortech C++ 2.01 will have been released by the time you read this. As

most of you know, I'm a big fan of the compilers written by Walter Bright, and an even bigger fan of C++. I beta-tested this latest release and can frankly say that this is one of the finest compilers ever written. It's one of the fastest compilers I've seen, and it produces very tight and reliable code.

Probably Zortech's biggest feature is its near-perfect implementation of AT&T C++ version 2.0. C++ 2.0 is probably the most amazingly complex language ever developed, and Zortech is the first MS-DOS vendor with a native-code compiler for it. I used a beta copy of Zortech C++ 2.01 while writing a book on C++ 2.0. The compiler's capabilities were very impressive.

The new Zortech compiler is missing a few C++ 2.0 features. For example, it does not support pointers to member functions yet. A release due sometime early next year will correct that. Otherwise, this is a faithful implementation of the language AT&T describes in its reference documentation. Even AT&T's own C++ translator, called cfront, does not support the full language.

Zortech C++ 2.01 comes in two packages. You can buy just the compiler for \$200, or you can purchase the "Developer's Version" for \$450. The latter contains the compiler, complete source for the run-time library, a set of C++ classes, and a C++ source-level debugger. Zortech no longer sells the vanilla C compiler separately; however, the C++ products still include it.

Another hot entry is Lattice's long-awaited upgrade. Going from version 3.4 to version 6.01 may seem a bit strange. While Lattice justifies the radical change in version numbers with some mumbo-jumbo, I suspect the real reason is that they wanted to impress upon people that Lattice C has *really* been upgraded.

Back in the early days of the IBM PC and its clones, Lattice C was the king. In fact, early versions of the Microsoft C compiler were licensed versions of Lattice. Then Microsoft wrote its own C compiler, and other vendors entered the market with faster, optimizing packages. Lattice C quickly slipped into obscurity. Now, they've returned with one of the best packages on the market.

For \$250, you get a globally optimizing compiler, a MASM-compatible assembler, a very good full-screen debugger, OS/2, Microsoft Windows, and MS-

DOS compatibility, and dozens upon dozens of tools and libraries.

In fact, the Lattice library is one of its most outstanding features; it contains more than 800 functions! To do this, Lattice simply bundled their dBASE III interface library, curses library, graphics library, and communications library with the compiler. Previously, these had been separate products.

Editor's note: The curses library? You mean the room where engineering and marketing discuss the features in the next revision?

Of course, a compiler cannot be judged wholly upon the size of its library. Lattice C 6.01 is a very good compiler; it produces competitive code both size- and speed-wise. It doesn't win all the benchmarks, but it certainly does better than average on all of them.

The compiler itself is reasonably fast, and Lattice supports several C-language extensions aimed at imbedded systems programmers. It's impressed me enough to install Lattice C 6.01 alongside the Borland, Microsoft, and Zortech compilers.

Resources

You may have noticed the BBS telephone number under my byline. Yes, I've put up a BBS, called Duck Tower (a menu selection on the BBS explains the name). It's at (303) 641-5125, and works at 300/1200/2400 bps, 8 data bits, 1 stop bit, and no parity. It's currently a node on the FidoNet system: 1:104/708.

The BBS is oriented towards my favorite subjects: science and computers. I try to maintain the best in MS-DOS applications and utilities, too. Programming languages supported are C, C++, Smalltalk, Pascal, Modula-2, and 80xxx assembly language.

I also have what I believe to be the world's largest collection of fractal generators and Chaos Theory programs. And I carry the issue disks for Micro C, along with the latest versions of Zip-Graph, CRITTERS, and the other software you've seen here. So come on over and visit; we'd love to have you!

Movin' On....

That's it for now. Next time, we'll look at line and curve drawing, followed by a 3D plotting module used to implement the fractal generator.



ERAC CO.

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

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 Wytek Co-Processor slot. Shadow RAM.

-20 \$630 -24 \$660
 1 meg SIMMS \$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

ELGAR

UNINTERRUPTIBLE POWER SUPPLIES

400 Watt MODEL IPS400 + \$650

Power distribution center and sine-wave UPS. Only 2" high.

560 Watt MODEL IPS560 \$350

Sinewave, 560W complete with batteries.

400 Watt MODEL SPR401 \$180

Supplies may have minor cosmetic damage, but are electrically sound. Squarewave output. Run on internal or external 24VDC battery when line goes down. Typical transfer time = 12MS. Battery supplied. For AT, XT & Kaypro.

★
NEW 24V INTERNAL BATTERY \$75

NiCds

AA Cells .6ah.....	\$1.00
12V Pack AA Cells .6ah	6.50
Sub-C Cells 1.5ah	1.50
12V Pack Sub-C	10.00
Double D Cell 2.5V 4ah unused	8.00
C Cells	1.75
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 ft. length.....	2.00
High Voltage Power Supply Input: 15-30V DC Output: 100V 400V 16KV	6.50

TERMINALS

Televideo 925	\$99
---------------------	------

KAYPRO EQUIPMENT BARGAINS We Repair CPM Kaypros

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

CPM COMPUTERS

K4-84 425 K4X 425

★
**We carry all IC's
for Kaypro repair.**

IC'S

81-189 Video Pal	\$15.00
81-194 RAM Pal	15.00
81-Series Char. Gen. ROMs	10.00
b81-Series Monitor ROMs	10.00

TEST EQUIPMENT

OSCILLOSCOPES

TEK 7403N/7A18N/7B50A 60 MHz	\$650
Leader LB0520 30 meg Dual Trace	300
TEK 475 Dual Trace 200 MHz	1499
Scope Probe x1, x10 100MHz	25

ANALYZERS

TEK 491 10MHz - 40 GHz	\$4000
Biomation 805 Waveform Rcrdr	195
Biomation 8100 2-Channel Waveform Recorder	295
HP1600A Logic Analyzer	295
HP1600A/1607A Logic Anlyzr	495
Gould K40 32 CH Logic Analyzer	950

MISC.

Optronics 550 MHz Freq Cntr	\$95
Exact 566 VCF/Sweep Function Generator	195

HOURS: Mon. - Fri. 9 - 6 — Sat. 10 - 4

MINIMUM ORDER — \$15.00

TERMS: VISA, MasterCard, Certified Checks, Money Order, NO COD. Visa and MasterCard add 3%. Personal checks must clear BEFORE we ship. Include shipping charges. California residents add 7% Sales Tax. For more information please call.

CPU & RAM & MISC.

41256-12	\$6.50
4164-10	2.50
4164-15	2.00
MK48Z02B-20	10.00
Dallas D1220Y	10.00
SIP DRAM 256-12	7.00
2716	3.50
2764	4.00
27256	5.25
MC68000-8 CPU	8.00
Z80 CPU75
Z80 CTC	1.50
Z80A PIO	2.00
8089-3	6.50
80C85A	4.50
8088	6.50
8212	2.00
8251	1.50
8255-2	3.50
D8284A	2.50
D8749	7.00
6845	5.00
1793	6.00
1797	7.00

SWITCHERS

AT 200W Pulls, tested	\$35.00
5V/75, 12V/8, -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 12V in 5/8, 12/5, -5/3 out	45.00

★ SPECIAL ★

VERSATEC 8222F 22"
Electrostatic Printer Plotter

200 dots per inch. Up to D size.
1" per second \$3,999

★ SPECIAL ★

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 \$99

★ SPECIAL ★

COLOR MOUSE — Red, Green, Yellow,
Blue or Standard with manual &
diskettes. Lifetime warranty.

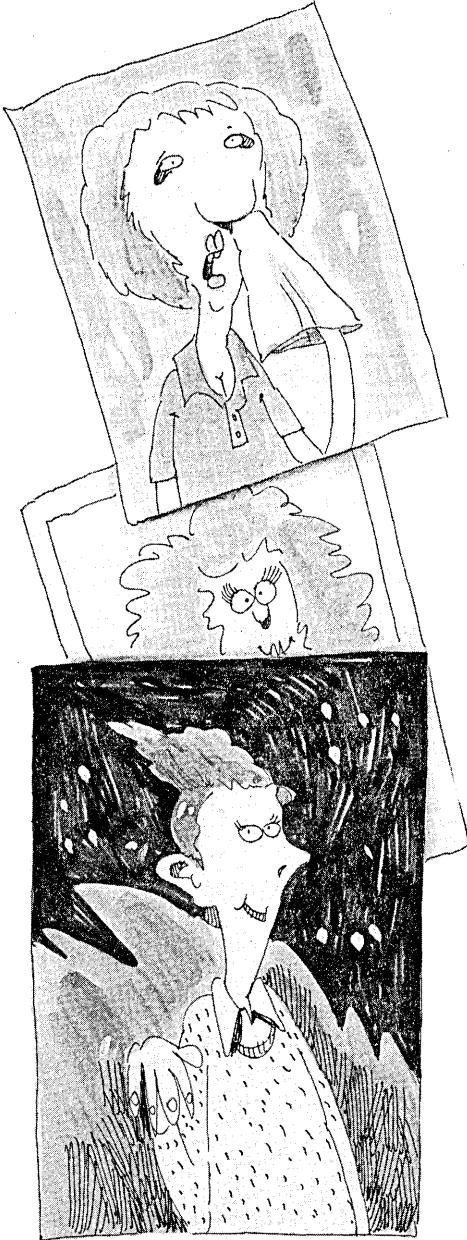
\$39

Reader Service Number 93



CULTURE CORNER

The Things Up With Which I Have To Put



I had just staggered in the door from another grueling vacation. The latest issue of *Micro C*, hot off the presses, sat in newly opened boxes. An exciting time, this; we catch our first glimpse of the end product of months of hard work (well, of months anyway).

I snagged a copy, settled in next to Cary's desk, leafed through the pages, and found an insert. An insert? *Micro C* never has inserts. It read:

SWM, WHO LOVES THE GREAT OUTDOORS: Rock Climbing, Mountain Biking, Snow Camping. Ex-physicist turned computer technician (nerd) seeks sweet young thing with amazonian strength to share the milk of Mother Earth. Must be petite, healthy, possess good genes, and the ability to carry a 50 pound pack up steep terrain. Must speak only when spoken to and, when permission is given, be independent. Please respond with photo and vital statistics to: Larry Fogg, at Micro C, P.O. Box 223, Bend, OR, 97701.

(If you wear L.L. Bean jammies and breathe heavily, call evenings.)

I looked around at the innocent faces and knew something was wrong (no one ever looks innocent around here unless they've hatched some sort of devility). The entire untrustworthy crew of *Micro C* wanted me to believe that they had sent out this implausible (but nicely worded, I had to admit) personal ad with each copy of the magazine.

Of course I didn't buy it—not for a minute. We all had a good laugh and I went back to my desk for a well deserved post-vacation nap.

A couple of weeks later a handwritten letter arrived, supposedly from a Sonja Travers of San Francisco, responding to "my" personal ad in *Micro C*. Sonja described herself and went on to say:

"It seems we are both lovers of the wilderness. I like to go hiking, but I can't go during ragweed season as I suffer from asthma and hayfever. I had a polyp removed from my sinuses last year which seems to help somewhat."

Sure. Once I had a polyp removed, too—from my brain. So I don't believe every little piece of BS my fellow *Micro C* inmates try to foist on me.

The next attacks on my incredulity came by phone, from people I'd always thought of as friends. Bruce (of Eckel fame) called to complain about my use of *Micro C* as a platform to further my own questionable personal needs. Thanks, Bruce. Next time you visit, you're sleeping outside with the cats and curs.

And Herr Entsminger.... Et tu? I learned long ago that Gary at his most sincere is also most suspect. But I didn't think he'd lower himself to aiding in this kind of nonsense.

No way I bought into this drivel, but dang if they weren't doing a fine acting job. I felt like I was stuck in a bad TV show—a cross between *The Dating Game* and *The Twilight Zone*.

Miss Esther Ruth Lancaster (a Mennonite from Pennsylvania) was up next.

"I am a hard worker. Every day I rise at 4:00 a.m. and milk the cows. Then I collect the eggs from the chicken house, feed the chickens and the ducks, and our little kitten. Then I start the day's bread baking, and while the bread

is in the oven I get my little sisters ready for breakfast and for school.

"Then I help mother with laundry which we do by hand and can take up the rest of the morning. Then we start preparing pies and roasts for dinner. After that we usually beat the rugs and sweep the floors and then go help Father and my brothers in the fields.

"I finished ninth grade. I am now 19 and I understand that in other communities, girls have more schooling. Perhaps you have met some of these girls, Mr. Fogg, and they must be very smart. But I can read and write and cipher quite well.

"I would like to have lots of babies and I think I would be a good mother and wife. I already know how to run a household and my cousin Sarah Bernstein says that I would get used to sex. Excuse me for speaking so openly, Mr. Fogg, but I am a farm girl after all and I'm sure it would be okay from time to time."

Now we were getting somewhere; I just love a good roast. And my rugs hadn't been beaten for at least nine years. I felt my raging disbelief easing into a fine, healthy skepticism.

I almost looked forward to the next letter. When it came from Ferrin Kennedy with a phone number, an address, a picture of a beautiful blond, and an invitation to lunch (her treat—my kinda woman), I knew it was time to see if Ms. Kennedy actually existed; I would accept her invitation.

(The only thing that worried me about Ferrin's letter was a closing remark that she found "Around The Bend" quite titillating. Did I really want

to be seen in public with a woman who would say such a thing?)

But the final letter caught me before I had a chance to take the plunge with Ferrin. I pulled up a chair in the production room.

"I saw your picture in *Micro Cornucopia* and it made me pucker! You seem like such a man!!"

Obviously an intelligent and discriminating woman, or so I thought.

"I'm petite, with blond hair and green eyes and I love to..."

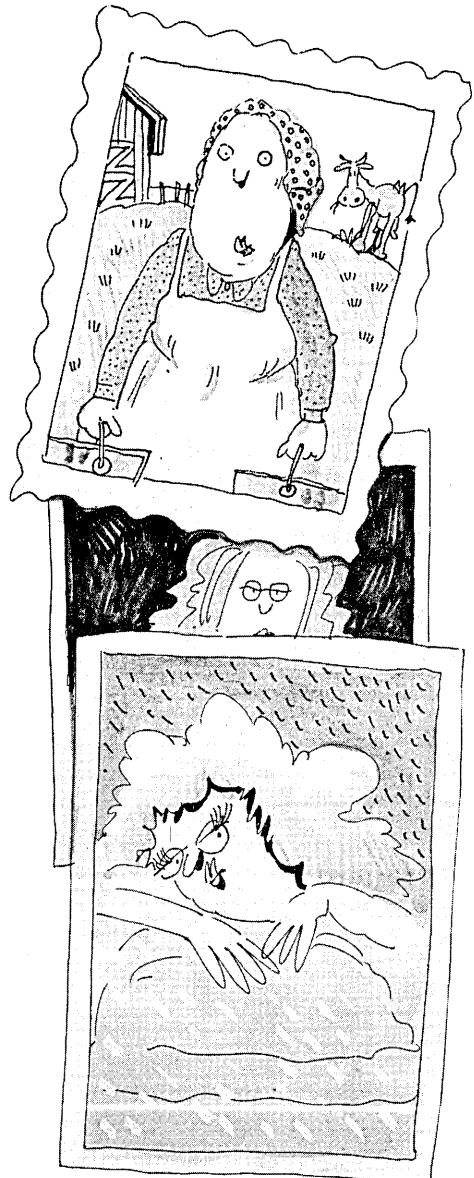
I won't continue; you can probably guess where the letter was heading. Besides, at that point a decidedly un-female picture fell out of the envelope and I heard a couple of snickers from the next room. I skipped to the end of the letter and read:

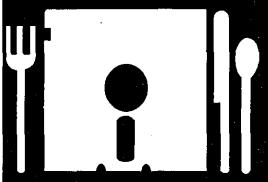
"I must be getting off now. I'm due at the humane society for my volunteer work. I just *love* those cocker spaniels.

"ps: Jackie and Cary helped me write this and all the others. Steamy enough yet???"

Suddenly I was surrounded by the same conniving crew that had watched me read the personal ad weeks before. They tried to appease me with pizza and chocolate frozen yogurt pie, and I let them think that they had. But now I get my final and poetic revenge: they wrote most of this article, but I get paid for it.

Editor's note: Of course the next pizza and the next yogurt pie are on you fella. And we'll take 'em as soon as you return from Yosemite.





ON YOUR OWN

By Kim Jindra with
Don Jindra

Information Modes
P.O. Drawer F
Denton, TX 76202
(817) 387-3339

Marketing The \$25 Network

Making It A Little Bit At A Time

The Jindras became special friends at the B.C. SOG, and the friendship has grown as I've seen them at SOG after SOG. Their \$25 network was also quite a hit, and despite the low price, they seemed to be doing quite well. So I asked Kim to send me an "On Their Own."

Our company, Information Modes, is a software development/publishing company with a grand total of two employees—my husband Don and me.

Until January, 1987, Don worked as a consultant. For approximately two years, our income came primarily from one small manufacturing company. Don developed software for their business with the understanding that he and the owners would market it in a separate partnership. As the project neared completion, it became clear that "marketers" and programmers are natural enemies, like cannibals and missionaries. We decided we didn't need their money.

IMODES

Suddenly we had virtually no income beyond the six month cushion in our savings account. Fortunately, I was working part-time at a local bookstore, making enough to cover food.

Don got busy. He had been thinking for some time of creating a simple and affordable network for the guy with two or three PCs. Now he had the time.

Before I knew it, he was disassembling and commenting DOS. "But what about the network?" I asked. He assured me DOS would gobble the network if he didn't understand it thoroughly. Also, I think he was just curious about how it worked.

The Ad

Soon Don was writing a classified ad for *Computer Shopper* to advertise his network and his DOS disassembly. He was sure other people might be interested in DOS. That \$34 ad

seemed like a big risk at the time. Plus, we would have to wait more than six weeks to see if there was any response. We decided to call our main product "The \$25 Network." If that didn't grab attention, nothing would.

I sent in the check as Don started work on phase two, The Weak Link, a subset of the network. He said he wanted to begin with a simple "network-like" project. He still hadn't started the network and we had an ad coming out in weeks. He hadn't entirely finished DOS, either! I told him six weeks was not a very long time.

"Don't worry," he said.

I worried.

Fortunately, he does his best work under pressure. Soon the big day came and we went down to the post office to check our box. Lo and behold, an order for the DOS disassembly. The *Computer Shopper* ad was out.

I asked Don if he had finished DOS yet.

"Don't worry," he said.

I asked how the network was coming.

"I think I'd better finish DOS first," he said, "but don't worry."

The next day another check arrived, this time for the network. I asked how long until the network would be ready to ship.

"I guess I'd better get busy," he said, "but don't worry."

Don put in 18-20 hour days. I helped by having nightmares—something about the Attorney General showing up at the door.

The Paperwork

By the time Don finished the network, he was so sick of the project he had a tough time with the documentation. He enlisted my help, but since most of the stuff was new to me, I just proofed the copy.

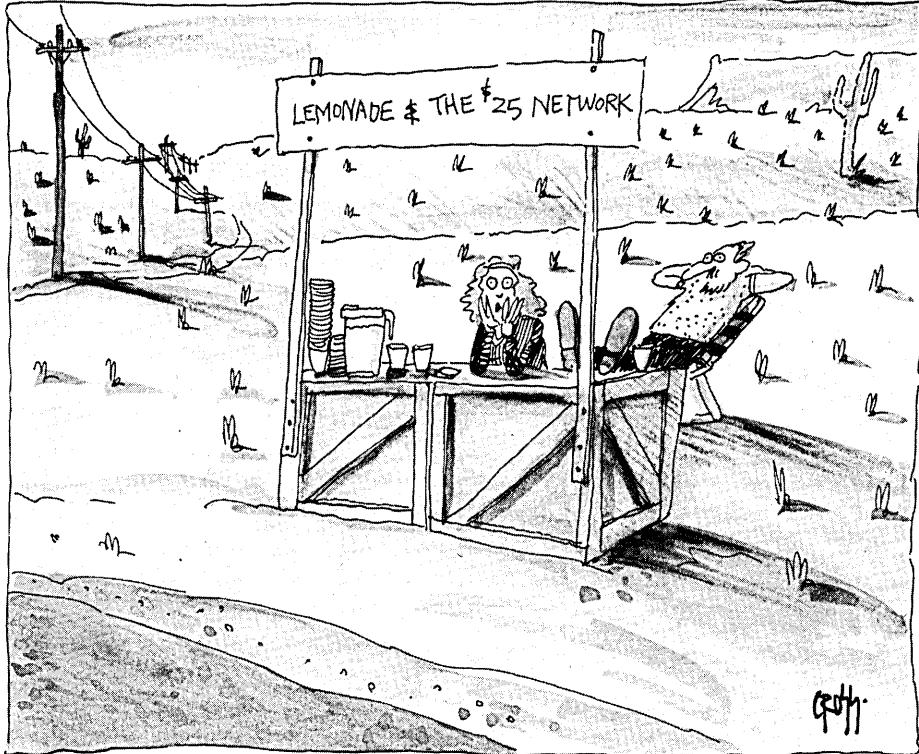
We printed the originals on our dot matrix printer, reduced the copy, pasted it up, and hit the local copy center. Then, stacks of pages in hand, we adjourned to the kitchen table to assemble manuals.

Five a.m. August 5, 1987, we finished. We'd received a handful of orders so we duplicated

disks, packaged everything in hand-cut cardboard squares, and shipped.

We promptly packed up the car and headed for south Texas and the Gulf. Once home, we got our first bug call. The program couldn't read MS-DOS 3.1 disks. We'd tested PC-DOS 2.1 and MS-DOS 3.2, assuming that if they worked, the rest would. Not true! We quickly fixed the bug and shipped updates.

We thought we had a spiffy product, but we didn't have the bucks to really advertise. We quickly rejected shareware for the network, but we did have The Weak Link.



The Weak Link

The hook on The Weak Link was source code. When people registered, they received the complete source. We also mentioned The \$25 Network on the shareware disk. (Registration is only \$15, but still only two or three register in a month.)

Our decision to test the shareware waters did not come lightly because we were quite skeptical. After several weeks of agonizing, we sent a copy of The Weak Link to PC SIG.

While we were waiting (it seemed like forever), we continued running the classified in *Computer Shopper*. We celebrated when sales hit \$100 a month. As more money came in, we slowly increased the size of our ad. We set modest goals and celebrated the gains.

A Foreign Agent

Then one day we received a letter from West Germany. Joseph Kirschbaum, a PC-SIG distributor, wanted exclusive rights to sell The \$25 Network in Europe. He had read about our network on The Weak Link disk. (That's how we discovered the PC SIG Library was distributing The Weak Link.) We laughed when we read the letter, we hadn't even considered overseas sales.

He called us before we had time to respond. He was serious. We tried our best to discourage him, but he was so persistent we finally relented and

Our decision to test the shareware waters did not come lightly because we were quite skeptical.

agreed to terms. We didn't expect too many similar offers. At the time, we looked upon European sales as bonus money, not really expecting much.

the HARD DISK FILE & DIRECTORY MANAGER

TREE TOP

TreeTop provides an easy to use interface to DOS commands. A visual tree of your directory structure is displayed with a point and shoot ability to move around your hard disk. Whether you are doing hard disk housekeeping, reorganizing your directory structure, or simply copying files, **TreeTop** makes it a breeze.

Check out these features:

- ◆ Fast and user friendly
- ◆ Pull down menus
- ◆ Context sensitive help
- ◆ User installable setup
- ◆ Full color or monochrome support
- ◆ Shell to DOS
- ◆ Execute COM, EXE, and BAT files
- ◆ Point and shoot user interface
- ◆ Execute your favorite editor
- ◆ Mouse support (not required)
- ◆ Sort files by name, ext, size, date/time, ascending or descending order
- ◆ Copy, move, delete, rename, or print single files or groups of files
- ◆ When copying, TreeTop will optionally:
 - Scan and display destination drive
 - Copy to root
 - Copy to user defined pathname
 - Prompt user when disk is full
- ◆ Add, delete, and rename directories
- ◆ Select and operate on files in a single directory or by the entire drive
- ◆ Select files by wildcards, date/time, attributes, or individually
- ◆ Change files' date/time or attributes
- ◆ Find files with speed searching
- ◆ Disk and directory status display
- ◆ View files in hex or text format
- ◆ Set, rename or delete disk volume labels
- ◆ And much, much more!

Only \$39.00

30 Day Money Back Guarantee

Price includes disks (3 1/2" and 5 1/4"), registration, bound manual, tax and shipping in U.S. Send check or money order to:

Kilgore Software
P.O. Box 2291
West Sacramento, CA 95691
(916)371-3715

or call toll free:
1-800-TREETOP

Runs on DOS 2.0 or greater, 256K RAM, with or without a mouse. Quantity discounts and site licenses available. TreeTop is shareware, so you can distribute copies for evaluation.

Reader Service Number 175

A week later we got another offer from Germany. But it turned out that signing with Kirschbaum was an excellent decision. We've had a good (and profitable) relationship for two years.

More Promotion

Meanwhile, business picked up in the U.S. We increased our product line and threw in a list of other products with the orders. We also sent this information with bingo card literature.

Other products include: disassemblies of DOS 1.1 (\$15) and 2.1 (\$45), The Weak Link (\$15), Hercules Graphics Drivers (\$15), an 8748 simulator (\$15), and the System Technical Reference complete with BIOS calls, DOS calls, and everything else Don needed to have on hand while writing drivers in assembly language (\$5). The Technical Reference has been very popular, even though we don't mention it in our ads.

Meanwhile, the want ad in *Computer Shopper* grew larger as we added information on the Hercules Graphics Drivers and the simulator.

As sales grew, Don put me to work nights (after spending days mucking about in my new archaeology job). I started by helping out with the mail.

Our next big step was to take out a small showcase ad just for the network. The ad cost \$250, very foolhardy. It meant we would spend close to \$400 per month advertising in *Computer Shopper*. Plus, they wanted a three-month commitment.

All The Way

By October 1988, I'd quit the digs to go full time on our business. It was a risk, but Don couldn't handle it alone. I worked on packaging and promotion while he improved the Network and worked on new products.

We began promoting our products at local shows. Don and I and our two boys stayed up half the night copying disks and collating manuals, getting ready for the first show in Fort Worth. The turnout was terrible and we sold only five copies of the Network. We had better luck in Dallas a few weeks later, but the experience taught us a lesson: THINK BIG, but don't be surprised by small.

Sharing The Network

Around this time we received some bad news. Someone was uploading our network to bulletin boards. This was ex-

tremely disappointing. We bought a modem and started checking into BBSs around the country. When we found our network, we asked the SYSOPs to remove it. Most did and apologized.

Some didn't help us at all. Others seemed downright criminal. So Don decided to serialize each copy. He also made significant improvements. This was early 1989. Since then we have updated the network every couple of months.

Big Time

Finally we decided it was time to advertise in other magazines. I contacted *Micro C* and signed up for a year's worth of micro ads. At \$79 per issue it seemed like a bargain, and that's how it turned out. (In fact, it has been one of the most profitable ads we've bought. It's obvious from the people who contact us that *Micro C* has extremely loyal and knowledgeable readers.)

Shortly thereafter I called *PC Tech Journal*. Despite their heavy push to sign us up for a long term contract, I committed to just one time. I had to find out if it was worth the money before committing to a long run.

I soon discovered I'd made the right decision. We'd caught the last issue of the magazine before Ziff-Davis folded it. Other small advertisers were not so lucky. They saw their ads continued in publications not of their choosing. This was only the first of several bad experiences we've had with Ziff-Davis. (And Ziff just recently bought *Computer Shopper*.)

Plastic

Obtaining a vendor account for VISA/MasterCard was another challenge. We had sent a significant number of orders "net 10," hoping the customer would send us a check when he received the product. Some customers were very impressed with our trust, and most eventually paid; but the few who didn't were a constant source of aggravation. We needed VISA/MasterCard.

Entering a bank and mumbling "mail order" is like putting a kerchief over your face and shouting, "Stick 'em up." It took months to convince just one bank we were legitimate, but it paid off. Our volume doubled.

It was time to increase our advertising again. Our goal had always been

BYTE, so we gave them a call. The ad was expensive, but if we'd learned one thing, it was that advertising sometimes pays. We also added *PC Resource*.

How Ads Work

All we expect from the first ad in any publication is that we get as much in orders as we paid for the ad. So, if the ad cost \$400, we expect to get \$400 in orders. After a few months, we expect an ad to bring in at least three or four times the money we spend. (Our Micro C Micro Ad does the best, generating a minimum of 10 times its cost.)

A long time ago we decided it would be better for us to take out a lot of small ads than gamble on bigger ads in fewer places. We think consistency and stability are more important than single full-page splashes. (Some customers will call only after seeing our ad in several places.)

Ads in *BYTE*'s Buyers Mart are \$475, but we got immediate response to our first ad. We received 34 bingo cards the first week; now we're up to 75 plus per week. About 25% of the bingo checkers purchase after receiving our literature. Plus, at least $\frac{1}{3}$ of our phone orders come from *BYTE*.

PC Resource wasn't as successful at first. We had just about decided to spend the \$306 per issue elsewhere, but response seems to be steadily improving so we'll probably stay.

Computer Shopper has been a consistent performer though we've expected response to go downhill as the magazine gets larger. So far that hasn't happened. We have been in their Shoppers Mart for over a year. It generates slightly less response than *BYTE*.

Besides trying different magazines, we've also tried different slogans. The one we've settled on is: "Skeptical? We make believers." Does it work? Every day we get several calls that begin, "Make me a believer." And we get letters signed, "Skeptical."

On The Road

Early in the spring of 1989, I read with great interest about the B.C. SOG in *Micro C*. I was convinced we should try to go on the road with The \$25 Network. Besides, we had planned a West Coast vacation. Don agreed.

We had no idea what a SOG was, but thought there was a chance we would meet the *editor* and perhaps convince him to look at our product. We

hooked up with an answering service and a mail service and drove to Canada.

The B.C. SOG was wonderful. We had so much fun that we followed up with the Rocky Mountain SOG. Sales-wise, our trip to Colorado was more successful than B.C. I decided in Colorado we would go to SOG East, but it took me about a month to convince Don. I just couldn't let Dave be the only one to complete the Tour de SOG '89.

Editor's note: Tour de What? I have to admit that Kim and Don really know how to network, and it was great seeing their friendly faces in Port Alberni, B.C.—Gunnison, Colorado—York, Pennsylvania—and Dallas, something. (Dallas used to have a football team. I think.)

As I mentioned, we'd originally attended SOG in hopes of meeting the editor of *Micro C*. Well, at the Longhorn SOG here in Texas, Dave became the fifth member of our family. We have a small two-bedroom house, so Dave bunked with the kids. There's nothing like rubbing elbows with the rich and famous!

Editor's note: I was sure I'd debunked the rich and famous part, but I must thank Kim, Don, Courtland, and Donald for putting on the Longhorn SOG and for inviting me to be a part of their wonderful (tight-knit) family.

Which brings us to today. Business is better than ever. The two of us barely keep up, and it's time to think about moving to a larger house and hiring help.

Why \$25?

Our success surprises many people. After all, they constantly tell us we can't make money selling a program for \$25. We hear that all the time. In fact, we get tired of hearing it.

We decided on \$25 for a few good reasons. At the time, \$25 seemed like a lot of money. (That was all we could have afforded if we were buying a network.) We had no money for advertising, so the name of the product had to grab attention. Finally, our goal was to sell to the majority who had two or three computers—not to the relatively few who were putting together large networks. Probably most important, though, people don't need a lot of convincing to spend \$25.

Also, we wanted our customers to feel like they got the buy of the year. We didn't just want their money—we wanted their word-of-mouth advertis-

ing. Plus, if Microsoft, with all its overhead, can sell Quickbasic for \$59, we should be able to prosper at \$25.

It's not our customers who complain about the price, it's the dealers. They want to make \$100 for a 15-minute sale. Fine. But until someone explains to us exactly why we aren't making money at \$25, we'll go happily about our business and bank our profits.

We started small, borrowed no money, and took no great risks. That's probably why we're doing well today. We've made a few mistakes. Don should have quit consulting six months earlier. We should have serialized each copy of the network from the beginning. We probably should have increased advertising faster. Maybe we should have charged more for the network. But this is all hindsight. Contrary to popular belief, hindsight is not 20/20.

The Future

Soon we'll be moving to larger quarters (room for help) and adding additional phone lines (including an 800

number, a BBS, and a FAX). Don's also finishing up a whole new network.

Our new net should handle 20 to 30 nodes, be very flexible, and support lots of different hardware such as serial ports, parallel ports, arcnet cards, and SCSI. It will also include all the specs for the hardware interface. That way purchasers can write their own hardware drivers. Target price will be \$75, and we'll offer a \$25 discount to current customers.

On the marketing side, we'll keep expanding the advertising slowly. What would we do if we got 10,000 orders in a month? We'll be adding some regional publications as well as some language magazines. Finally, we'd like to try *PC Magazine*, if we can justify the expense.



NOW — A COMPLETE PASCAL — FOR ONLY

\$29.95!

Goodbye BASIC, C, COBOL—hello PASCAL! Now, to make this most advanced language available to more micro users, we've cut our price—to an amazing **\$29.95!** This astonishing price includes the complete JRT Pascal system on diskette and the comprehensive new user manual. Not a subset, it's a complete Pascal. Check the features.

Separate compilation of external procedures • Auto-loading • 14 digit FLOATING POINT arithmetic • True dynamic storage • Verbal error messages • Fast one-step compiler: no link needed • Graphing procedures • Statistics procedures • Activity analyzer prints program use histogram • Operating system interface

THIS IS THE SAME SYSTEM WE SOLD FOR \$295!

So how can we make this offer?—why the unbelievable deal? Very simply, we think all software is overpriced. We want to build volume with the booming IBM market, and our overhead is low, so we're passing the savings on to you.

AND AT NO RISK!

When you receive JRT Pascal, look it over, check it out. If you're not completely satisfied, return the system within 30 days and your money will be refunded in full! THAT'S

RIGHT—COMPLETE SATISFACTION GUARANTEED OR YOUR MONEY BACK!

In addition, if you want to copy the diskette or looseleaf manual—so long as it's not for resale—it's o.k. with us. *Pass it on to your friends!* This is a Limited-Time-Offer: SO ACT TODAY—DON'T DELAY ENJOYING PASCAL'S ADVANTAGES—AT \$29.95, THERE'S NO REASON TO WAIT!

To: **JRT SYSTEMS**
P.O. Box 187
Enola, PA 17025
phone 717/732-1093



O.K. You've sold me. Send me JRT Pascal by return mail. I understand that if I'm not completely satisfied, I can return it within 30 days for a full refund.

I need 5 1/4" disk or 3 1/2" disk. Send me the JRT Pascal program formatter too, for only \$14.95.

Name _____

Address _____

City _____ State _____ Zip _____

Check/M.O. COD Company P.O. [add \$20]

PA residents add sales tax. Add \$10 for shipping outside US/Canada. US funds on a US bank only. Needs only 192K and 1 floppy drive.

Letters *continued from page 6*

reformat) and it's like I have a new drive. I've not had a single read/write error in over six months and PCTools can't find any sectors to balk about. Sure, it took about 12 hours for SpinRite to do its stuff, but it beats having to reformat my hard disk every month and spend an hour or two feeding it disks.

I've since used SpinRite on the Miniscribe cards in my other computers and have not had *any* problems with the drives afterwards. In your editorial you imply that the problems that have been reported seem to be limited to Seagate drives. Maybe you should pan the Seagates (as you usually do—that's why I bought Miniscribe) instead of SpinRite!

Well, you wanted feedback. For me, SpinRite has been a lifesaver and I think that you've done both your readers and Gibson research a disservice by steering those of us with XT-type hard drives away from such an outstanding product. I feel bad writing such a critical letter; *Micro C* is the only magazine (computer or otherwise) that I read from cover to cover as soon as it arrives, and I recommend it most highly. But I think that you really goofed on this one....

Joel M. Goldberg
RD#1 Box 230
Richmond, VT 05477

Editor's note: If you felt bad enough, you could bring over your lawn mower. Apparently the SpinRite folks thought I'd trashed their product, too, and I feel bad about that. (I'll help you with the grass.)

We've been chasing a problem with our BBS for the past few months; it's been going off into never never land fairly regularly. Larry noticed some sector errors on the little Microscience drive so he ran SpinRite on it. The system's been flawless since (and it's an XT).

Plus, of course, 286 and 386 hard drive controllers don't have the homing problem (the problem really resides in the XT's controller). I don't know; use SpinRite if you wish, we're using it. However, you might want to read the following letter before letting it change the interleave.

SpinRite Questioned

Noticed that you have taken a liking to SpinRite. I have a little nagging doubt about using that program,



though. Perhaps you can answer this for me.

Assume I have a hard drive reformatted at, say, 4:1 interleave. This hard drive has had some use over the years, and it has a few bad sectors that were not on the original bad track table. No real problem, though, because DOS's FORMAT command noticed it couldn't read those sectors and marked them (in the FATs) as unusable.

Now I get a copy of SpinRite, and I set to it, "Figger me out the best interleave and rebuild my drive." Off it goes on its merry way, and lo and behold it finds out the best interleave for my system is really 5:1, not 4:1. It merrily reformats my drive by reading in a track at a time, then reformatting that track to 5:1, then writing back that same track of data.

Here's the fun part. It gets to the bad spot. Reading in the track, it notices it cannot read sector 7 of this track correctly. It reformats the track and tries to write the data back again. The bad spot is no longer in sector 7! We just moved things around, and now the bad spot is in sector 12!

If it writes this track back, it will be guilty of the major crime of putting good data on a nonreadable bad sector, as well as the minor one of leaving a now good sector marked bad.

The safest way to reformat your hard drive out from under existing data would require skipping any track that had a bad cluster marked on it, determining the interleave for each track and if that track sprouts an error during for-

matting, reformatting to the old interleave before writing the data back. Needless to say, this is time consuming and tedious and very, very slow going.

Till I can get an answer to this problem, I'll just stick to the old tried and true approach of a complete backup (preferring my trusty Archive 60 Meg tape with Syngen controller), then reformat the drive and restore the data. This not only gives me the proper interleave, but also has a side effect of making all my files contiguous for faster access.

Other notes. Does SpinRite allow you to set the skew factor? This is a fixed offset in the interleave for each succeeding head, in case your controller or drive is slow in switching from head 2 to head 3. Skew factor can amount to about 5% of the throughput. Normal values of 0 or 1 usually are fine. Note, 0 gives you no skew.

Another thing—can you reformat only part of the drive? I wrote a reformatter in C for IBM AT clones that does both of these. DOS can handle 2:1 MFM using a WA-2 controller, but Microport UNIX can only keep up with 3:1. I did not want to slow DOS down, and didn't want UNIX to run slow either, so I reformatted the back two-thirds of the disk to 3:1. Pretty slick, but not a project for the faint of heart.

I do appreciate the magazine; it's the last known true hacker (in the old style of the word) magazine around. *DDJ* has gone commercial, *BYTE* has become a small *Computer Shopper*, and everybody else is just dying to know Bill Gates' latest breath. *Micro C* just keeps on cranking out interesting hardware and software for true hobbyists.

John Welch
1310 Kenneth Circle
Elgin, IL 60120

Editor's note: Any answers for John? Anyone? As for Bill Gates, he's welcome to stop by the Micro C office anytime, especially if he has a lawn mower.

Beware The Big Time

I take keyboard in hand to write my first-ever letter to an editor. The motivation comes from reading "Around The Bend" in Issue #50, which makes me wonder if maybe you haven't finally gone (around the bend, that is).

On one hand you make a good point about a marketeer's value to an enterprise and the need to find one's niche; on the other, you say you're ready for some heavy duty "focusing" and "significant playing" in the publishing biz. While "focusing" and "playing," you are going to keep *Micro C* as appealing as ever. Good luck.

One of the things I find most enjoyable about your magazine is precisely its lack of focus. It is one of the few publications I read cover to cover because every article is different, and every article is interesting.

While I'm not knowledgeable in the yins and yangs of magazine publishing, I can't help but think that being a bi-monthly and operating away (a long way away) from the "media centers" contributes mightily to this. Good gravy, ask for a partner of the type you described and you're liable to get Ziff-Davis. I can see it now: "Tidbits" by John Dvorak.

You started from scratch, you've attained a loyal following, and maybe now it's time to cash in and head for the mountain (I thought you were already there). I don't have any arguments with that; but I'd rather see you take your gains and pack it in than let *Micro C* become another *PC Magazine*.

While I'm spilling my guts, let me put in a plug for Bob Morein and more articles on UNIX for 386-based PCs (as opposed to dedicated work stations). I'd like to know more about the hardware requirements to run UNIX efficiently and reliably on a personal computer.

Also: will UNIX ever have a standard? If so, which of the current crop is the best bet? Will UNIX ever get a Mac-like interface? Just how portable is UNIX across different CPUs?

I don't remember when I first subscribed to your magazine; however, I do recall that I was still using a Kaypro 2. Since I recently re-upped for 18 more issues, I feel I have the right, indeed an obligation, to make my views known on the proposed shift to the "big time."

P.S. I think you've already found your niche.

C. R. Bartchy
4340 South Hopkins Ave.
Suite 230
Titusville, FL 32780

Editor's note: Thanks C.R. However, I think it's time for Micro C to reach out, and I think it can be done without losing Micro C's style.

SMUG Lives!

I read with interest your discussion of "The End of User Groups" in your editorial of Issue #47. I would like to make the following points.

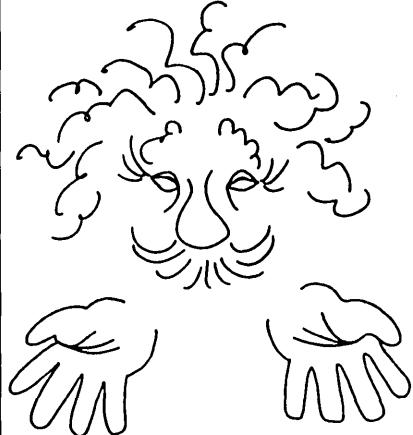
(1) The Sacramento Microcomputer Users' Group (SMUG) still exists and is healthy in spite of reduced membership.

(2) SMUG meetings have presentations and discussions involving presently popular computers, including IBM PCs (and clones), Apple II GSs, Atari STs, and Amigas. Your article stated that SMUG was going out of business because "it stayed with CP/M, S100, and other (more and more) unique systems." That is not accurate. SMUG supports a wide variety of popular microcomputers.

(3) SMUG's members consist of people whose interests lie in the more technical aspects of computer hardware and software. They tend to have an interest in "what goes on under the hood" and why. Frankly, SMUG members are the audience that *Micro Cornucopia* targets.

I enjoy reading your publication. Glad to see that you are reading ours!

Don Del Porto
SMUG President
P.O. Box 161513
Sacramento, CA 95816



*Organize, Query,
& Make Connections
Between Files of Information*

MICRO EINSTEIN

The Expert System Shell

- Create expert systems easily in minutes
- With pulldown menus and windows
- Automatic rule generator
- Context-sensitive help
- Free example expert systems
- Interactive full-screen text editor
- DOS access from shell
- Turbo fast execution **(NOW 5 times faster!)**

For Diagnosing...
Monitoring...
Indexing...
Organizing...
Classifying...
& Discovering links
between files of information.

ONLY \$99.95!



ACQUIRED INTELLIGENCE
P.O. BOX 2091 • DAVIS, CA 95617 • (916) 753-0360

Reader Service Number 72

Continuing AROUND THE BEND

"It must have been your significant other, but as long as I've got you on the phone...."

"Look, I didn't request any information. My significant other didn't request any information, and I'm certainly not interested in connecting up with a dishonest futures trader." (After watching the papers, I'd suspect that adding "dishonest" was redundant.)

It turns out I'm not interested in futures, at least not the kind that cause grown men to stand in pits shouting and waving their hands. (All the while hoping the guy on their elbow doesn't work for the government.)

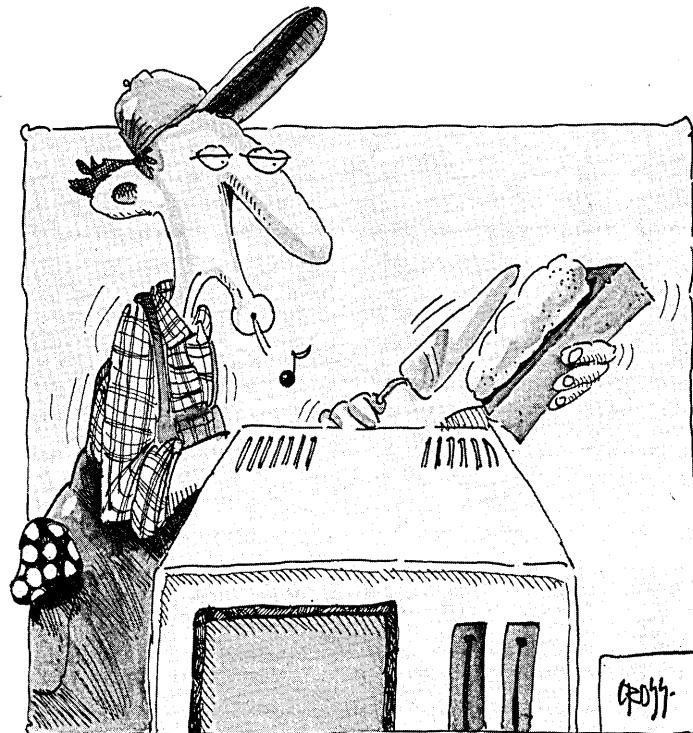
This is just one of a plethora of sales calls I've received in the past month. Stock and bond brokers every other day, all trying their best to sneak past Nancy.

"Dave, someone on the phone says he's calling about your new business card. I don't think it's legit."

It wasn't. He was a stock broker.

Folks selling penny stocks (stocks in the 10¢ to \$5 range, not traded in a major exchange) have been particularly active. I bit once, lost most of \$1,500 on Envirosure, a waste processing company that apparently was dumping, not processing.

I purchased the stock because the Blinder Robinson salesman told me the stock was moving up rapidly. Curiously, its direction changed a few days after I bought it.



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 CC RIDER 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.

Then I read in *The Wall Street Journal* that penny stock-brokers (including Blinder) often purchase large blocks of shares and then push the shares at a big profit. Once the push is over, the price nearly always falls into a hole. In this case, not only did the push end, but the government started criminal action.

After all that, the Blinder salesman asked if I'd like to move into something a little more solid—part of a race horse. (I didn't ask which part, but I figured he was trying to saddle me with yet another slow mover.)

This experience made me really appreciate *The Wall Street Journal*. The *Journal* not only gives me the latest on financial scams (I read about diamond, platinum, and oil drilling rights scams just days before the calls started), but it also covers computer companies. In fact, it often covers things untouched by the computer journals.

For instance, there's Miniscribe.

Miniscribe has a reputation for solid drives, but I didn't realize just how solid until I read the *Journal's* piece on the company. Miniscribe was apparently shipping bricks (solid new bricks) instead of drives to make it appear that sales were growing. I can imagine some dumb dealers, serious anti-technical types, but it's hard to believe they were fooled.

Plus, I'm sure dealer training firms were quick to add the appropriate classes:

"This is a hard drive—this is a brick."

"This is a hard drive—this is a brick."

"You don't slop mortar on a hard drive."

"You don't install a brick in a Mac."

Usually.

Western Wares

Standard Professional

\$89

\$279

For DOS and OS/2

(303) 327-4898

FREE DEMO DISK

Box C, Norwood, CO 81423

Reader Service Number 169

The \$25 Network

We were well into SOG East (great SOG, by the way). Don Jindra had cabled up three computers and started his presentation on networks (mostly his network). It was early afternoon so the previous night's Jolt SIG had nearly worn off.

Like his presentation at the BC SOG, the group response began with the usual, "Bet it can't..." passed through, "Wow, really?" and finally settled into an easy banter.

"What if you were on system one and the guy on system two started harassing you? Could you do something nasty?"

That brought the few still-fuzzy heads off the table.

"Could you lock up his keyboard?"

"How about really slowing down his system with something disk intensive?"

"Format his hard drive?"

"Yeah, how about formatting his hard drive?"

"Nope, you couldn't run FORMAT on system one and have it format a drive on system two," Don broke in. "DOS won't let you do that. You'd have to be running FORMAT on system two. However, my next version will not only let you communicate with the operators of the other machines, it'll also let you take control of those machines and run programs on them just as though you were sitting in front of them."

"The ultimate multitasking?"

"Right."

"That includes FORMAT?"

"Yep."

"If operator two were distracted for a second?"

"Ummmm."

"We're all weird. You know that."

With networks the latest rage, I'm constantly asked by local people what kind of system we use.

I used to tell them, "We don't have a network."

Now I tell them, "Oh sure, we run the \$25 network."

I like it, too. I can network any two systems in the office just by connecting them together via serial cable and booting them both off the floppies I set up with the network and DOS. At 115K baud it's not as fast as ARCNET, but it's a lot cheaper.

I just built an AT to replace the XT we've used for receipts and labels. Counting everything, it was about 10 megs of program and data. In five minutes, I had the two systems cabled together, had booted them up on the network disks, and voilà: drive C on the XT became drive F on the AT. I simply used COPY *.* to move all the files from F to C and that was it.

I'll leave the two connected and once a week back up the AT's files to the XT. Not bad.

Cards In Magazines

I was speaking to the PC SIG in Dallas when the question came up. Why do magazines contain so many cards, the ones that fall out everywhere.

Actually, we don't have loose cards in *Micro C*, but there's a

If you thought Borland's popup product
was great just wait until you see
OpalFire's MyFLIN for Pascal.

MyFLIN is a TSR program that captures procedure and function details directly from the screen while you are programming. It saves this information in an indexed database for instant recall at any time you need it.

MyFLIN will save you hours of searching thru pieces of paper looking for parameter details when calling a procedure. Simply type part or all of the name and press the hotkey. MyFLIN will popup over your source code and display the nearest named description you have stored in your database, complete with parameters and comments.

To save a new description, position the cursor on the procedure name, popup MyFLIN and press "A" to add. The details will be captured from the screen and saved with a single keystroke and you can add a comment line as well.

Price \$69.00 + \$5.00 P&Post.

Visa / Mastercard / American Express are accepted.

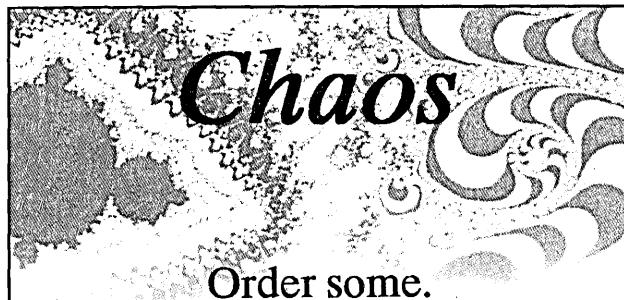
OpalFire Software Inc.

329 North State Street, OREM, UTAH 84057

Phone 1-800-336-6644

MyFLIN requires PC/XT/AT Computer and MS-/PC - DOS 2/3.xx.

Reader Service Number 161



Mandelbrot Explorer 3.2

Fantastic fractal graphics on 16-color EGA/VGA to 800x600. Magnifies up to 16.5 trillion times. Stop and start at will, save and retrieve, collage, full control over color boundaries, "zoom box," display of periodic orbits, auto-backup, all optimizations for speed including pixel interpolation and 386 integer support. Comes with seven ready-made pictures for immediate gratification.

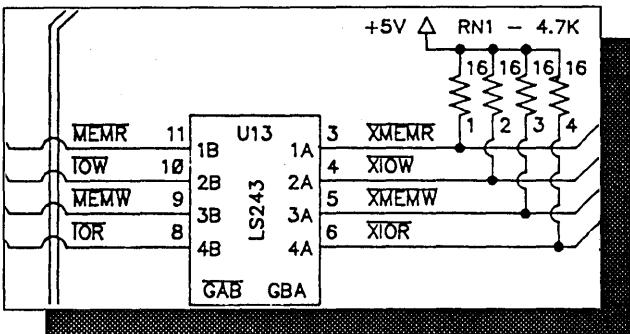
\$34.95

Peter Garrison
1613 Altivo Way
Los Angeles, CA 90026
213 665 1397

When ordering please specify EGA/VGA and disk format
Overseas orders please add \$4

Reader Service Number 112

MICRO CORNUCOPIA XT SCHEMATIC



At last you can plumb the mysteries of your computer with this single sheet schematic of the IBM XT's main board. A wealth of information for both True Blue and clone owners.

Although clones use slightly altered board layouts and different chip location names, they're close enough to the original for this schematic to be very useful. As an example — you have a dead clone. Lil sucker won't even beep. A look at the schematic shows the location of parallel port A. You know that the power-on self test loads a checkpoint number into port A before each test. So now all you have to do is read port A with a logic probe to see how far the system went before it puked.

We include these checkpoints and other trouble shooting information with the schematic.

IBM PC-XT Schematic \$15.00

CP/M KAYPRO SCHEMATICS

Of course, we still provide a complete schematic of the processor board in your CP/M Kaypro. It's logically laid out on a single 24" by 36" sheet and comes complete with an illustrated theory of operation that's keyed to the schematic. You get detailed information available nowhere else.

Kaypro II & IV (pre-84) \$20.00

Kaypro 10 (without modem) \$20.00

Kaypro 2, 4, and 10 (84 series) \$20.00

NOTE: These packages cover only the main boards. You're on your own when it comes to disk drives, power supplies, video cards, etc.

Phone Orders: (503) 382-5060 or 1-800-888-8087
Monday-Friday, 9AM-5PM PST

Micro Cornucopia,
P.O. Box 223, Bend, Oregon 97709

very good (but little known) reason other magazines do. No, it's not an overabundance of trees, it's an underabundance of postal workers.

You see, for many years the postal service was losing mailmen. Yep, they'd find a good man, train him for a month on his new route, send him forth alone, and never see him again.



For a while they suspected pit bulls or UPS drivers, but it turned out their new charges were simply getting lost. So magazine publishers got together and came up with the idea of inserting loose cards. Within a month, every mail route had become a trail of cards. (Hansel and Gretel should have carried magazines into the woods rather than bread.)

The post office was happy, the publishers were happy, and now that you know why the cards are there, you can be happy, too. (Now if we could just keep those vicious little Boy Scouts from going out on litter patrols....)

286, 386, 386SX, 486

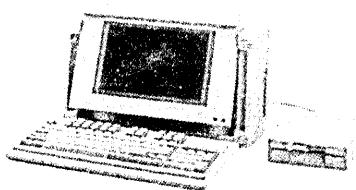
There's been a bit of controversy about chip choices. Over the past year, we've bought four 12 MHz 286 machines. After the 8 MHz 186s and the 6 MHz AT, they've been wonderful. With Intel getting \$300 a piece for 386s, there didn't seem much reason to purchase anything beyond the 286.

Now, however, there might be a few reasons.

- Intel's dropped the price of the 386 and the 386SX.
- We're beginning to hear about some interesting applications that take advantage of the 386. There's windows

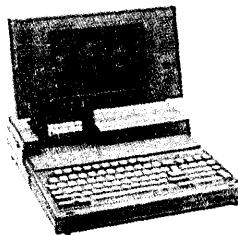
RABBIT

GAS PLASMA PORTABLE



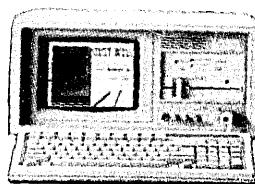
CHICONY

LAP-TOP



McTEK

EGA PORTABLE



*LCD PORTABLE ALSO AVAILABLE

- 80286-12 CPU
- 640K MEMORY, EXPANDABLE TO 4MB
- DUAL SPEED 6/12 MHZ, 0 WAIT STATE
- LANDMARK 16 MHZ
- 5 SLOTS, EXTERNAL 5 1/4" 25 PIN DRIVE PORT
- CGA/MGA/EGA 640x400, 4 GRAY SCALE
- 101-KEY ENHANCED KEYBOARD
- GAS PLASMA DISPLAY
- 1 PARALLEL, 1 SERIAL
- 200W AC 110/220 AUTOSWITCHABLE
- EGA/MGA MONITOR PORT 9 PIN
- HARD DISK/FLOPPY DISK CONTROLLER
- 1.44MB FLOPPY DRIVE AND 20MG HARD DISK 40MS
- 9.45"x16"x8.27" 19.8 LB.
- CARRYING BAG w/SHOULDER STRAP
- ONE YEAR LIMITED WARRANTY

GAS PLASMA \$2199
LCD PORTABLE \$1729

- 80286-16 CPU (0 WAIT STATE)
- NEAT, TURBO PAGE MODE SPEED UP TO 21.4 MHZ
- 1MB ON BOARD, EXPANDABLE TO 5MB (EMS V 4.0)
- CGA/MDA/EGA, 640x400, 4 GRAY LARGE GAS PLASMA DISPLAY
- 1.2MB FLOPPY AND 40MB HARD DISK
- "84 + FN" ENHANCED KEYBOARD
- 1P/2S (D-9 and D-25), REAL TIME CLOCK
- 100W, AC 110/220V SWITCHABLE
- 14.8" x 13.4" x 3.7", 15.4 LBS.
- CARRYING BAG w/SHOULDER STRAP
- ONE YEAR WARRANTY

..... \$3400

- 286-12 CPU, LANDMARK = 16MHZ
- 640K DRAM (EXPANDABLE TO 4MB)
- 0 WAIT STATE, DUAL SPEED 6/12 MHZ
- DUAL FLOPPY AND HARD DISK CONTROLLER (1=1 INTERLEAVE)
- ONE 1.2MB FLOPPY DRIVE
- ONE 20MB HARD DISK (40MB)
- TRUE OS/2, XENIX, MS DOS AND NOVELL COMPATIBLE
- 80287 MATH CO-PROCESSOR SOCKET
- ENHANCED 101 KEYBOARD w/TACTILE FEELING
- 2 SERIAL PORTS, 1 PARALLEL PORT AND 1 GAME
- LED INDICATORS
- 200W UL POWER SUPPLY
- ONE YEAR WARRANTY

..... \$2299

McTEK 286/12MHZ

Assembled & Tested IBM® AT Compatible
MS-DOS® OS/2® Compatible

- 80286 12/6 MHz
- Phoenix BIOS
- 640K of RAM Expandable to 4MB
- 0 Wait State
- 200W Power Supply
- 1.2MB Floppy Drive
- Ports: 1 Serial, 1 Parallel, 1 Game
- Dual Floppy/Dual H.D. Controller
- Monochrome Graphic Card
- 101 Key Enhanced Keyboard
- TTL Monitor 12"
- 20MB Hard Disk (40MS)

Options: Call

\$1,199⁰⁰

McTEK 386/165X

Assembled & Tested IBM® AT Compatible
MS-DOS® OS/2® & UNIX® Compatible

- 80386 16/8 MHz Norton V4.0 SI 17.6
- Phoenix BIOS
- 1MB expandable to 16MB RAM
- 80387 Coprocessor Socket
- 200W Power Supply
- 1.2MB Floppy Drive
- Ports: 1 Serial, 1 Parallel, 1 Game
- Dual Floppy/Dual H.D. Controller
- Monochrome Graphic Card
- 101 Key Enhanced Keyboard
- TTL Monitor 12"
- 20MB Hard Disk (40MS)

Options: Call

\$1,495⁰⁰

McTEK 386-20MHZ

Assembled & Tested IBM® AT Compatible
MS-DOS® OS/2® & UNIX® Compatible

- 80386 20/8 MHz Norton V4.0 SI 22
- Phoenix BIOS
- 1MB expandable to 16MB RAM
- 80387 Coprocessor Socket
- 220W Power Supply
- 1.2MB Floppy Drive
- Ports: 2 Serial, 1 Parallel, 1 Game
- Dual Floppy/Dual H.D. Controller
- Monochrome Graphic Card
- 101 Key Enhanced Keyboard
- TTL Monitor 12"
- 40MB Hard Disk (28MS)

Options: Call

\$1,995⁰⁰

DISK DRIVES

Fujitsu 360k	\$69	Citizen CD 120	\$149
Fujitsu 1.2MB	\$89	Citizen CD 180	\$189
Teac	\$75	HPLASAR Serial2	\$1699
Teac 1.2MB	\$89	Epson LX-800	\$219
Teac 3 1/2" 1.4MB	\$89	Epson LQ-500	\$379
20MB Hard Disk Kit	\$279	Toshiba 321 XL	\$519
30MB Hard Disk Kit	\$309	NEC P2000	\$369
KL320 (40MS)	\$215	Call for prices of other brands	
8425 Miniscribe	\$249		
8438 30MB Miniscribe	\$259		
3650 40MB Miniscribe	\$349		
3675 60MB Miniscribe	\$379		
ST-157 49MB 3 1/2"	\$479		

PRINTERS

300/1200	\$79
2400 external	\$139
2400 internal	\$99

MONITORS

Samsung amber	\$79	640K TurboMotherboard	\$80
Samsung EGA color	\$359	10MHz TurboMotherboard	\$85
Samsung RGB color	\$259	Multi I/O w/disk contrir	\$59
NEC Multisync	\$559	640K RAM card	\$39
Sony Multiscan	\$619	2MB Expansion card	\$89
HGC-compat.mono card	\$49	RS232 2-port card	\$35
Color graphic card	\$49	4-serial port card	\$79
EGA Paradise 480	\$149	Game I/O card	\$15
VGA Paradise	\$279	384K Multifunction card	\$69
Genoa Super VGA	\$299	FCC-app. slide XTcase	\$29
		150W power supply	\$49
		XT keyboard	\$42
		Clock Card	\$19
		Floppy Controller	\$19

MOUSE

Logimouse C7	\$69
Logimouse H1 Res	\$99

PC/XT

McTek 286-20MHz	\$359
McTek 286-12	\$259
McTek 386-16SX	\$429
McTek 386-20MHz	\$699
McTek 386-24MHz	\$899
Locking slide case	\$59
200W power supply	\$65
Enhanced keyboard	\$59
WD FD/HDC	\$129
DTC FDC/HDC 1:1	\$189
3MB EMS (3K)	\$99

PC/AT

Kingtech CRT Portable Kits: XT/AT (power supply, case keyboard, monitor)	\$380/\$410
Eprom burner 4-socket ..	\$139
LCD Portable	\$799
Plasma Portable Kits ..	\$1499
AC power strips	\$15
Diskette file box	\$9
Printer or serial cable	\$8
Archive Tape Backup 40MB	\$339
XT 10MHz 640k 2 Drive System	\$759

DESKTOP

McTek Systems, Inc. • 1521 San Pablo Avenue • Berkeley, CA 94702 • 415-525-5129

Reader Service Number 42

MICRO CORNUCOPIA, #51, Jan-Feb, 1990 75

386, OS/2 (forget PM), and Nu-Mega's SoftIce (I've heard wonderful things about it from C developers).

- An Intel marketing type did a presentation at the Longhorn SOG. Fortunately he had been a technical type in a previous life and he knew the 486. He said that as far as architecture is concerned, there's no real difference between the 386 and the 486. So, unlike the jump from 286 to 386, the 386 will do everything the 486 can do.

That doesn't mean there's no reason to buy a 486. The 486 will run instructions in half the clocks (an average of 2) per instruction. Plus the 486 is supposed to be out the chute at 33 MHz with 40 MHz and 50 MHz parts very soon now. (His words, not mine.)

The initial production will run about \$1,000 per part with prices scheduled to settle into the \$300 range by fall 1990. That sounds high, but it includes an on-board 387 (487?). Float times will improve drastically because the ALU won't have to go through all the motions of waking up an external chip, giving it control, and waiting for notification that the arithmetic's finished.

Also, there's 8K of on-board cache and transfers from cache to the ALU happen in 128-bit chunks. Interesting.

Now the 386SX (sounds like it should have a V6 and fuel injection) is a bit controversial. Some folks think it's a 386 that Intel crippled to make it easy for manufacturers (IBM) to sell

their 286 boards as 386 machines. That's probably true; but if you need the 386 architecture and can live with a 286 bus, it's probably not a bad way to go.

Anyway, with 386 and 386SX prices way down and with the 386's ability to watch itself run (necessary for the ICE and serious multi-anything), plus its ability to create memory segments of any size (it can view all your RAM as one large segment), I'd think twice before purchasing another 286.

On the other hand, those little baby AT boards are so cheap and those tiny tower cabinets so cute, they make tempting replacements for 7 MHz XT "turbo" boards. I've heard that most of the baby AT cards will accept all the original XT plug-in cards, including the hard drive controllers. (Ask your dealer to try it before he says it won't work.)

Attention Spies

There's no longer a reason to dig out your stock of master keys ... no need to sneak into the Micro C office at midnight. We're now available on microfilm. This is not part of a plot but rather a way to make *Micro C* available to ants, amoeba, gnats, and squinty-eyed librarians.

If you want to order a reprint or just an article from an issue that's out of print, you can contact:

University Microfilms International
300 North Zeeb Road
Ann Arbor, MI 48106-1346
(313) 761-4700

Of course, if the issue is still available, you'll probably want to get the original version directly from Micro C. See the back issues list on page 94 for more information.

Transparent ROMs

About half way through the Embedded Systems article in this issue, I realized I was:

1. Very interested in the C-Thru-ROM package, and
2. Very hungry (it being halfway through my lunch hour). Knowing I could either:

1. Call Datalight immediately, meant missing lunch, or
2. Grab a byte of lunch and then call Datalight.

I called Datalight on a full stomach.

That was fortunate, because I had some zingy questions:

"Is it all that easy?"

"As easy as what?"

"Does it only work with Microsoft C 5.X?"

"No."

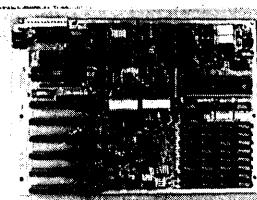
It turns out that C-Thru-ROM 1.50 also works, in a limited fashion, with Turbo C 2.0. By early January they're supposed to have a new release that does everything with Turbo C.

Then Datalight sent me the package. Unfortunately, their sample program (SIEVE.X) didn't work because they hadn't included the right .MAP file. Did we have Microsoft C? A quick perusal showed the cupboard was nearly bare.

No Microsoft C, but some extended scrounging turned up Quick C (probably too quick) and Turbo 2.0. Ah ha!

So I recompiled, linked, and MAPped with Turbo. Doing all the usual new-package fumbling, it worked. I connected

68000 SINGLE BOARD COMPUTER



- 512/1024K DRAM
- FLOPPY DISK CONTROLLER
- 4 RS-232 SERIAL PORTS
- REAL TIME CLOCK

NEW LOWER PRICES!

BASIC KIT (8MHz) - BOARD, MICROPROCESSOR, HUMBUG MONITOR/
BASIC IN ROM, 4K SRAM, 2 SERIAL PORTS \$200

PACKAGE DEAL - COMPLETE KIT WITH 10MHz MICROPROCESSOR,
SK*DOS OPERATING SYSTEM, 512K DRAM \$499

SYSTEM BOARD (12MHz) - ASSEMBLED/TESTED, 1MEG RAM, 6 PC/XT
PERIPHERAL PORTS, SK*DOS \$699

COMPLETE INFORMATION AVAILABLE UPON REQUEST

PERIPHERAL TECHNOLOGY

1710 CUMBERLAND POINT DR., #8

MARIETTA, GA 30067

404/984-0742

COD/MASTER CARD/VISA/CHECK

SK*DOS IS A TRADEMARK OF STAR-K SOFTWARE SYSTEMS CORP.

my laptop to my AT via a serial cable, ran their little kernel routine on the laptop (so it pretends it has a kernel ROM plugged in), and fired up Datalight's remote debugger.

Wow, the debugger downloaded SIEVE to the laptop via the serial connection and then let me single step through the program, either in C or in assembly. I could even watch the variables....Oops. Unfortunately, it didn't understand most of the variables. Oh yeah, this version only understands Turbo's globals; next version adds the locals and statics.

Ah well, I went back to the source and made all the variables global. Wow, the AT's screen displayed the contents of C variables as the laptop ran one line at a time.

This isn't a particularly fast environment (the serial link between the two systems is one reason), so they suggest you break up large routines for debugging. But a wonderful environment? Yeah. Seriously.

Datalight C-Thru-ROM \$495
17505 68th Ave. N.E., Suite 304
Bothell, WA 98011
(206) 486-8086

This Issue Is Listing

Boy, are we on somebody's list this issue. Gary threw in 9K of Turbo Pascal with Tidbits. Bruce Eckel's voice contained

16K of C. Russ Eberhart and Roy Dobbins couldn't neural network without 18K of C. We bit our lips and made room. Then, not to be outdone, Scott Ladd sent in 26K with his C column and Greg Landheim is our champion with 34K of 3-D.

Between Scott and Greg, that's 18 pages of C, space for at least four (unlisted) articles.

All right, already. If you authors can't be a little briefer, (you know about "little briefers," right?) then you'll be relegated to the BBS and the issue disk(s). Starting right now. (Great Scott! What'll Mr. Ladd say?)

So, to get Scott and Greg's code (along with the printed code), you'll just have to:

1. Download ISSUE-51.ARC from the Micro C BBS. (503) 382-7643. (300-1200-2400, 8,1,N, 24 hours)

2. Send \$6 (\$8 foreign) to Micro C, P.O. Box 223, Bend OR 97709, or call (800) 888-8087 (Visa/Mastercard) and ask for the Issue 51 disk. Our issue disks contain all the code, printed or not. Price includes postage. (Sometimes an issue disk is as many as three disks, but it's still only \$6).



David Thompson, Editor and Publisher



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

ICs PROMPT DELIVERY!!!

SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN FOR OCT. 29, 1989

OUTSIDE OKLAHOMA: NO SALES TAX!

DYNAMIC RAM			
SIMM	AST Prem386/33Mhz	\$300.00	
SIMM	(1) 256Kx36	80 ns	300.00
SIMM	{ 1Mx9	80 ns	120.00
SIMM	{ (2) 256Kx9	100 ns	35.00
1Mbit	1Mx1	100 ns	10.99
41256	256Kx1	60 ns	4.75
41256	256Kx1	80 ns	3.75
41256	256Kx1	100 ns	2.80
41256	256Kx1	120 ns	2.65
4464	64Kx4	120 ns	3.75
41264	(3) 64Kx4	100 ns	7.50
EPROM			
27C1000	128Kx8	200 ns	\$20.75
27512	64Kx8	200 ns	7.80
27256	32Kx8	150 ns	6.50
27128	16Kx8	250 ns	3.95
STATIC RAM			
62256P-10	32Kx8	100 ns	\$11.95
6264P-12	8Kx8	120 ns	4.50
6116AP-12	2Kx8	120 ns	4.25

PS2/70
PS2/80
30/286
1Mbyte

(2)
FOR:
AT&T
AT&T
COMPAG
DELL
MAC'S
MYLEX
PS/2
Tandy
WE
WD

VIDEO
RAM

80387-16
80387-20
80387-35

80387-8
80387-16
80387-20
80387-35

\$21.50
\$215.00

8087-2
\$125.00

8087-2
\$125.00

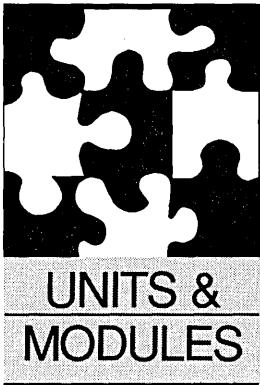
OPEN 6 1/2 DAYS, 7:30 AM-10 PM: SHIP VIA FED-EX ON SAT.

SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY:

Th \$2 \$6.25/lb
Fr: P-1 \$14.25/lb

MasterCard/VISA or UPS CASH COD
MICROPROCESSORS UNLIMITED, INC.
24,000 S. Peoria Ave., BEGGS, OK. 74421

(918) 267-4961
No minimum order. Please note: prices subject to change!
Shipping, insurance extra, up to \$1 for packing materials.



Balanced Trees

What happens when your sorts are off by one? How do you balance trees? (Without a chain saw....)

About a week after Issue #49 came out, Ed Stimpson called me from Salt Lake City. He had obtained a downloaded copy of the GenSort module and the routines didn't work. All sorts of possibilities raced through my mind. Had the file been corrupted when I uploaded it to Micro C? Had I uploaded the right file? Did Ed know how to use the GenSort module? I asked myself all these questions.

After talking with Ed a few minutes, I was confident he knew how the module worked. It never occurred to me that I might have written buggy code. Pride can be very hard to swallow, and it leaves a bitter aftertaste.

I had tested the code with what I thought was a very general test case. In reality, because of a minor oversight concerning low level pointer manipulation, I had only tested a very specific case. What a way to be reminded to test my code *very, very* carefully and to cut back on after-midnight programming.

Thoroughly humbled, I listened carefully as Ed explained the problem and his solutions. If you are using the GenSort module, you'd probably benefit by contacting Ed. He has used the sort in a cross-reference program and is building a shell around it so it can replace the DOS sort command. By press time, he will have added the GenAvlTree module to his sorts.

GenSort Changes

Ed had discovered several problems with the GenSort module. Even though the GenBinTree module was deleting the tree node, in GenSort I had left out

the code to delete the key and data memory areas after the sort had released the information. I inserted a deletion of the key memory in GenBinTree and a deletion of the data memory in the GenSrtReleaseF function.

The more important problem was that the key building routine for strings had an index off by one. I hadn't caught the problem because I set up my index incorrectly. If you use strings in the data then you must count Turbo Pascal's length byte. If you pass a string[25] to GenSort, it receives 26 bytes. The first one is the length byte. I do most of my programming in Modula-2 and Fortran which, like C, do not have a length byte. If you want to use a string for a key then the correct offset is the second byte of the string.

AVL Trees

If you insert semi-sorted data into a binary tree, the tree begins to resemble a linked list. The search time for this tree becomes unpredictable, quite long if the search key is at the end of this pseudo list. One solution, as suggested by Adelson-Velskii and Landis, is to keep the tree balanced. They defined a balanced tree as one where, in every node, the heights of the left subtree and right subtree differ by, at most, 1.

The storage overhead to keep the tree balanced is a balance factor (the difference in height between two subtrees), stored with each node in the tree. This can be as little as two bits per node since only the values of -1, 0, and 1 are required. If the balance factor becomes -2 or +2 because of an insertion or deletion in the tree, an appropriate rotation around the nodes with balance factors of +2 or -2 restores the balance.

GenAvlTree is pretty much a drop-in replacement for GenBinTree. (See Figure 1.) The procedure names are almost

By Michael S. Hunt
2313 N. 20th
Boise, ID 83702
(208) 336-7413

identical. Some of the parameter order has changed and a new parameter has been added. The "boo" parameter is needed for the routine's recursive calls. They need to be provided when called from GenSort, but not initialized.

I will include the source for my corrected GenSort, GenAvlTree, and several sample programs on the issue disk. If you have more questions about the routines, contact Ed or myself.

Next Time

In the next issue I'll start discussing Turbo Pascal, Modula-2, and objects. Through issue #49, I've written all the source code for the column in both Turbo Pascal and Modula-2. Although the two languages are very similar, if you are squeezing them for performance or using many of Turbo Pascal's extensions, translating between the two can be very time consuming.

Because I have started a large project for Micro C and I would like to spend more time on the column, I will provide only the Turbo Pascal source code. And thanks Ed, for your time, effort, and suggestions.

Ed Stimpson
3862 Millcreek Rd.
Salt Lake City, UT 84109

References

G. M. Adelson-Velskii, E. M. Landis; *Doklady Akademii Nauk SSSR*, 146, (1962); (English translation in Soviet Math)

Holub, Allen; "C Chest," *Dr. Dobb's Journal*; August 1986

Knuth, Donald E.; *The Art of Computer Programming*, Vol. 3; Addison-Wesley; 1973

Wirth, Niklaus; *Algorithms & Data Structures*; Prentice-Hall; 1986



Figure 1—Balanced Tree Code

```

UNIT GenAvlTree;
(*
   Michael S. Hunt
   Released as Public Domain Software
*)
INTERFACE

TYPE dataType = ^dataType;
   dataType = array [1..32767] of CHAR;
treePtr = ^treeNode;
treeNode = RECORD
   data, key : dataType;
   dataLen, keyLen : WORD;
   llink, rlink : treePtr;
   bf : SHORTINT
END;

PROCEDURE GenAvlTrIns(k, d: dataType;
   keyLen, dataLen: WORD;
   VAR p : treePtr; VAR h : BOOLEAN);
PROCEDURE GenAvlTrDel(k: dataType; VAR p: treePtr;
   VAR h : BOOLEAN);
PROCEDURE GenAvlTrDis(root: treePtr; tab: INTEGER);
PROCEDURE GenAvlTrRetDelSmRec(VAR p: treePtr;
   VAR key, data: dataType;
   VAR keyLen, dataLen: WORD; VAR h: BOOLEAN);

IMPLEMENTATION

CONST tabinc = 3;
VAR boo : BOOLEAN;

FUNCTION CompArr(VAR arr1, arr2: dataType;
   len: WORD) : INTEGER;
VAR k : WORD;
BEGIN
   k := 1;
   CompArr := 0;
   WHILE k < len DO BEGIN
      IF arr1[k] < arr2[k] THEN BEGIN
         CompArr := -1;
         k := len + 1
      END
      ELSE IF arr1[k] > arr2[k] THEN BEGIN
         CompArr := 1;
         k := len + 1
      END
      ELSE
         Inc(k);
   END
END;

PROCEDURE BalLeft(VAR p: treePtr; VAR h: BOOLEAN);
VAR p1, p2 : treePtr;
   b1, b2 : SHORTINT;
BEGIN
   IF p^.bf = -1 THEN
      p^.bf := 0
      ELSE IF p^.bf = 0 THEN BEGIN
         p^.bf := 1;
         h := FALSE
      END
      ELSE BEGIN
         p1 := p^.rlink;
         b1 := p1^.bf;
         IF b1 >= 0 THEN (* single RR rotation *)
            BEGIN
               p1^.rlink := p1^.llink;
               p1^.llink := p;
               IF b1 = 0 THEN BEGIN
                  p^.bf := 1;
                  p1^.bf := -1;
                  h := FALSE
               END
               ELSE BEGIN
                  p^.bf := 0;
                  p1^.bf := 0
               END;
               p := p1
            END
         ELSE BEGIN
            p2 := p1^.llink;
            b2 := p2^.bf;
            p1^.llink := p2^.rlink;
            p2^.rlink := p1;
            p^.rlink := p^.llink;
            p2^.llink := p;
            IF b2 = 1 THEN
               p^.bf := -1
            ELSE
               p^.bf := 0;
            IF b2 = -1 THEN
               p^.bf := 1
            ELSE
               p^.bf := 0;
            p := p2;
            p2^.bf := 0
         END
      END
      END;
   END; (* BalLeft *)

PROCEDURE BalRight(VAR p: treePtr; VAR h: BOOLEAN);
VAR p1, p2 : treePtr;
   b1, b2 : SHORTINT;
BEGIN
   IF p^.bf = 1 THEN
      p^.bf := 0
   ELSE IF p^.bf = 0 THEN BEGIN
      p^.bf := -1;
      h := FALSE
   END
   ELSE BEGIN
      p1 := p^.llink;
      b1 := p1^.bf;
      IF b1 <= 0 THEN (* single LL rotation *)
         BEGIN
            p^.llink := p1^.rlink;
            p1^.rlink := p;
         END
      ELSE BEGIN
         p2 := p1^.rlink;
         b2 := p2^.bf;
         p1^.rlink := p2^.llink;
         p2^.llink := p1;
         IF b2 = 1 THEN
            p^.bf := -1
         ELSE
            p^.bf := 0;
         p := p2;
         p2^.bf := 0
      END
   END
END;

```

```

IF b1 = 0 THEN BEGIN
    p^.bf := -1;
    p1^.bf := 1;
    h := FALSE
END
ELSE BEGIN
    p^.bf := 0;
    p1^.bf := 0
END;
p := p1
END
ELSE BEGIN
    p2 := p1^.rlink;
    b2 := p2^.bf;
    p1^.rlink := p2^.llink;
    p2^.llink := p1;
    p^.llink := p^.rlink;
    p2^.rlink := p;
    IF b2 = -1 THEN
        p^.bf := 1
    ELSE
        p^.bf := 0;
    IF b2 = 1 THEN
        p^.bf := -1
    ELSE
        p^.bf := 0;
    p := p2;
    p2^.bf := 0
END
END; (* BalRight *)

```

```

PROCEDURE GenAvlTrIns(k, d: dataPtr; keyLen,
                      dataLen: WORD;
                      VAR p : treePtr; VAR h :BOOLEAN);
VAR p1, p2 : treePtr;
BEGIN
IF p = NIL THEN (* insert *)
BEGIN
    GetMem(p, SizeOf(treeNode));
    h := TRUE;
    p^.data := d;
    p^.key := k;
    p^.dataLen := dataLen;
    p^.keyLen := keyLen;
    p^.llink := NIL;
    p^.rlink := NIL;
    p^.bf := 0
END
ELSE IF CompArr(p^.key^, k^, p^.keyLen) <=0 THEN
BEGIN
    GenAvlTrIns(k,d,keyLen,dataLen,p^.rlink,h);
    IF h THEN (* right branch has grown *)
BEGIN
    IF p^.bf = -1 THEN BEGIN
        p^.bf := 0;
        h := FALSE
    END
    ELSE IF p^.bf = 0 THEN
        p^.bf := 1
    ELSE IF p^.bf = 1 THEN BEGIN
        p1 := p^.rlink;
        IF p1^.bf = 1 THEN (*single RR rot*)
BEGIN
            p^.rlink := p1^.llink;
            p1^.llink := p;
            p^.bf := 0;
            p := p1
        END
        ELSE (* double RL rotation *)
BEGIN
            p2 := p1^.llink;
            p1^.llink := p2^.rlink;
            p2^.rlink := p1;

```

```

        p^.rlink := p2^.llink;
        p2^.llink := p;
        IF p2^.bf = 1 THEN
            p^.bf := -1
        ELSE
            p^.bf := 0;
        IF p2^.bf = -1 THEN
            p1^.bf := 1
        ELSE
            p1^.bf := 0;
        p := p2
    END;
    p^.bf := 0;
    h := false
END
END (* GenAvlTrIns *);

PROCEDURE GenAvlTrDel(k :dataPtr; VAR p: treePtr;
                      VAR h : BOOLEAN);
VAR q : treePtr;

PROCEDURE del(VAR r : treePtr; VAR h : BOOLEAN);
BEGIN
    IF r^.rlink <> NIL THEN BEGIN
        del(r^.rlink, h);
        IF h THEN BalRight(r, h)
    END
    ELSE BEGIN
        q^.key^ := r^.key^;
        q := r;
        r := r^.llink;
        h := TRUE
    END
END; (* del *)

BEGIN
    IF p = NIL THEN (* key not in tree *)
    ELSE IF CompArr(p^.key^, k^, p^.keyLen) = 1 THEN
        BEGIN
            GenAvlTrDel(k, p^.llink, h);
            IF h THEN BalLeft(p, h)
        END
    ELSE IF CompArr(p^.key^, k^, p^.keyLen)=-1 THEN
        BEGIN
            GenAvlTrDel(k, p^.rlink, h);
            IF h THEN BalRight(p, h)
        END
    ELSE (* delete p^ *)
        BEGIN
            q := p;
            IF q^.rlink = NIL THEN BEGIN
                p := q^.llink;
                h := TRUE
            END
            ELSE IF q^.llink = NIL THEN BEGIN
                p := q^.rlink;
                h := TRUE
            END
            ELSE BEGIN
                p^.bf := 1;
                p2 := p^.rlink;
                p2^.bf := -1;
                p1 := p2^.llink;
                p1^.bf := 0;
                p := p2;
                p^.bf := 0;
                h := false
            END
        END
    END; (* GenAvlTrDel *)
END;

(* FreeMem(q^.data, q^.dataLen); { resp.
   of calling program } *)
FreeMem(q^.key, q^.keyLen);
FreeMem(q, SizeOf(treeNode));
END;
END; (* GenAvlTrDel *)

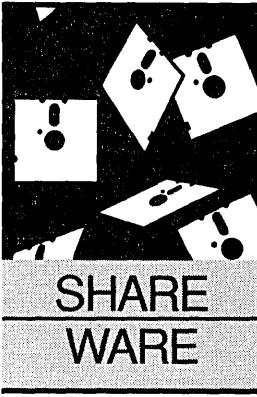
PROCEDURE GenAvlTrDis(root: treePtr; tab: INTEGER);
VAR space, k : INTEGER;
BEGIN
    IF root <> NIL THEN BEGIN
        IF root^.rlink <> NIL THEN BEGIN
            GenAvlTrDis(root^.rlink, tab + tabinc)
        END;
        FOR space := 1 to tab DO
            write(' ');
        FOR k := 1 to 6 DO
            write(root^.data^[k]);
        writeln(' ',root^.bf);
        IF root^.llink <> NIL THEN
            BEGIN
                GenAvlTrDis(root^.llink, tab + tabinc)
            END
        END (* root # nil *)
    ELSE
        writeln('Nil')
    END (* GenAvlTrDis *);

PROCEDURE GenAvlTrRetDelSmRec(VAR p :treePtr;
                               VAR key, data :dataPtr;
                               VAR keyLen, dataLen:WORD; VAR h:BOOLEAN);
VAR q : treePtr;
BEGIN
    q := p;
    WHILE p^.llink <> NIL DO BEGIN
        p := p^.llink
    END;
    data := p^.data;
    dataLen := p^.dataLen;
    key := p^.key;
    keyLen := p^.keyLen;
    p := q;
    GenAvlTrDel(key, p, h);
END; (* GenAvlTrDel *)

BEGIN
END.

♦ ♦ ♦

```



Victim Of Success

Anthony Barcellos

P.O. Box 2249
Davis, CA 95617-2249
Voice: (916) 756-4866
Data: (916) 758-1002

It's always great when someone betters himself, especially when he starts out in academics. In this, his last column, Tony tackles text and graphics, appropriate tools for his next life.

So long, everyone. This is my 15th—and final—shareware column for *Micro C*. A lot of user-supported software has streamed through this column during the past 2½ years, and there's no end in sight. I have only sampled the flood.

Although there remain so many attractive packages to examine, I must redirect my efforts toward other interests. My arrival in the pages of *Micro Cornucopia* coincided with my appointment to the mathematics faculty of American River College in Sacramento. While I was delighted by this long-sought opportunity to teach full-time, I couldn't anticipate all the repercussions—all of them, so far, good.

While a California civil servant, I was fairly successful as a freelance mathematical writer. My articles on Mandelbrot's fractal geometry won a couple of awards and my interview of Benoit Mandelbrot was published in *Mathematical People*. (Yes, I now admit that I have long been smitten with fractals—just like everyone else at *Micro C*.)

My stock as a writer, however, has risen enormously since I returned to academia. It helps that American River College already has several successful authors on its faculty, so perhaps I'm gilt by association.

During the past summer, I was courted by several publishers looking for textbooks. Over the next two years my name will appear on texts and supplements for algebra, trigonometry, and calculus courses. It will be a lot of work, but I'm looking forward to it.

In the meantime, I have to reorder my priorities. I've submitted my resignation as software librarian to the Sacramento PC Users Group. I have also decided to withdraw from this column, much as I regret doing so. I hope to retain the editorship of *Sacra Blue*, which in

the future will be my only significant commitment to users group activities.

Parting Glances

There are a few loose ends that I would like to tie up as I depart. I've long been intrigued by desktop publishing and sophisticated tools like Adobe's PostScript.

Under pressure from Microsoft and Apple, Adobe has decided that they have to give up their secrets and make PostScript an open standard. PostScript's power has attracted many users despite its high costs. Recent developments may make PostScript more available than ever before (in either the Adobe version or as clones from competitors).

HortIdeas Publishing latched on to PostScript early in the game, putting out a shareware drawing program that generates PostScript files. One attraction of PictureThis is its low cost—only \$50. But as a PostScript-based application, its attractions are many. Besides being compatible with every desktop publishing program and high-end word processor that works with PostScript printers, PictureThis enjoys PostScript's immunity from the "jaggies."

Recall that PostScript incorporates a scaling technology that adjusts for changes in size without creating the jagged edges and broken lines that appear whenever you magnify bit-mapped graphics. Jagged bitmaps can be imported into PictureThis and then traced to create a scalable version. Nice!

One problem with PictureThis may have been addressed by the time you read this. The program currently requires a CGA or compatible graphics adapter. Given that PostScript applications are toward the high-end of today's applications, most users who would be interested in PictureThis are probably already using EGAs or VGAs. PictureThis definitely needs to migrate to a higher platform.

Congratulations to Greg and Pat Williams of HortIdeas for being ahead of the crowd in moving shareware into the PostScript domain. This could be the start of something big.

PictureThis, \$50
HortIdeas Publishing
Greg & Pat Williams
Route 1, Box 302
Gravel Switch, KY 40328
(606) 332-7606

Quick Editor Keeps Moving

SemWare has updated QEdit, the "Quick Editor," to version 2.08. New features include sorting and support for macros invoked from the command line. Those of you with laptops will appreciate the optional large-block cursor that shows up so well on LCD screens.

QEdit costs \$54.95 with \$3 for shipping. As before, it fits nicely on floppy-based systems and can run in only 64K. (Where did they find a 64K PC to test it on?) They also have an OS/2 version at the same price. It offers several OS/2-specific features. A whole new area is opening up here and SemWare is on the scene.

QEdit 2.08
\$54.95 plus \$3 shipping
SemWare
4343 Shallowford Road, Ste. C-3
Marietta, GA 30062-5003
(404) 641-9002

Browsing Quickly

Quicksoft has long been famous for PC-Write, the leading shareware word processing program. A new Quicksoft program called PC-Browse is in beta test and should be out by the time you see this.

PC-Browse is a pop-up utility that lets you look at files or disk directories while you work in your application program. But that description doesn't do it justice. PC-Browse lets you search for text files by matching a given word or phrase inside the file. It hunts for files

Under pressure from Microsoft and Apple, Adobe has decided that they have to give up their secrets and make PostScript an open standard.

by name across drives and directories. It also lets you paste information into your application.

The most unusual and exciting feature of PC-Browse is the hypertext tool that you can use to create help systems or linked menus. Quicksoft has started advertising in user group newsletters and they've gotten rave testimonials from their beta testers. PC-Browse may soon be encroaching on SideKick's territory. Watch for it soon at a users group near you, or buy it directly from Quicksoft for \$54.

PC-Browse, \$54
Quicksoft, Inc.
219 First Avenue North, #224
Seattle, WA 98109
(800) 888-8088

Zipping Along

PKWare has released version 1.01 of PKZIP/PKUNZIP. The new ZIP utilities can work with files created with version 0.92, but the compatibility is only one

way. PKZIP still produces quick and tight file compression, while PKUNZIP performs the reverse operation even more quickly.

PKWare is branching out a little, offering PKZFind to search for files across disks and directories and automatically log you onto the directory where the desired file resides. It will even peek into zipped files to see if the target is hiding there.

PKZIP/PKUNZIP 1.01, \$50.50

PKZFind, \$25
PKWare, Inc.
7545 North Port Washington Road
Glendale, WI 53217-3422
(414) 352-3670

Post Script

Last time I asked where users like to get their shareware. Although I am vacating this space, I plan to collect any responses to pass along to the good editors here at Micro C. If I accumulate a worthwhile list, perhaps it will see print in these pages.

Goodbye, folks. Keep registering that shareware. And remember the motto of the Sirius Cybernetics Corporation: "Share and enjoy!"

Editor's note: Apparently we're looking for a new shareware columnist. Any librarians out there yearning to write? (You can even turn in a friend.)



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

0	1	1
1	0	0
1	0	0

TIDBITS

By Gary Entsminger
P.O. Box 2091
Davis, CA 95617

A Case Of Optical Character Recognition:

From .PCX To .TXT Via A Neural Network

Recognizing characters isn't the easiest thing for a computer (but then, recognizing friends isn't either). Gary applies a neural net to a pile of type-written pages and.... At least he doesn't paper over the problems.

Amy's desk, piled high with typewritten pages, was a research assistant's nightmare. A gothic soap opera. For years her boss and a legion of bosses had accumulated enough reports, abstracts, and just plain gossip to fill up the Greek Theatre. Now, they wanted all those pages WordStar-compatible.

Her mission (would she take it?)—was to retype the pages, line by line, character by character, until her fingers and mind numbed with futility. Going crazy, really.

She looked at me. Glancing up, I pretended I was handsome, dark, and perhaps mysterious; as if I were Fred MacMurray in *Double Indemnity* or Jack Nicholson in *Chinatown*—two of my favorite movies. Amy loves movies.

"Can't you do something?" she asked.

I cleared my throat in my hand, considered the possibilities, remembering the plots of *Double Indem* and several other "B" movies. A diabolic solution like murder, of course, was out of the question. I really liked her boss. Anyway, it's messy and immoral. Fred found that out. I took the Jack Nicholson approach.

"Well, Ms. Seidel," I said, "I don't know what I can do; you're in an unusual situation."

"Not really," she interrupted, "a lot of people are in my position."

I noticed her legs, long and crossed and turned a little towards me, but not far enough to be suggestive. I glanced past her, out toward the grassy hill that led up and then broke into a forested valley.

Sighing, I didn't know what to do; I said, "Okay, I'll poke around a little and get back to you."

She may have mistaken my tone. I don't know.

"Thanks," she managed a semi-smile, semi-sweet, and semi-ambivalent.

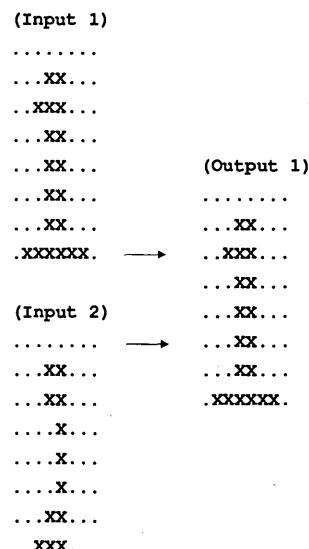
"Save the thanks," I muttered underneath my breath, "until I deserve it."

Her problem was simple enough, in theory, but I must admit I felt a little uneasy. She had a bunch of typed paper; she wanted it to appear magically in WordStar. A case of optical character recognition, or OCR as we say in the business. She was right: there are a lot of people in her position.

I investigated and got my hands on a few tools—

(1) A scanner—which I could use to get an image of a page into a computer. An “image,” I say, since scanners inherently scan pixels, not characters. A graphic image of a character (a matrix of pixels) may look like a character to you and me, but to a computer it’s garbage.

Figure 1—Both Inputs Produce Output 1



Images can be beautiful, of course; but moving, entering, and deleting characters, pixel by pixel, isn't a giant step for computer science.

(2) A graphics editor—which I could use to manipulate pixels (to realign them, for example).

(3) A neural network—which I could use to evaluate and correct input patterns.

Neural networks effectively filter input to produce output. More specifically, a net looks for patterns in a set of examples (called a training set, the input) and learns from these examples to produce new patterns (the output). Then using what it's learned (these input/output pattern-associations), it classifies a new set of examples (called a test set).

The net uses patterns of bits instead of individual bits to process information. Since a network recognizes patterns, not bits, no particular bit is crucial for the network to recognize a pattern successfully. The network will accept minor variations in bit alignment (see Figure 1) and still produce the right pattern.

You input a pattern into a neural network; it compares the pattern with the patterns it knows (the patterns it learned from its training set); and outputs a "best guess" pattern. (*Note: the network's best guess might even be no pattern. We tell the network (using a threshold value) how much variation (or error) we'll accept. If no output exceeds the threshold, the network could produce a space, for example.*)

Scanning typewritten pages is, at best, an imperfect operation. We can expect the scanned patterns (our input) to be incomplete or contain "errors," such as bit misalignment. A neural network can correct these kinds of problems and

Since a network recognizes patterns, not bits, no particular bit is crucial for the network to recognize a pattern successfully.

produce reliable output by making good guesses.

We train the network by showing it a set of input patterns and their corresponding output patterns. For example, we show the network an 8 by 8 bit picture of a character and the corresponding ASCII character. We say, in effect, when you see this input pattern (a bit picture), output an ASCII character.

This part is tricky and requires a shuffling of examples. Eventually, after we toggle a switch or two and add or delete a few examples, the network "learns" to "guess" well enough for our problem. A pattern is good enough when it reaches the threshold value we set.

I shopped around for a neural network. I was in a hurry and it beat programming one from scratch, tho' Russ Eberhardt and Roy Dobbins show us how to do just that in their neural net-

work article this issue. I found four contenders—

NeuroShell (from Ward Systems)

NeuralWorks (from Neuronware)

NNT (Shareware)

BrainMaker (from California Scientific).

Each works well alone, but has even more interesting potential for use within other systems. I found, at first surprisingly, that developers have taken distinctive approaches to implementing neural networks. Everything from the user interface to operational theory varies.

I conclude that neural networks are at that inchoate stage where they share some fundamental qualities (such as the use of training sets, thresholds, normalization, etc.), but diverge widely in the details. I chose BrainMaker for the OCR project because it:

(1) is relatively inexpensive;

(2) has a neat batch mode which I can call from my programs;

(3) is fast enough (at least with small sets);

(4) allows two-dimensional picture input (which makes it easy for me to see and edit 8 by 8 bit characters).

I had three tools.... It was a start, the nucleus of a project, but much was missing. As usual, the tools didn't fit together well enough to solve Amy's problem. My computing skills had gotten my foot in the door, but I still wasn't sure I'd solve anything. I proceeded.

Step 1: I scanned in an image (you can simulate this step in any of the Turbo languages using the Borland Graphics Interface, or BGI)—

(1) Initialize graphics;

(2) Open a Viewport;

(3) Use OutTextXY to create some graphics text;

(4) Save the text to a file.

Step 2: I loaded the image into a graphics editor (PaintShowPlus) and examined it bit by bit. Right away I discovered that text sizes, fonts, styles, and horizontal and vertical alignments vary. Too much variation is a problem. I decided to teach the neural net to handle size, font, and style matters. We teach networks by showing them a training set.

Realignment I did by bit (in PaintShowPlus). I didn't teach my network much, so it wasn't all that smart. A one column pixel shift introduced more variation than it could handle. So I corrected a lot of bits. Ugh! In the future, I'll either train the network more extensively or automate column and row shifts.

Step 3: PaintShowPlus writes three kinds of output files: .TIF, .IMG, and .PCX. BrainMaker, the neural net, wants its input as a text pattern (a matrix of "X"s and periods representing a character). Problem: bridge the abyss between these two.

I had a copy of the PCX Toolkit, which displays, loads, saves, and in various other ways manipulates .PCX images, so I decided to work toward it.

I created a .PCX file with PaintShowPlus. I planned to translate images from the screen instead of from a file, so any file format will do (if you can display it). Eventually I had to create a two-dimensional "text" picture for BrainMaker (see Figure 2).

The programming problem at this juncture—

(1) To prepare input for BrainMaker (in this case, to translate a bit by bit image of a .PCX file, or screen) to a text picture;

(2) To evaluate and translate the output from BrainMaker to a WordStar (or other text editor) readable format.

Objects

I decided to glue everything together in a couple of Turbo Pascal objects: one to translate a graphics screen into a text picture for BrainMaker; and one to translate BrainMaker's output (symbols plus numerical values; see Figure 3) into pure text (i.e., characters).

In sum, the case seemed to hinge on five parts—

- (1) A scanner (or simulated) part;
- (2) A .PCX (or screen display) part;
- (3) A screen to neural network translator;
- (4) A neural network part;

Figure 2—Examples of Brain Maker Input

```
.xxxxx..  
xx...xx.  
xx...xx.  
xx...xx.  
xx...xx.  
xx...xx.  
.xxxxx..  
  
.....  
..xx...  
.xxx...  
..xx...  
..xx...  
..xx...  
..xx...  
.xxxxxx.  
  
.....  
.xxxxx..  
xx...xx.  
....xx.  
...xxx..  
.xxx...  
xx...xx.  
xxxxxxx.  
  
♦ ♦ ♦
```

(5) A neural network to pure text part.

I created Parts 3 and 5 with Turbo Pascal, and ran Parts 2 and 4 out of a Turbo Pascal shell. The Turbo Pascal program (see Figure 4):

- (1) initializes graphics;
- (2) opens a viewport;
- (3) translates the screen (looking at 8 bit x 8 bit images) into a text picture;

(4) writes an output file for BrainMaker;

(5) calls BrainMaker (via a batch file executed by the built-in procedure exec);

(6) translates BrainMaker's output to characters.

See the figure for details or download the entire system from the Micro C BBS.

In brief, note:

- To use the Turbo Pascal exec procedure, you need to reduce the large heap size in order to have enough memory to run the neural network from a batch file. I used 64K for this one. Experiment.
- Viewports and the many constants in my OCR shell are screen specific. An 80 by 25 character screen needs to have an array of 2000 characters (Num_of_characters).

The PCX files on the Micro C BBS are screen-specific. I've set the constant in my code (Figure 4) for a CGA, 600 by 200 screen. The BBS (.EXE) version uses variables instead of constants and allows you to input your screen type at runtime. OCR_shell then sets the adapter type, Viewport size, and displays the correct .PCX for your screen.

Notes & Conclusions

As I told Amy later, my initial results were revealing and incomplete. I initially trained the network to recognize numbers by using a set of number "facts" that came with BrainMaker. Then I tested a screen of numbers I created with the BGI. I wrote text (using OutTextXY and a default 8 by 8 bit font). Results—terrible. BrainMaker couldn't recognize more than 2 or 3.

Text Continued on page 89

Figure 3—BrainMaker Output; 3 Digits Weight and 1 Digit Output

```
.909 0 .001 1 .018 2 .009 3 .001 4 .033 5 .021 6 .050 7 .097 8 .081 9  
.000 0 .988 1 .009 2 .049 3 .034 4 .091 5 .018 6 .067 7 .012 8 .001 9  
.017 0 .034 1 .929 2 .033 3 .002 4 .038 5 .000 6 .050 7 .068 8 .034 9  
.009 0 .017 1 .053 2 .918 3 .000 4 .053 5 .003 6 .020 7 .100 8 .087 9  
.003 0 .085 1 .005 2 .017 3 .911 4 .021 5 .016 6 .027 7 .030 8 .000 9  
.081 0 .014 1 .008 2 .072 3 .000 4 .900 5 .006 6 .042 7 .070 8 .031 9  
.095 0 .025 1 .000 2 .049 3 .001 4 .003 5 .936 6 .020 7 .100 8 .001 9  
.066 0 .046 1 .030 2 .008 3 .000 4 .006 5 .000 6 .972 7 .041 8 .004 9  
.020 0 .000 1 .032 2 .078 3 .001 4 .001 5 .071 6 .007 7 .902 8 .018 9  
.088 0 .000 1 .006 2 .050 3 .000 4 .000 5 .005 6 .064 7 .094 8 .910 9  
  
♦ ♦ ♦
```

Figure 4—Optical Character Recognition Shell

```

Program OCR_Shell
uses
  Crt, Dos, Graph, pcx_tp; {pcx_tp from Genus Toolkit}

var
  F, F2 : text;
const
  { BGI fonts }
  Fonts : array[0..4] of string[13] =
  ('DefaultFont', 'TriplexFont', 'SmallFont',
   'SansSerifFont', 'GothicFont');
  { BGI text directions }
  TextDirect : array[0..1] of string[8] =
  ('HorizDir', 'VertDir');
  Num_of_patterns = 10;
  Num_of_characters = 2000; { for 80x25 viewport }
  Input_file_from_neural_net =
    'C:\TP\EXE\BrainRTS.Out';
  Output_file_for_neural_net =
    'C:\TP\EXE\BrainRTS.In';

```

```

OCR_Output_file = 'C:\TP\EXE\OCR.Out';
PCX_file = 'C:\TP\EXE\o.PCX';
Line_length = 79; Threshold = 0.60;
PCX_type = pcxCGA_6;
type
  Weights = array[1..Num_of_characters]
    of string[4];
  Patterns = array[1..Num_of_characters]
    of string[1];
  { objects }
  NNIPtr = ^neural_net_interpreter;
  neural_net_interpreter = object
    Array_index: integer; First_char, S: string;
    Weight: Weights; Output_pattern: Patterns;
    constructor Init; destructor Done; virtual;
    procedure Get_weights;
    procedure Output_characters;
  end;
  Screenptr = ^screen;
  screen = object
    GraphDriver: integer; { Graphics device driver }

```

Continued on page 88

Data Junction® converts data files!

Any ASCII • EBCDIC • c-tree • Packed data
 Binary • dBASE • R:BASE • Paradox • Xdb
 Excel • 1-2-3 • SC4 • Btrieve • Oracle
 Informix • Clarion • DIF • SYLK
 ... plus many more file types.

- No programming — completely menu driven!
- Selects & rearranges records, fields and characters!
- Instant script file creation for total batch processing!
- Maps new data structures automatically by name!
- Ideal for OEM/run-time applications!

"... an invaluable integration tool." —Byte
 "... more than worth its price." —PC Magazine
 "... a surefire winner." —Computer Language

Standard \$99 — dBASE, ASCII, 1-2-3, Q&A, Enable, DIF, askSam & merge files for WordPerfect, Word & Wordstar
Professional \$199 — Standard formats plus Binary/EBCDIC, SC4, SYLK Paradox, R:BASE, Clarion, Excel, c-tree & Btrieve
Advanced \$299 — Standard & Professional formats plus Informix, Xdb, Oracle & C-ISAM

Call 800/444-1945 or 512/482-0824

MC/VISA • AMEX • COD • Corp. PO.

available for LANs, Xenix & Unix

Tools & Techniques Inc • 1620 W 12 • Austin TX 78703



Reader Service Number 181

"It's Glorious!"

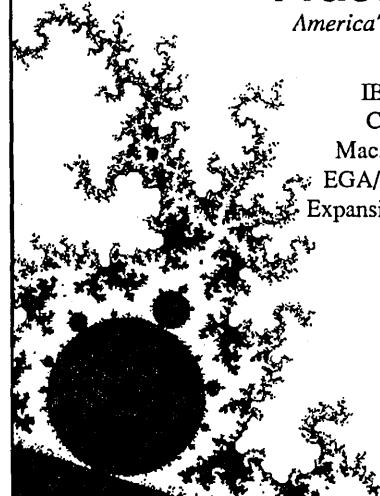
"No color PC compatible is complete without a copy of this;
 get one and see what I mean." — Jerry Pournelle, BYTE

"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 (M/A 89)

FractalMagic 5.0

America's Premier Fractal Program



IBM PC, PS/2 (3.5" or 5.25")

CGA, EGA, & VGA: \$35.00

Mac, Apple II, Atari ST: \$25.00

EGA/VGA Picture Disks: \$15.00

Expansion Modules Coming Soon!

Sintar Software

"Software for the Mind"

P. O. Box 3746

Bellevue, WA 98009

(206) 455-4130

Visa - MC - AmEx

```

GraphMode: integer;      { Graphics mode value }
MaxX,MaxY: word;        { Maximum screen resolution }
ErrorCode: integer;      { Reports graphics errors }
MaxColor: word;          { Max color value available }
pcxReturn: integer;     PixelStatus: integer;
ViewInfo: ViewPortType; constructor init;
destructor done; virtual; procedure Initialize;
end;
var
  OldExitProc : Pointer;   {Saves exit proc addr}
{$F+}

procedure MyExitProc;
begin
  ExitProc := OldExitProc; {Restore exit proc addr}
  CloseGraph;    { Shut down the graphics system }
end; { MyExitProc }
{$F-}

procedure screen.Initialize;
{ Initialize graphics and report errors}
var
  InGraphicsMode: boolean; { Flags graphics init}
  PathToDriver : string; {path to *.BGI & *.CHR}
begin
{ When using Crt & graphics, turn off Crt's
  memory-mapped writes }
  DirectVideo := False;
  OldExitProc := ExitProc; { Save prev exit proc }
  ExitProc := @MyExitProc; { Insert our exit proc}
  PathToDriver := '';
repeat
  GraphDriver := Detect; {detect graphic adapter}
  InitGraph(GraphDriver,GraphMode,PathToDriver);
  ErrorCode := GraphResult; {Preserve err return}
  if ErrorCode <> grOK then { Error? }
begin
  Writeln('Graphics error: ',
         GraphErrorMsg(ErrorCode));
  if ErrorCode = grFileNotFoundException then begin
    Writeln('Enter full path to BGI driver');
    Writeln('or type <Ctrl-Break> to quit.');
    Readln(PathToDriver); Writeln;
  end
  else Halt(1); { Some other error: terminate }
end;
until ErrorCode = grOK;
end; { Initialize }
{ object constructors & destructors }

constructor screen.init;
begin end;

destructor screen.done;
begin end;

constructor neural_net_interpreter.init;
begin end;

```

```

destructor neural_net_interpreter.done;
begin end;
{ object methods }

procedure neural_net_interpreter.Get_weights;
var
  This_weight : string[4];
  This_pattern : string[1];
  Count       : integer; Char_Ptr: integer;
begin
  FOR Count := 1 TO Num_of_characters DO
    begin { Initialize arrays }
      Weight[Count] := ' ';
      Output_pattern[Count] := ' ';
    end;
  Assign(F,Input_file_from_neural_net);
  Reset(F); Array_index := 1;
  WHILE Array_index <= Num_of_characters DO begin
    Readln(F,S); First_char := Copy(S,1,1);
    IF First_char = ' ' THEN begin
      Char_Ptr := 2;
      FOR Count := 1 TO Num_of_patterns DO
        begin
          This_weight:=Copy(S,Char_Ptr,4);
          Weight[Array_index]:=This_weight;
          Char_Ptr := Char_Ptr + 5;
          This_pattern:=Copy(S,Char_Ptr,1);
          Output_pattern[Array_index] :=
            This_pattern;
          Char_Ptr := Char_Ptr + 2;
          Inc(Array_index);
        end;
    end;
    Close(F);
  end;

procedure neural_net_interpreter.output_characters;
var
  Output_char   : string;
  Pattern_count, Char_count, ReturnCode : integer;
  Wt, New_weight : real;
begin
  Assign(F2,OCR_Output_file); Rewrite(F2);
  Array_index := 1; Char_count := 1;
  WHILE Array_index <= Num_of_characters DO
begin
  Pattern_count := 1;
  Wt := 0; Output_char := ' ';
  WHILE Pattern_count <= Num_of_patterns DO
    begin
      Val(Weight[Array_index],New_weight,
           ReturnCode);
      IF New_weight > Wt THEN begin
        Wt := New_weight;
        Output_char :=
          Output_pattern[Array_index];
      end;
    end;
  end;

```

```

Inc(Pattern_count); Inc(Array_index);
end;
IF Wt >= Threshold THEN Write(F2,Output_char)
ELSE
  Write(F2,' ');
IF Char_count > Line_length THEN
begin
  writeln(F2); Char_count := 0;
end;
Inc(Char_count);
end;
Close(F2);
end;
var
  NNI : NNIPtr;

procedure pcx_to_neural_net;
{get PCX; display; convert to txt for BrainMaker.}
var
  SPort      : Screenptr;
  X, Y       : integer;
  XPt, YPt, RowPt : integer;   S: string;
begin
  New(SPort,init);
  WITH SPort^ DO begin
    Initialize;
    Maxx := GetMaxx; Maxy := GetMaxY;
    SetViewPort(0, 0, Maxx, Maxy, ClipOn);
    SetTextStyle(DefaultFont, HorizDir, 1);
    pcxReturn := pcxSetDisplay(PCX_type);
    pcxReturn := pcxFilDisplay(PCX_file, 0, 0, 0);
    IF (pcxReturn = pcxSuccess) THEN begin
      Assign(F,Output_file_for_neural_net);
      Rewrite(F);
      GetViewSettings(ViewInfo); { coordinates of
                                   Viewport }
      XPt := 0; YPt := 0; RowPt := 0;
      WHILE RowPt <= ViewInfo.y2 DO begin

```

```

        WHILE XPt <= ViewInfo.x2 DO begin
          FOR Y := YPt to (YPt + 7) DO begin
            FOR X := XPt to (XPt + 7) DO begin
              PixelStatus := GetPixel(X,Y);
              IF PixelStatus = 0 THEN
                write(F,'.') {BrainMaker wants}
              ELSE { either a . or an X }
                write(F,'X');
            end;
            writeln(F);
          end;
          YPt := RowPt; XPt := XPt + 8;
        end;
        XPt := 0; RowPt := RowPt + 8; YPt := RowPt;
      end;
      Close(F);
    end;
    Dispose(SPort,done);
  end; { pcx_to_neural_net }

begin { main program body }
  pcx_to_neural_net; {Prep PCX file for BrainMaker}
  New(NNI, init);
  WITH NNI^ DO begin
    SwapVectors;
    exec('C:\COMMAND.COM','/C C:\BATCH\net'); {BM}
    SwapVectors;
    IF DosError <> 0 THEN
      writeln('Dos error # ',DosError);
    ELSE
      Get_weights; {Translate BrainMaker's output}
      Output_characters; {write a file of ASCII}
      Dispose(NNI, done); { Clean up }
    end;
  end. { main }           ♦ ♦ ♦

```

Problem: the default BGI font and the BrainMaker font both use 7 rows and lines of pixels to represent a number. But the .BGI's font is shifted 1 row and 1 column down and right of BrainMaker's. So I retrained the network using the .BGI default font. Results—great.

Conclusion—to create a full-blown Optical Character Recognizer, you'll need to focus on training the neural network to recognize different fonts, styles, and sizes. Labor intensive, of course, but doable. And an exercise, of course, for smart Micro C'ers.

"Amy," I said, "I figured it out and it works, but I'll have to get back to you."

"There's no more room in this Tidbits."

Thanks to Dave Schultz at California Scientific Software and Chris Howard at

Genus Programming for timely conversations while I was working on this project.

For more information—

Turbo Pascal Professional 5.5 \$250

Borland International
4585 Scotts Valley Dr.
Scotts Valley, CA 95066
(800) 345-2888

BrainMaker \$99.95

California Scientific Software
160 E. Montecito #E
Sierra Madre, CA 91024
(818) 355-1094

PCX Toolkit \$195

Genus
11315 Meadow Lake
Houston, TX 77077
(800) 227-0918

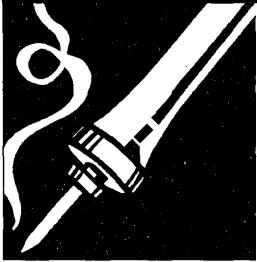
PaintShowPlus plus scanner \$339

Logitech
(800) 231-7717
(800) 552-8885 (in CA)

NeuralWorks \$1,495
Neuralware
Penn Center West-Bldg. IV
Suite 227
Pittsburgh, PA 15276

NeuroShell \$195
Ward Systems Group
228 West Patrick St.
Frederick, MD 21701
(301) 662-7950

♦ ♦ ♦



TECHTIPS

Floppy Data Recovery

If you have a floppy that's giving you read errors, try putting it in the refrigerator or freezer for a half hour or more. No joke. The magnetic domains are more active at lower temperatures. The party who turned me on to this used to teach physics and has used this technique many times to recover "lost" data from disks.

David E. Michener
7466 S.E. 112th Ave.
Portland, OR 97266-5033

Editor's note: Of course you could skip the disk and just keep the raw data in the freezer. I've found that it doesn't need blanching, just be sure to double wrap it so it doesn't get freezer burn. (I also have a recipe for binary freezer jam made from half-baked Apples (with the cores) and a little sugar.)

RAM Refresh Revisited

I have just tracked down a very strange and annoying bug relating to the DRAM refresh rate change you ran several months ago ("DMA Control on the PC," Issue #37). It seems only to happen with the AMI BIOS. As you increase the time between refreshes, the floppy disk motor start-up time also increases as some (large) multiple.

I have been setting the timer to hex 80 ("normal" is hex 12) and it takes about 15 seconds for the floppy to come up to speed. I finally got fed up enough to check out why, and it turns out to be DRAM refresh-time related.

As I'm an incurable speed-freak, I could not live without the extra 5% horsepower I got from changing the refresh. I wound up writing a short TSR that hangs on the INT 40H chain (where hard drives revector the floppy BIOS), restores the rate to its normal value

Frozen Floppies, Etc.

Figure 1—Speed Up CPU By Slowing Down DRAM Refresh

```
; This macro gets around a bug in 80286 that won't popf correctly
POPFF MACRO
    JMP    $+3
    IRET
    PUSH   CS
    DB     0E8H,0FBH,0FFH ;Call far $-2
    ENDM

REFRESH SEGMENT PARA PUBLIC 'CODE'
ASSUME CS:REFRESH, DS:REFRESH, ES:REFRESH, SS:NOTHING
ORG 100h           ; .COM format

REFR  PROC FAR
BEGIN:
    JMP    OVER          ; Jump over resident section of code
OLD_40 DD    0             ; Place to store old INT 40H vector
I_40:
    PUSHF
    PUSH   AX            ; Save the registers we're changing
    MOV    AL, 74H
    OUT   43H, AL          ; Set up to write LSB-MSB clock rate
    JMP    $+2            ; Delay for lousy IBM AT design
    MOV    AL, 12H
    OUT   41H, AL          ; Set refresh to every 12h clock ticks
    OUT   41H, AL          ; (15.3 u-secs ???)
    JMP    $+2            ; More delay
    XOR    AL, AL
    OUT   41H, AL          ; MSB of timer value to 0
    POP    AX            ; Recover our trashed registers
    POPFF
    PUSHF
    CALL   CS:[OLD_40]      ; (make this call look like an INT)
                                ; and call the original stuff
    PUSHF
    PUSH   AX            ; INT 40 done, restore fast DMA value
    PUSHF
    PUSH   AX            ; Save the registers we're changing
    MOV    AL, 74H
    OUT   43H, AL          ; Set up for LSB-MSB rate again
```

Micro Ads

A Micro Ad is the inexpensive way to reach over 22,000 technical folks like yourself. To place a Micro Ad, just print out your message (make it short and sweet) and mail it to Micro C. We'll typeset your ad (no charge) and run it in the next available issue. You can also send camera ready copy. Rates: \$99 for 1 time, \$267 for three times, \$474 for 6 times (a best buy at only \$79 per insertion). Full payment must accompany ad. Each ad space is 2 1/4 inches by 1 3/4 inches.



Briefly, a hypertext help system, context sensitive, up to 80% compression, requires less than 10K, includes editor for documents, and source for Pascal, C, and Basic.

(203)368 - 0632

INTERGAID, 955 Connecticut Ave. Bridgeport, CT 06607

WISDOM OF THE AGES

Electronic book puts over 1,000 of the world's greatest minds to work for you. They will give you tools to act, write and speak better; earn more.

Point to a subject and select. Receive a vivid parade of quotes, sayings and ideas. Read, print or send some to disk.

Subjects are organized into sections to give you the most benefit in the least time. Filters customize output; Dynamic mode simulates creativity.

It's never too early to get a head start. Introductory Offer: \$79.00 - 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. #81M
Oakland, CA 94611
1-800-767-6797
Fax: 415-444-6561

Reader Service Number 181

NEW COMPUTER BOOKS

up to 40% OFF LIST

SMALL, MINIMUM ORDER
SEND 3 STAMP SASE TO:

C. B. Co.

PO 1758
SOLANO BEACH, CA 92075

Reader Service Number 167

WHITNEY EDITOR \$49

Small and fast
Uses all available memory
Split-screen editing
Configurable keyboard
Regular expression search
One key compile
Features for writing documentation
Condensed/Outline display
Runs on IBM PC's, AT's, and PS/2's

USA shipping & handling \$3; Outside USA \$15
CA residents add sales tax

Whitney Software, Inc.
P.O. Box 4999, Walnut Creek, CA 94596 (415) 933-9019

Reader Service Number 164

CONVERT PCX FILES TO COLOR POSTSCRIPT!

PCX Printshop will convert all of your EGA and VGA PCX files to 16-color, 300 d.p.i. Color Postscript format.

Great for grabbing screens, touching them up, and printing them out at "coffee-table" resolution.

\$99

Argon Technology Group
131 Rindge Ave., Cambridge, MA 02140
(617) 492-7442

Reader Service Number 179

Technology for the Arts

PC MPU-401 Compatible
MIDI Interface Board \$119
Willow Publishers' VGA \$599
(video frame grabber/VGA)
New hi-res frame grabber \$495
Panasonic CCTV camera \$225
Special: Sanyo Color camera \$395
Free MIDI or Digitizer Catalogs
joe1 Sampson Engineering
P.O. Box 550363 (214) 328-2730
Dallas, TX 75355 BBS 328-6909

Reader Service Number 176

8031 µController Module \$39.95

Shipping: \$3.00 US/\$5.00 Canada

Ideal for prototypes, one of a kind devices or short production runs. Perfect building block for PC or Macintosh data acquisition interface or stand alone control projects. Assembled and tested board includes 8031 microprocessor, crystal, 8K of EPROM, 128 bytes of RAM, 2 byte-wide I/O ports and provisions for a MAX232 to provide industry compatible serial I/O. All ICs are socketed and I/O is via a 2X17 header. Size: 2.75" by 4.0". OEM discounts.

Cottage Resources
Suite 3-672C, 1405 Stevenson Drive
Springfield, Illinois 62703
(217) 529-7679

Reader Service Number 158

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

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



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

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.

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
iModes	Information Modes P.O. Drawer F Denton, TX 76202 817-387-3339	

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

Paul W. Carlson
602 North Avenue, #23
Burlington, Vermont 05401

before chaining on to the floppy BIOS, then sets it back to hex 80 before exiting.

I'm sending you the code for this TSR (see Figure 1) in the hopes that it will help anybody else having this problem. Note that this code is *dumb*; it assumes that anybody trying to increase CPU power must already have a hard drive.

I suppose it's possible that somewhere in Podunk, Iowa, there lives a true power user with dual floppies, but I've never met him. I am not very happy with this assumption, but it's what *I* needed. If the guy in Podunk needs this, he can rewrite it.

John Welch
1310 Kenneth Circle
Elgin, IL 60120

Back Up! (Let me repeat that...)

You say that backing up is hard to do? Let me assure you that it's harder, by far, to recover from a hard disk crash if you don't back up regularly. Yes, this is the voice of experience speaking. My old faithful 20 meg Miniscribe experienced a hardware failure.

Hindsight being what it is, backing files up daily is now a part of my routine, and I promise myself to keep it up as long as possible. The only problem is that I will have probably forgotten this trauma and let my habits get sloppy a month or so before the next failure.

Hard disks can fail in several modes. The most feared is a head crash. When that happens, the data is generally gone since the surface of the disk is physically damaged. In my case, I know the data is still in there, locked up like jewels in an impenetrable vault, and I don't have a combination to the door.

My failure was in the speed sensor; it wasn't a crash. So if any of you out there have a non-functioning 20 Megabyte MiniScribe (model 3425), I would be greatly interested in it for the parts.

I knew that the failure was not related to the heads because the Miniscribe told me. You see, the drive's LED activity light can flash error codes. The LED will flash four or five pulses of light, pause a few seconds, and repeat the code. A steady light indicates a 1, a flickering light indicates a 0. Figure 2 shows the error codes.

Although the codes seem to be the same from one Miniscribe drive to the

```

        OUT    43H, AL
        JMP    $+2          ; More lousy delay
        MOV    AL, 80H       ; A reasonable value for timeout
        OUT    41H, AL       ; That nets about 5% more CPU speed
        JMP    $+2          ; for free
        XOR    AL, AL       ; Set MSB to 0
        OUT    41H, AL

        POP    AX           ; Restore trashed registers
        POPFF
        RET    2            ; IRET without popping flags

; This ends resident code section. From here on, we set things up and
; parse command line. Since this gets done only once, we don't want to
; save this code, so it also contains the means to decide how much
; memory to save when going resident.

OVER: PUSH   DS
      CLI           ; Turn off INTs while revectoring INT
      XOR   AX,AX     ; INT vectors are at SEGMENT 0
      MOV   DS,AX     ; Set DS SEGMENT to zero

;Redirect the floppy
      MOV   AX, DS:[100H] ;Get old addr, and store it here
      MOV   WORD PTR CS:OLD_40,AX
      MOV   AX, DS:[102H]
      MOV   WORD PTR CS:OLD_40+2,AX

      MOV   WORD PTR DS:[100H],OFFSET I_40 ; Then replace it with
      MOV   WORD PTR DS:[102H],CS ; vector to floppy interceptor

      POP   DS           ; Put DS back where it belongs

      MOV   AL, 74H       ; Set DMA timer to slower value
      OUT   43H, AL       ; (less refresh)
      JMP   $+2
      MOV   AL, 80H
      OUT   41H, AL
      JMP   $+2
      XOR   AL, AL
      OUT   41H, AL

      MOV   AX,OFFSET OVER ; Get the addr of the end of INT code
      SHR   AX,1
      SHR   AX,1
      SHR   AX,1
      SHR   AX,1       ; Divide by 16 and add 1,
      INC   AX           ; to get how much memory to save
      MOV   DX,AX       ; Save the calculated amount of memory
      MOV   AH,31H       ; DOS func to terminate & stay resident
      STI
      INT   21H          ; Allow interrupts again
                           ; Do DOS call (never returns)

REFR  ENDP
REFRESH ENDS
END   BEGIN

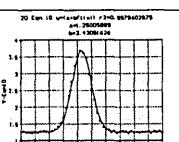
```

♦ ♦ ♦

more Micro Ads...

CURVE FITTING FOR PROGRAMMERS

- TABLECODE™ fits any X-Y data table to 211 different equations in a single step.
- Full graphical selection of most appropriate equation.
- Generates function code for selected equation and a twin-window calling program to test equation code.
- Reads Lotus, dBASE, ASCII, and binary files.
- Full working demo available for \$5.



Code Libraries Include: Turbo C & MSC 5.x, Turbo Pascal, BASIC, Turbo BASIC, QuickBASIC, Laney FORTRAN, JPL Modula2, dBASE IV, Clipper, dBASE III+.

AISN
Software

\$149
MC.VISA.Prc.appr.PO
P.O. Box 32277
Phoenix, AZ 85064
602-266-1925

Reader Service Number 170

C PROGRAMMERS

A Central Oregon technology company engaged in remote sensing and information management systems development has openings for:

1. Manager of software systems development/ Sr. Analyst.
2. Programmer/Analyst.

The ideal candidate will hold a BSCS and have a minimum of 5 years hands-on experience in C language, database integration, Unix, and workstation applications.

Comprehensive benefits and competitive salary offered.

Please send resumes to: Personnel
Recon Technologies, Inc.
P.O. Box 7497,
Bend, Oregon 97708
(503) 389-8149

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

BS DEGREES

... from fully-accredited Colleges. We help Computer Professionals avoid years of unnecessary class work. Get the job offers and recognition you deserve. Phone UDA for more information and our free booklet "Career Tactics and Strategies."

University Degree Advisory
800-765-7272

Reader Service Number 168

HIGH RESOLUTION TIMER TOOLBOX

PCHRT is the definitive answer to execution profiling and embedded timer applications. 30 functions manage 100 timers with microsecond accuracy. Self calibrating. Generates extensive and flexible timer reports. Profiles selected BIOS interrupts. Supports TC, TP 5.0, MSC 5.x. Libraries, examples, manual, and full source included. \$24.95 postpaid USA, elsewhere add \$4.00. VISA/MC.

Ryle Design

Reader Service Number 171

Figure 2—Miniscribe Error Messages

Pulses	HEX	Error reported
00000	00	Microprocessor RAM error
00001	01	Microprocessor ROM checksum error
00010	02	Interface Chip diagnostic error
00011	03	-WRITE FAULT will not reset
00100	04	Index pulse not detected or lost
00101	05	Unable to maintain spin speed (0.5%)
00110	06	Loss of +FINE TK during idle mode
00111	07	More than one seek retry
01000	08	Time out on +END DECL signal
01001	09	Time out on track crossing (-CYL PULSE)
01010	0A	Overshoot
01011	0B	Time out on +FINE TK
01100	0C	+TKO signal not detected
01101	0D	Comparator mismatch
01110	0E	Comparator mismatch
01111	0F	Unexpected Microprocessor interrupt
10000	10	Time out on TKO pat
10001	11	Time out on GB1 pat
10010	12	Time out on GB2 pat
10011	13	Seek range error
10100	14	Voltage unsafe with -WRTGATE inactive
10101	15	Voltage unsafe with -WRTGATE inactive
10110	16	Chip unsafe (-WRITE FAULT)
10111	17	Step pulses received with -WRTGATE active
11000	18	Time out on +END DECEL signal
11001	19	Time out on track crossing (-CYL PULSE)
11010	1A	Overshoot
11011	1B	Time out on +FINE TK
11100	1C	+TKO signal not detected
11101	1D	Comparator mismatch after rezero
11110	1E	Servo adjust failure - no closure
11111	1F	6301 Trap

♦ ♦ ♦

next, some drives pulse four times and some pulse five. If your drive pulses four times, ignore the leading zero in the above table. If there are only four pulses, the error cannot be higher than 01111.

Beverly Howard
Byte Aid!
205 Canyon Rim Drive
Austin, TX 78746-5016

Editor's note: John, Beverly, and David are each receiving three copies of this issue plus a genuine Micro C author's tee shirt (available only to Micro C authors). If you have any juicy little technical tips, send them to: Juicy Tips, Micro Cornucopia, P.O. Box 223, Bend, Oregon 97709.

♦ ♦ ♦

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

Complete Your Education . . .**Fill Out Your Collection of Micro C today!**

ISSUE #1 (9/81) Power Supply 1/2 PFM.PRN 16 pages	ISSUE #16 (2/84) Xerox 820 Column Restarts BBI Double Density BBII 5"8" Interface Fix Kaypro ZCPR Patch Adding Joystick To Color Graphics Recovering Text From Memory 52 pages	ISSUE #27 (12/85) SOLD OUT	ISSUE #41 (5/88) Artificial Intelligence 3-D Graphics Neural Networks Logic Of Programming Languages Applying Information Theory 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 Practical Fractals 96 pages
ISSUE #2 (10/81) Parallel Print Driver Drive Motor Control 16 pages	ISSUE #17 (4/84) Voice Synthesizer Kaypro Morse Code Interface 68000-Based System Review Inside CP/M 86 56 pages	ISSUE #28 (2/86) Rescuing Lost Text From Memory Introduction To Modula-2 Inside The PC 104 pages	ISSUE #42 (7/88) Maintaining PCs Keeping Your Hard Drives Running Troubleshooting PCs XT Theory of Operation Simulating A Bus Ray Tracing 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 #3 (12/81) 4 MHz Mods Configuring Modem 7 Reverse Video Cursor FORTHwords Begins 16 pages	ISSUE #18 (6/84) Kaypro EPROM Programmer I/O Byte: A Primer Kaypro Joystick Serial To Parallel Interface Business COBOL 60 pages	ISSUE #30 (6/86) PROLOG On The PC Expert Systems Logic Programming Building Your Own Logic Analyzer 256K RAM For Your 83 Kaypro PC-DOS For Non-Clones 104 pages	ISSUE #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 #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 The Parallel Port The LIMBO Project—Part Two PCX Compatibility A 68000-Based Multitasking Kernel The Very Early Days of Computing 96 pages
ISSUE #4 (2/82) Keyboard Translation More 4 MHz Mods Modems, Lync, and S10s Undelete CP/M ERASE 20 pages	ISSUE #19 (8/84) Adding Winchester To BBII 6 MHz On The BBII Bulletin Boards Track Buffering On Slicer 4 MHz For The 820-1 64 pages	ISSUE #31 (8/86) RAM Resident PC Speedup Practical Programming In Modula-2 Unblinking The PC's Blinkin' Cursor Game Theory In PROLOG and C 104 pages	ISSUE #44 (11/88) Object-Oriented Programming A Taste of Smalltalk Actor Thinking Objectively Building MicroCad Peripheral Technology—PT68K-2 Hercules Graphics Printer Dump 96 pages	ISSUE #49 (9/89) I/O, I/O... Build A Computer The Easy Way PostScriptals Prog. Logic Controllers Driving Stepper Motors Writing TSR Programs Low Cost I/O For The PC Interfacing 16-Bit Devices 96 pages
ISSUE #6 (6/82) BBI EPROM Programmer Customize Your Chars Double Density Update 24 pages	ISSUE #20 (10/84) HSC 68000 Co-Processor DynDisk For The BBII Serial Printer On BBI Sans S10 Cheap & Dirty Talker For Kaypro Extended 8" Single Density 72 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) Parallel Processing Sound Generator For the STD Bus Reviews Of 256K RAM Expansion 88 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 #7 (8/82) 6 Reviews Of C Adding 6K Of RAM On Your Own Begins 24 pages	ISSUE #21 (12/84) Analog To Digital Interface Installing Turbo Pascal Low Intensity BBI Video Turbo Pascal, The Early Days 80 pages	ISSUE #34 (2/87) 3-D Graphics Xerox 820-II To A Kaypro-8 Sound Generator For the STD Bus Reviews Of 256K RAM Expansion 88 pages	ISSUE #35 (4/87) PC Graphics Drawing The Mandelbrot And Julia Sets Desktop Graphics Designing A PC Workstation Board Around the TMS-34010 96 pages	ISSUE #46 (3/89) 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
ISSUE #8 (10/82) SOLD OUT	ISSUE #22 (2/85) Kaypro 820-II To A Kaypro-8 Sound Generator For the STD Bus Reviews Of 256K RAM Expansion 88 pages	ISSUE #36 (6/87) Parallel Processing Laser Printers, Typesetters And Page Definition Languages Build A Graphics Scanner For \$6, Part 2 Writing A Resident Program Extractor In C 96 pages	ISSUE #37 (9/87) PC Graphics Drawing The Mandelbrot And Julia Sets Desktop Graphics Designing A PC Workstation Board Around the TMS-34010 96 pages	♦ ♦ ♦
ISSUE #9 (12/82) BBI EPROM Program Relocating Your CP/M Serial Print Driver Big Board I Fixes 32 pages	ISSUE #23 (4/85) Automatic Disk Relogging Interrupt Drive Serial Printer Low Cost EPROM Eraser Smart Video Controller Review: MicroSphere RAM Disk 86 pages	ISSUE #38 (11/87) Parallel Processing Laser Printers, Typesetters And Page Definition Languages Build A Graphics Scanner For \$6, Part 2 Writing A Resident Program Extractor In C 96 pages	ISSUE #39 (1/88) PC Graphics Drawing The Mandelbrot And Julia Sets Desktop Graphics Designing A PC Workstation Board Around the TMS-34010 96 pages	
ISSUE #10 (2/83) ISSUE #11 (4/83) SOLD OUT	ISSUE #24 (6/85) C'ing Into Turbo Pascal 8" Drives On The Kaypro 68000 Versus 80x86 Soldering: The First Steps 88 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 #12 (6/83) Bringing Up BBII Double Sided Drives for BBI Packet Radio 5 MHz for Kaypro 40 pages	ISSUE #25 (8/85) Why I Wrote A Debugger The 32-Bit Super Chips Programming The 32032 Modula II RS-232C: The Interface 104 pages	ISSUE #42 (7/88) Maintaining PCs Keeping Your Hard Drives Running Troubleshooting PCs XT Theory of Operation Simulating A Bus Ray Tracing 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 #13 (8/83) CP/M Disk Directory More 256K for BBII Mini Front Panel Cheap Fast Modem BBI Printer Interface Kaypro Reverse Video Mod 44 pages	ISSUE #26 (10/85) Inside ZCPR3 Two Megabytes On DS-32 SOG IV The Future Of Computing Graphics In Turbo Pascal 104 pages	ISSUE #44 (11/88) Object-Oriented Programming A Taste of Smalltalk Actor Thinking Objectively Building MicroCad Peripheral Technology—PT68K-2 Hercules Graphics Printer Dump 96 pages	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 #14 (10/83) BBII Installation The Perfect Terminal BBI Video Size Video Jitter Fix Kaypro Color Graphics Review 48 pages	ISSUE #27 (12/85) SOLD OUT	ISSUE #46 (3/89) 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	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 #15 (12/83) Screen Dump Listing Fixing Serial Ports Playing Adventure Upgrading Kaypro II To 4 Upgrading Kaypro 4 To 8 48 pages				

To Order:

Phone: 1-800-888-8087
Mail: PO Box 223
 Bend, Oregon 97709

United States,
 Issues #1-34 \$3.00 each ppd.
 Issues #35-current \$3.95 each ppd.

Canada & Mexico
 All issues \$5.00 each ppd.



A MICRO C ADVERTISER'S INDEX

ISSUE 51

170 AISN Software	93
72 Acquired Intelligence	71
160 Annabooks	55
179 Argon Technology Group	91
4 Austin Codeworks	35
147 Berry Computer	5
31 CC Software	77
** Carlson, Paul	91
15 Cascade Electronics	25
167 Computer Book Outlet	91
7 Compview Products, Inc.	7
158 Cottage Resources	91
143 Covox	6
174 Daytron	55
16 Dreamtech	Inside Back
10 Emerald Microware	33
135 Epoch Data	52
93 Erac Co	63
** Eckel, Bruce	95
112 Garrison	73
** Genus Microprogramming	19
180 IDEC	15
149 Information Modes	21, 91
22 Integrant	45
154 JRT Systems	69
175 Kilgore Software	67
181 MCR	91
42 McTEK	75
** Micro Cornucopia	71, 93
37 Microprocessors Unlimited	77
2 Microsphere, Inc.	Inside Front
165 Mt. St. Helens Software	13
** NTERGAID	91
110 Nu-Mega Tech	2
161 Opal Fire Software	73
3 PC Tech	Back Cover, 91
119 Peripheral Technology	76
139 Quantasm Corp	15
** Recon Technology	93
129 Research Group	61
** Revolution 2	49
** RJSwantek Inc	37
171 Ryle Design	93
176 Sampson Engineering	91
127 SemWare	53
162 Semi-Disk Systems	36
177 Silicon Alley	47
** Sintar Software	87
40 Star-K Software Systems	44
173 Sutrasoft	21
182 Symmetric Research	91
183 Tools and Techniques	87
178 Traxel Labs	12
168 University Degree Advisory	93
62 V Communications	57
124 Wenham Software Co	91
169 Western Wares	72
164 Whitney Software	91
39 Xenosoft	93
70 Zortech	1

** Contact Advertiser Directly.

When you write for information, please tell these folks you read about their products in Micro Cornucopia.

Now Available From Micro C

Computer Interfacing with Pascal & C by Bruce Eckel

- Use your PC parallel port for digital input and output
- Build an Adapter Card for your PC
- Design and build electronic circuits

"With wit and superb technical figures, Bruce captures the essence of making electrons out of bits and vice versa."

Jeff Dunteman, *Dr. Dobbs*

Order From:
Micro Cornucopia
PO Box 223
Bend, OR 97709
1-800-888-8087

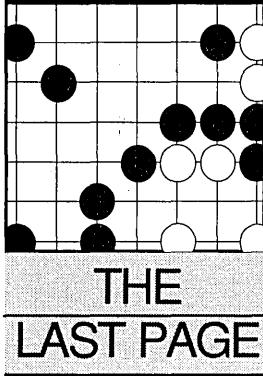
Only \$30 ppd.
Add \$10 for foreign orders
Includes Book & Disk



Issue #52

Object Oriented Programming

- C++ Pointers, from Walter Bright
- Inside C++ 2.0
- Building an AT Keyboard Interface
- TTL Families Explained
- Filling Memory Holes on Your XT
- Neural Networks, Part 2



By Gary Entsminger

P.O. Box 2091
Davis, CA 95617

Brains, Neural Networks, & Expert Systems

Gary takes a top down approach to bottom up.

How do we learn, recognize, and remember? In other words, *how do we think?* And to what degree can a computer model us? Duplicate our thinking? (*Editor's note: Assuming a computer would want to model us.*)

Much thinking, paper, and many computer bits have been used to explore these and similar questions. Computer scientists, neurobiologists, psychologists, philosophers, and just plain folk delight in concocting visions and experiments, in hopes of edging a bit closer to something plausible.

The AI community, for example, broadly splits along two paradigms for explaining thinking: Thinking => the way the brain does it; Thinking => the results the brain gets. These views represent the strong and weak schools of AI.

The weak school at its most pliant (according to philosopher Keith Gunderson) claims that computers can perform tasks that previously required intelligence, without using any intelligence at all.

An expert system belongs in this paradigm since it typically classifies new information based on rules extracted from a human expert. The expert was (presumably) intelligent. The expert system will seek answers, give advice, or classify by looking things up. Looking things up isn't all that intelligent.

Rule-based systems also belong to the domain of the top-downers in the AI community. Alan Turing, Herbert Simon, Allen Newell, Edward Feigenbaum, and Roger Schank are all top-downers, and their thinking about thinking has dominated AI research for 30 odd years.

Rule-based systems, scripts, and frames are the tools of the AI top-down trade. The "intelligence" of any of these systems is generally static. No dynamic learning system based on rules has ever been successful.

In a fascinating study of the "big issues" in science, called *Paradigm's Lost*, John Casti sums up the top-down position—

Put crudely, the top-down thesis is that human thought processes take place as a result of rule-based symbol processing in the brain.... Top-downers blithely forget about brain states altogether, and just assign machine states to cognitive states, much in the same manner as we assign ASCII codes to alphanumeric symbols.

A set of rules (usually termed a semantic network or conceptual dependency graph) telling how these machine states can combine with each other is then postulated, and the resulting machine states are decoded to give an interpretation.... This, in a nutshell, is the strategy of the entire top-down approach to AI.

The top-down approach is software oriented, ignoring the hardware (our brains) almost entirely.

Bottoms Up

The bottom-uppers take the opposite approach. In their view we'll learn more about thinking by working from the level of primitive processors (the neurons in our brains) up. Strong AI.

Neural networks and the work of bottom-uppers—Douglas Hofstadter, Geoffrey Hinton, Douglas Lenat, and (strangely enough, considering his early criticism of perceptrons—neural network-like systems) Marvin Minsky—are beginning to shed a bit of light on the thinking question and show promise.

Neural networks aren't technically that complicated (see Russ and Roy's article in this issue) but are, I think,

philosophically and technically appealing.

The net uses patterns of bits instead of individual bits to process information. Since a network looks for patterns rather than at the status of individual bits, no single bit is crucial. The network can deal with minor variations in input and still produce the right output.

I'm inclined to think that this more nearly mirrors our thought processes. Humans are associators, often guessers—we take informal statistical samples and make decisions based on a best-fit scenario. One memory leads to another. We often forget details, but still recall the overall picture.

The bottom-uppers' view is also consistent with most physicists—we can't know individual particles precisely (their momentum and location), but we can know quite a bit about the behavior of a group of particles.

Bottom-uppers haven't answered the big-think question, and probably won't, but they're tilting the AI community in an intriguing direction.

For now of course, who knows how we do it. Think, I mean. According to Aristotle—

It often happens that a man cannot recall at the moment, but can search for what he wants and find it.... For this reason some use places for the purposes of recollecting. The reason for this is that men pass rapidly from one step to the next: for instance from milk to white, from white to air, from air to damp; after which one recollects autumn, supposing that one is trying to recollect that season.

References

Casti, John L., *Paradigm's Lost*, 1989, William Morrow & Co.

And thanks to Al Goldberg for stimulating this discussion.



Lowest Prices
Top Quality
Great Selection
Excellent Service

The Computer Store

America's finest in computer mail-order is having their *biggest* sale ever...

The Dream-88®

30 Megabyte IBM Compatible

12 MHz CPU with 640K RAM installed
 30 Megabyte hard disk drive (3.5" 40ms average access)
 128K Expanded memory installed (LIM 4.0)
 1.2 Megabyte or 360K 5.25" floppy drive
 Floppy controller
 (will control up to 1.44MB floppy drives)
 High resolution amber monitor with tilt/swivel base
 Hercules compatible graphics
 Professional enhanced keyboard
 1 year warranty

Reg. \$995

Sale \$ 795



All computer systems can be customized
to your exact specifications

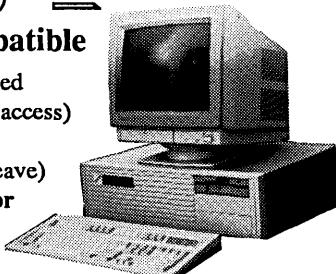
The Dream-286®

Our 40 Megabyte IBM AT Compatible

12 MHz CPU with 1 Megabyte RAM installed
 40 Megabyte hard disk drive (28ms average access)
 1.2 Megabyte 5.25" floppy drive
 Hard & floppy disk controller card (1:1 interleave)
 14" High resolution flat screen amber monitor
 Hercules compatible graphics
 Professional enhanced keyboard
 1 year warranty

Reg. \$1495

Sale \$ 1195



Upgrade any of these computers to color (monitor and adapter)

CGA +\$150

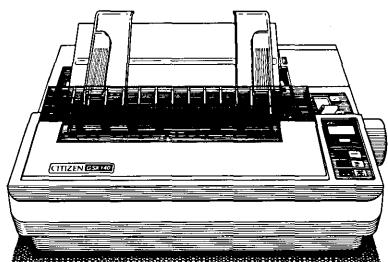
EGA +\$325

VGA +450

Citizen GSX-140

24-wire 192 CPS Printer
 Color on command®
 Advanced Paper Handling
 Command Vue® LCD console
 eliminates dip switches
 Advanced paper handling
 including paper parking
 Push/pull tractor with bottom,
 top, and rear feed
 2 year warranty

Reg. \$595



Sale \$ 345

If you need it, we have it... for less. Give us a call...

6' Parallel Printer Cable

Molded & shielded with thumb screws

\$ 3

A/B switch Box

For parallel printers

\$ 19

Joystick

With auto centering & rapid fire

\$ 15

Game & Clock Card

Each

\$ 12

Logitech C-7 Mouse

Serial Mouse with software

\$ 65

Nashua DSHD Disks

Box of 10 high density disks in
plastic carrying case

\$ 10

VGA Card

\$ 185

Capable of 1024 X 800 with 256 colors

The Dream-386®

Our 40 Megabyte 386 Speed Demon

16 MHz CPU with 1 Megabyte RAM installed
 40 Megabyte hard disk drive (28ms average access)
 1.2 Megabyte 5.25" floppy drive
 Hard & floppy disk controller card (1:1 interleave)
 14" High resolution flat screen amber monitor
 Hercules compatible graphics
 Professional enhanced keyboard
 1 year warranty

Reg. \$1995

Sale \$ 1495

Above with 20MHz CPU

Reg. \$2295

Sale \$ 1795

We carry a complete selection of...

Hard/Floppy Drives

Memory upgrade cards

With name brands like...

Apple Citizen

RAM chips

IBM Seagate

I/O cards

Everex Maxtor

Modems

Micropolis Sony

Monitors

Miniscribe Samsung

Printers

Borland Logitech

Diskettes

Microsoft Xerox

Digitizers

SCO Aldus

Cables

Novell Hayes

Ribbons

NEC Curtis

Software

Toshiba Fujitsu

...and more

...and more

DreamTech (408) 996-2373

'In the heart of Silicon Valley'

5175 Moorpark Avenue San Jose, CA 95129



Prices are subject to change without notice. All sales are FOB San Jose, and are subject to prior sale. A 3% surcharge applies for all MC and VISA transactions. California residents add 7% sales tax. The following trademarks are recognized: IBM PC/XT, AT to IBM Corporation; Hercules to Hercules Computer Technology, Inc. Other product names are trademarks of their manufacturers. This ad is a copyrighted publication of DreamTech with all rights reserved. Distributed Micro Cornucopia, January, 1990.

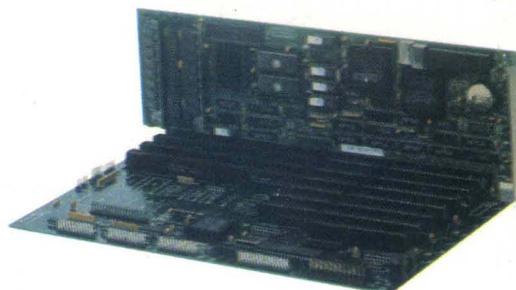
VERY HIGH PERFORMANCE

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.



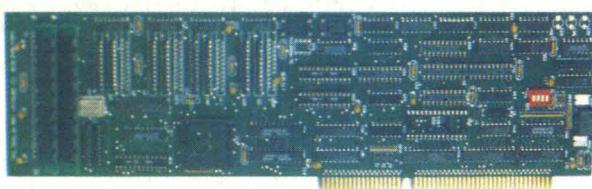
PC Tech X24 and ASMB

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 TMS34010 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.

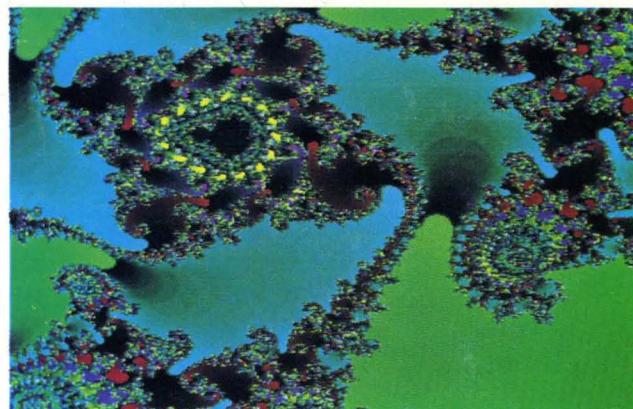
About PC Tech

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