

MICRO CORNUCOPIA

3-D Graphics / AI Theory

Three-Dimensional Graphics, Part 1 page 8

Earl Hinrichs uses high performance graphics IC to create and display depth.

Neural Networks page 16

Modeling human reasoning is the first step in creating a useful robot.

The Logic Of Programming Languages page 22

Proving that a language is logically valid beats testing it for 10 years.

Applying Information Theory page 42

Calculating the maximum theoretical data compression and then applying it.

Plus:

Updating The C Reviews page 48

RS-232 Interfacing page 30

Button's Great Sharew page 58

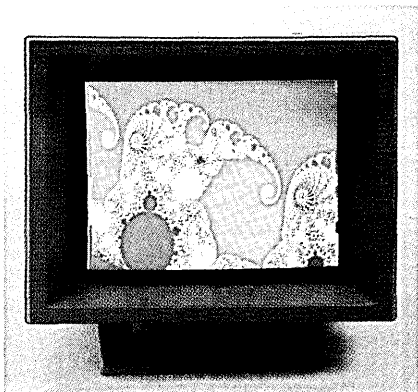
Marketing your prof

And Much, Mu



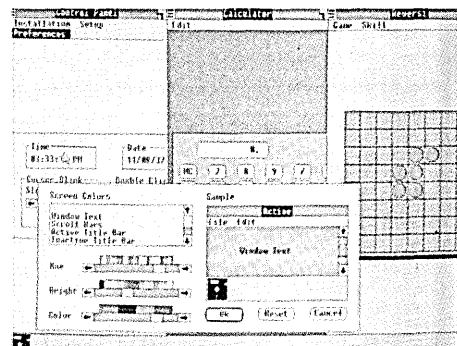
VERY HIGH RESOLUTION

The PC Tech COLOR and MONOCHROME video processor boards employ the TMS 34010 high performance graphics co-processor to insure the best possible video performance at reasonable prices.



Color 34010 Video Processor:

- Featured on the cover of Micro Cornucopia.
- From 800 x 512 through 1024 x 800 resolution (depending on monitor and configuration).
- 8 Bits per pixel for 256 simultaneous colors
- Hardware support for CGA/MDA emulation.
- PC, XT, and AT compatible



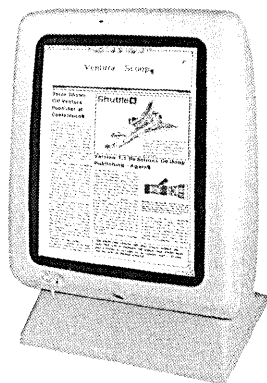
The PC Tech Color 34010 video processor is a superior 34010 native code and DGIS development tool. We support up to 4 megabytes of program (non-display) 34010 RAM as well as up to 768K bytes of display RAM. *Compare our architecture and prices to any other intelligent graphics board. Then choose the PC Tech Color 34010 Video Processor for your development engine and your production requirements as well.*

Color 34010 Video Processor \$1,195.00

Price includes 512K display RAM, 1024K program RAM, and utility software. Monitor not included.

Also available: DGIS, 34010 C compiler, assembler, 34010 fractal software, additional display and program memory, and various monitor options.

PC Tech Monochrome 34010 Video Processor and Monitor



- 736 x 1024 resolution (other options available)
- 2 bits per pixel for 4 hardware gray shades
- Hardware support for CGA/MDA/Hercules emulation
- PC, XT, and AT compatible
- Full page 66 line text editing with many popular editors
- Excellent windows 2.0 application development system

The graphics and bit manipulation capabilities of the TMS 34010 make the PC Tech Monochrome 34010 Video Processor 66 line full page text and graphics display faster than many 25 line systems. The video processor is available separately or with the high resolution white phosphor monitor shown above.

Monochrome 34010 Video Sub-System \$1,295.00

Price includes Monochrome Video Processor and monitor pictured above.

Also available: DGIS, TI 34010 C compiler, TI assembler.

Monochrome 34010 Video Processor also available separately.

Designed, Sold and Serviced By:



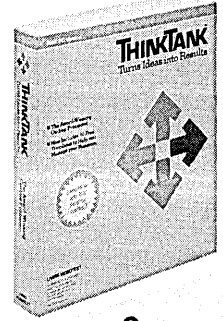
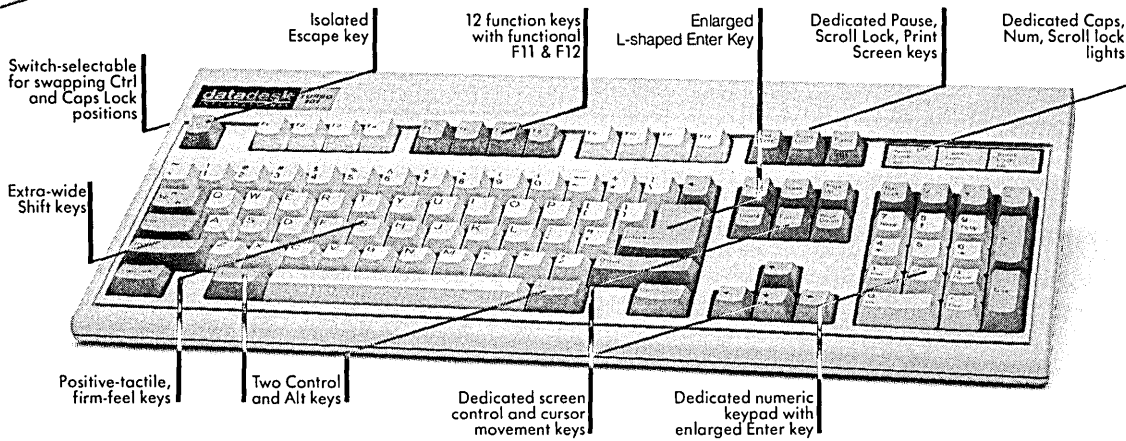
SOG SPECIAL! PC Tech will offer a 10% SOG DISCOUNT on all 34010 based video products ordered within 1 week of SOG! Special pricing effective July 7-21, 1988

904 N. 6th St.
Lake City, MN 55041
(612) 345-4555
(612) 345-5514 (FAX)

TURBO 101 ENHANCED KEYBOARD
BEST OF 1987
 PC MAGAZINE

It's A No-Thinker!

DataDesk's New Turbo-101 Enhanced Keyboard Bundle!



**FREE! \$200
 THINKTANK™
 SOFTWARE**

You don't have to think twice. Get both — the best-selling Turbo-101™ Enhanced Keyboard and ThinkTank™, the award-winning outline processor. For an unthinkable \$149.95!

Now Both You and Your PC can be State-of-the-Art

The Turbo-101 Enhanced Keyboard gives you all of the features of IBM's newest keyboard standard (see above photo). It's ideal for word processing with its traditional "Selectric" typewriter layout featuring enlarged Shift, Enter and Control keys. Spreadsheet entry couldn't be easier because we give you both a separate Cursor and Numeric keypad eliminating the need to constantly toggle Nums Lock. And the Turbo-101's exclusive tactile-feedback keys let you type faster with fewer mistakes than ever before.

ThinkTank is the world's most powerful outliner and idea processor. It will introduce you step-by-step to a fascinating new way of organizing and managing your ideas, your time, your business, and even your thinking. And to get you up and running quickly and easily we even include 30 predefined ThinkTank Productivity Templates. ThinkTank is considered by many to be one of the most important productivity enhancers on a micro-computer. It turns thoughts into reality and ideas into results!

Instantly Upgrade Your Productivity

The Turbo-101 and ThinkTank are both designed to instantly streamline and improve your computing performance regardless of which model PC, XT, AT, Tandy or compatible you use. With our Turbo-101—ThinkTank bundle, you'll feel the difference from the tip of your fingers to the center of your brain.

**TO ORDER BY PHONE CALL
 (800) 826-5398
 IN CA CALL (800) 592-9602**

You Don't Have To Be A Brain Surgeon To Understand This Special Offer.

ThinkTank is already a best seller at \$195. So it's easy to figure out that getting both the critically-acclaimed Turbo-101 keyboard and the ThinkTank software for the price of the keyboard alone—\$149.95—is a deal that turbocharges your PC without flattening your wallet. And that's not all, our USA-made Turbo-101 keyboard is so well built it carries a full 2-year warranty—which is probably a lot longer than your PC's warranty! But just in case you are not completely satisfied, you can always change your mind, with our no-questions-asked 30-day money-back guarantee.

Here's What the Experts Think...

“ I really prefer the feel of the DataDesk Turbo-101. The keys have tactile feedback. No mush at all.

Jerry Pournelle
Byte Magazine, August 1987 ”

“ Don't plopp down your simoleons for a Keytronic or others... for ingenuity of design and sheer dollar-value those from DataDesk can't be beat.

Curt Suplee
Washington Post, May 1987 ”

Models also available for the Tandy 1000, AT&T, PCjr and all Macintosh computers.
 *Limited offer with purchase of Turbo-101 enhanced keyboard. All DataDesk International products are registered trademarks or trademarks of DataDesk International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders. Copyright 1987 DataDesk International.

“ Great typing touch... there's only one DataDesk Turbo-101—the first keyboard to challenge IBM products seriously.

Jim Seymour
PC Magazine, January 1988 ”

“ The Turbo-101 can satisfy all... it's a rock-solid product that does double-duty and then some.

Robert Luhn
PC World, October 1987 ”

datadesk
 INTERNATIONAL

**7651 HASKELL AVENUE
 VAN NUYS, CA 91406
 (818) 780-1673**

**BOTH
 TURBO-101 ENHANCED
 KEYBOARD AND THINKTANK
 SOFTWARE FOR ONLY:**

\$149.95
LIMITED OFFER

Please add \$10 shipping and handling. CA residents must also include \$9.75 sales tax for each keyboard.

Name _____
 Company _____
 Street Address _____
 City _____ State _____ Zip _____
 Phone _____
 Computer Type _____
 Credit Card No. _____
 Exp. _____

IT'S WHAT'S UNDER THE HOOD THAT COUNTS!

XT KIT W/ 2 Floppy Drives.

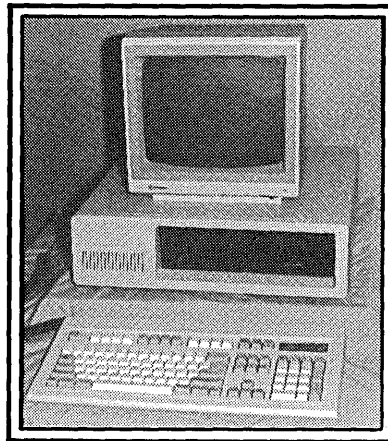
Includes: 0K RAM, Serial, parallel and game ports, clock/calendar, AT-Style keyboard, cabinet, power supply, mono graphics card and amber or green monitor. Keyboard switchable turbo.

8 mhz with lock, LED, Reset & Turboswitch 579.00
10mhz with lock, LED, Reset & Turboswitch 599.00

XT KIT W/20MB Hard Drive.

Includes: 0K RAM, Serial, parallel and game ports, clock/calendar, AT-Style keyboard, cabinet, power supply, mono graphics card and amber or green monitor. Keyboard switchable turbo.

8 mhz with lock, LED, Reset & Turboswitch 879.00*
10mhz with lock, LED, Reset & Turboswitch 899.00*
*(For 30MB Miniscribe add \$50.00)



— Pictured keyboard is 5339 —

NEW PRICES!

80386 KIT —

Includes: 8/16 mhz, 1MB RAM, 1 360K floppy drive, 1-1.2 MB FD, 1 40MB HD, Award bios, switchable keyboard, monochrome monitor, monographics. Serial/parallel ports, case, power supply, game port, clock/calendar. Main board made in U.S.A. **2675.00**

80286 - AT KIT

Includes: 0K RAM, 1.2 MB FD, 1 360K floppy drive and 40 MB Seagate St 251 hard drive, 6/10mhz, serial, parallel and game ports, clock/calendar, AT-style keyboard, cabinet, power supply, monographics card, amber or green monitor, keyboard switchable turbo. **1595.00**

CASES & POWER SUPPLY

150 Watt Power Supply (XT) 50.00
200 Watt Power Supply (AT) 80.00
XT Slide Case 34.00
XT Flip Top or XT Slide with Lock & LED 38.00
AT with Lock & LED 65.00

MONITORS

EGA/CGA (Auto Switch) 452.00
VGA/EGA/CGA Color 600.00
CGA Color 295.00
Amber 12" TTL 89.00
Green 12" TTL 89.00

VIDEO CARDS

Color/Graphics/Parallel 52.00
256K EGA Graphics 125.00
Mono/Graphics/Parallel 49.00
ATI Graphics Solution—
Mono, Herc. Color Emulation on Mono CGA (List 299) 125.00
ATI Wonder Auto Switch Mono, Herc Any monitor, Any software, Auto conversion
CGA, EGA, VGA (List 499) 299.00
EGA, CGA, PGA (640x480) 185.00

EXPANSION CARDS

Clock Card 25.00
Dual Floppy Disk Controller 25.00
Joystick 25.00
Gravis Analog Joystick 49.95
Game Port 19.00
Multi-Function, 1 ser/par/clock/game/2 floppy 61.00
Parallel (printer) 19.00
Serial Port (RS232) 1 port 29.00
640K RAM (0K installed) 35.00
XT/AT RS232 (4 port/2 installed) 59.00

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

KIT OPTIONS

*MS DOS 3.21 or 3.3 w/
GW Basic 95.00
*5339 Keyboard Sub 24.00
*Color Options:
(Includes video card & monitor)
CGA Color 200.00
CGA/EGA Color 410.00
CGA/EGA/VGA Color 590.00
ASSEMBLY AND TESTING
XT Systems 60.00
AT/80386 Systems 80.00

MOTHERBOARDS

XT/Turbo 4.77/10mhz 109.00
AT 6/10 mhz (4 layer) Choice of Phoenix or DTK Bios 350.00
XT/Turbo 4.77/8 mhz (2 layer) 99.00
80386 8/16 mhz/Award Bios & 1MB RAM, made in U.S.A. 1595.00
For XT/AT memory \$Call

FLOPPY DISK DRIVES

Fujitsu 360K 89.00
Toshiba 360K 99.00
Teac 1.2 MB 125.00
Toshiba 3½" Drive Kit 720K 125.00
Toshiba 3½" Drive 1.44mb 145.00

KEYBOARDS

5339 Professional XT-AT w/12 function key 69.00
5060 Keyboard AT Style 55.00
KB101 Keytronic 67.00

Free Instructions with Each System

DEPEND ON MICROSPHERE

We've been building computer equipment for 4½ years. The components and products we sell are chosen specifically because they have been proven in our own use and testing. We guarantee our cards will be compatible when purchased all together.

HARD DRIVES & CONTROLLERS

AT 40 MB Seagate #251-1 495.00
AT Hard Drive & floppy controller (WD) 140.00
20 MB Miniscribe HD with controller 349.00
30 MB Miniscribe HD with controller 399.00

SOFTWARE

The Twin Spreadsheet 49.00
Leading Edge Word Processor 49.00
Ventura Desktop Publisher by Xerox 525.00
Turbo C by Borland 89.00
Turbo Pascal V4 89.00

ACCESSORIES

1200 Baud Modem — Internal (Leading Edge Model L) Hayes compatible 99.00
2400 Baud Modem — Internal (Leading Edge Model L) Hayes compatible 219.00
1200 Baud Modem — External Hayes compatible 119.00
V20-8mhz 14.00
Memory Chips (call for prices)

BUILDING YOUR OWN CLONE

****FREE BOOKLET****

*90-day warranty/30-day money back (subject to restrictions)



MicroSphere, Inc.
P.O. Box 1221
Bend, Oregon 97709
(503) 388-1194

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



Reader Service Number 2

MICRO CORNUCOPIA

MAY/JUNE 1988 - ISSUE NO. 41

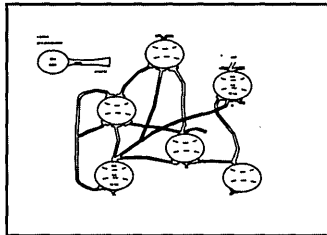
FEATURES

8 Earl Hinrichs Graphics In 3-D (Part 1)

There's a lot more to three dimensions than gluing a balsa-wood model to the face of your CRT. Earl tackles the nits and grits (mathematics and chip manipulation) required to give depth to your flat screen.

16 Diane Ingraham, Gurmail Kandola, Mark Pillon Neural Networks

Making computers think the way you think isn't as easy as you might think.



22 Paul Voda The Logic Of Programming Languages

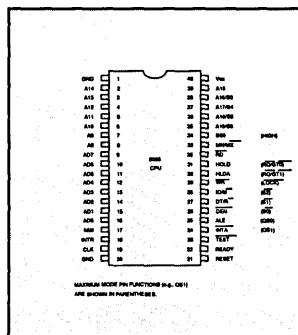
If you can prove mathematically that a language works and prove that the program is valid, then you've proven the results. Unfortunately many languages are unprovable.

30 Bruce Eckel The Mysteries Of RS-232

If you need connections in the computer world, let Bruce put in a plug for you (so all your handshakes will be successful).

36 Larry Fogg Intel's 8088

Larry exposes PC brains in this, his last look at the monster chips in the MS-DOS machines.



42 Ramachandran Bharath How Much Information Does A Message Contain?

Know anything about information theory? It's that stuffy theoretical stuff they teach in graduate school. Right? Well, it's not so stuffy and it's not just for students anymore.

COLUMNS

48 C'ing Clearly

Scott Ladd updates his review of C packages and then updates us on memory model sizes.

54 86 World

Laine returns to the U.S. and celebrates by building a keyboard translator.

58 ShareWare

62 On Your Own

66 Pascal Column

71 Culture Corner

85 Technical Tips

CP/M CORNER

72 CP/M Notes

73 Kaypro Column

FUTURE TENSE

82 Tidbits

96 Last Page

Cover illustration by Paul Leatherwood

Editor & Publisher
David J. Thompson

Associate Editors
Gary Entsminger
Cary Gatton

Technical Department
Larry Fogg

Director of Advertising
Laura Logan

Accounting
Sandra Thompson

Order Department
Tammy Westfall

Graphic Design
Carol Steffy

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

SUBSCRIPTION RATES:

1 yr. (6 issues).....\$18.00
2 yr. (12 issues).....\$34.00
3 yr. (18 issues).....\$48.00
1 yr. (Canada & Mexico).....\$26.00
1 yr. (Other foreign)\$36.00
Make all orders payable in U.S. funds on a U.S. bank, please.

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

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

CUSTOMER SERVICE: for orders & subscription problems call 503-382-5060, 9 am to 5 pm, Pacific time, M - F.

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

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

Copyright 1988 by Micro Cornucopia Inc.
All rights reserved
ISSN 0747-587X



By David Thompson

Museum Pieces

There's something magic about climbing into a Stinson, a fabric and frame tail-dragger that's pushing the ragged side of 40. And there's even more magic when I bring to life its cranky old Franklin engine, six giant cylinders, 165 horses, sounding every bit like 600.

Forty years ago Stinsons were fast, but since then the rest of the world has more than caught up. They were once the most popular plane in the air. Now they're rare enough to be a curiosity and wherever one goes, a group gathers.

"A friend had one of these. Once." _

"I built a model Stinson when I was in the fifth grade. Or was it the sixth grade? I've always dreamed of flying in one."

"Do you still fly it?"

Occasionally a burled hand gently presses the varnished fabric. The hand remembers another Stinson, perhaps.

The plane is quiet too.

"It used to have a wooden prop," I tell them. "But a rough landing shortened it a bit."

That happened before the Stinson and I met, and I've often wondered how it flew with that hand-rubbed, wood-grained prop. Better, I think.

You can tell the pilots from the non-pilots. Pilots know that props suffer occasionally and let it pass. Non-pilots glance at the gray-metal replacement and then, casually, move away.

They needn't worry, I'm not offering rides.

I remember one afternoon. I was preparing to depart from a wind-torn coastal airstrip and was trying to wrestle the Stinson away from a fence when a gnarled wisp of a man approached.

"Could you use a hand?"

"Great," I shouted against the gale, wondering if he could really help. But together we moved it. Then, as he held the quivering frame I jumped in and cranked up the Franklin.

As the ancient bird and I rose from the runway I spotted the old man, standing alone by the fence, the damp coastal wind whipping at his thin clothes. I waved, the Stinson dipped a wing, and we were gone.

I think he would have liked a ride. Metal prop or not.

(Continued on page 77)

Interlocking Pieces: Blaise and Turbo Pascal.

Whether you're a Turbo Pascal expert or a novice, you can benefit from using professional tools to enhance your programs. With Turbo POWER TOOLS PLUS™ and Turbo ASYNCH PLUS™, Blaise Computing offers you all the right pieces to solve your 4.0 development puzzle.

Compiled units (TPU files) are provided so each package is ready to use with Turbo Pascal 4.0. Both POWER TOOLS PLUS and ASYNCH PLUS use units in a clear, consistent and effective way. If you are familiar with units, you will appreciate the organization. If you are just getting started, you will find the approach an illustration of how to construct and use units.

◆ **POWER TOOLS PLUS** is a library of over 180 powerful functions and procedures like fast direct video access, general screen handling including multiple monitors, VGA and EGA 50-line and 43-line text mode, and full keyboard support, including the 101/102-key keyboard. Stackable and removable windows with optional borders, titles and cursor memory provide complete windowing capabilities. Horizontal, vertical, grid and Lotus-style menus can be easily incorporated into your programs using the menu management routines. You can create the same kind of moving pull down menus that Turbo Pascal 4.0 uses.

Control DOS memory allocation. Alter the Turbo Pascal heap size when your program executes. Execute any program from within your program and POWER TOOLS PLUS automatically compresses your heap memory if necessary. You can even force the output of the program into a window!

Write general interrupt service routines for either hardware or software interrupts. Blaise Computing's unique intervention code lets you develop memory resident (TSRs) applications that take full advantage of DOS capabilities. With simple procedure calls, "schedule" a Turbo Pascal procedure to execute either when pressing a "hot key" or at a specified time.

◆ **ASYNCH PLUS** provides the crucial core of hardware interrupts needed to support asynchronous data communications. This package offers simultaneous buffered input and output to both COM ports, and up to four ports on PS/2 systems. Speeds to 19.2K baud, XON/XOFF protocol, hardware handshaking, XMODEM (with CRC) file transfer and modem control are all supported. ASYNCH PLUS provides text file device drivers so you can use standard "Readln" and "Writeln" calls and still exploit interrupt-driven communication.

The underlying functions of ASYNCH PLUS are carefully crafted in assembler and drive the hardware directly. Link these functions directly to your application or install them as memory resident.

Blaise Computing products include all source code that is efficiently crafted, readable and easy to modify. Accompanying each package is an indexed manual describing each procedure and function in detail with example code fragments. Many complete examples and useful utilities are included on the diskettes. The documentation, examples and source code reflect the attention to detail and commitment to technical support that have distinguished Blaise Computing over the years.

Designed explicitly for Turbo Pascal 4.0, Turbo POWER TOOLS PLUS and Turbo ASYNCH PLUS provide reliable, fast, professional routines—the right combination of pieces to put your Turbo Pascal puzzle together. **Complete price is \$129.00 each.**

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441
Reader Service Number 5

Now for
Turbo Pascal 4.0!

**THE BLAISE
E N U**

Turbo POWER SCREEN \$129.00
NEW! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. Now for Turbo Pascal 4.0, soon for C and BASIC.

Turbo C TOOLS \$129.00
Full spectrum of general service utility functions including: windows; menus; memory resident applications; interrupt service routines; intervention code; and direct video access for fast screen handling. For Turbo C.

C TOOLS PLUS \$129.00
Windows; menus; ISRs; intervention code; screen handling and EGA 43-line text mode support; direct screen access; DOS file handling and more. Specifically designed for Microsoft C 5.0 and QuickC.

ASYNCH MANAGER \$175.00
Full featured interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem control and XMODEM file transfer. For Microsoft C and Turbo C or MS Pascal.

PASCAL TOOLS/TOOLS 2 \$175.00
Expanded string and screen handling; graphics routines; memory management; general program control; DOS file support and more. For MS-Pascal.

KeyPilot \$49.95
"Super-batch" program. Create batch files which can invoke programs and provide input to them; run any program unattended; create demonstration programs; analyze keyboard usage.

EXEC \$95.00
NEW VERSION! Program chaining executive. Chain one program from another in different languages; specify common data areas; less than 2K of overhead.

RUNOFF \$49.95
Text formatter for all programmers. Written in Turbo Pascal; flexible printer control; user-defined variables; index generation; and a general macro facility.

**TO ORDER CALL TOLL FREE
800-333-8087**

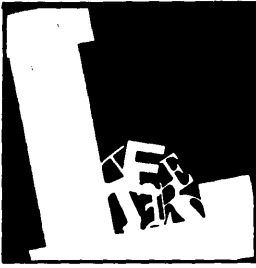
TELEX NUMBER - 338139

YES! Send me the right pieces!

Enclosed is \$_____ for _____ copies of _____
 Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add \$4.00 for
UPS shipping, \$10.00 for Federal Express standard air.

Name: _____ Phone: (____) _____
Address: _____ State: _____
City: _____ Zip: _____
VISA or MC#: _____ Exp. Date: _____

Microsoft
and QuickC are
registered trademarks of
Microsoft Corporation. Turbo Pascal is a regis-
tered trademark of Borland International.



Letters

C Vs. Assembly Language - One

This letter is in response to the article *C vs. Assembly Language* in *Micro C* issue #40.

On the whole, I agree with Mr. Isaacson's assertion that a program is more efficient when written in assembly language than in any other language. Additionally, I agree it can take less time to develop the assembly language program than the C version of the same program—given that well-qualified programmers are working on each team.

I take issue, however, with his absurd example. He makes several statements about the C language that show he knows as much about C as those who write unmaintainable assembly code do about 8086 assembly language.

Most glaring is his statement justifying the construct:

```
iopmem [strlen (iopmem) - 1] = c;
```

He says: "Because of the limitations of the C language, it's impossible at compile time to determine the location of the last element of the fixed-length array..."

In fact, the sizeof operator is a unary operator that, according to ANSI section 3.3.3.4, "yields the size (in bytes) of its operand... When applied to an operand that has array type, the result is the total number of bytes in the array."

Hence the following code:

```
#define iop_user
(sizeof (iop_name) - 1)
volatile char iop_name [] =
"/dev/iopmem/8a1X0*";
main()
{
iop_name [iop_user] = 'c';
}
```

yields the following assembly code when compiled with Microsoft C Vers. 5.0 using "cl -Fa -Gs test.c":

```
;misc constant declarations
_DATA SEGMENT
_iop_name DB '/dev/iopmem/8a1X0*',00H
_DATA ENDS
_TEST SEGMENT
;pre 5.0 MASM setup stuff
PUBLIC _main
_main PROC NEAR
mov BYTE PTR _iop_name+19,99
ret
_main ENDP
_TEST ENDS
```

This is equivalent to Mr. Isaacson's one line of assembly language. To write code like this requires a good understanding of the language.

He points at the common I/O functions defined as part of many languages but implemented as functions in C. He says this is a defect of C; I see it as an advantage. The code isn't linked in unless necessary, and I don't use it very much in a large system where I'm likely to use my own I/O library.

Mr. Isaacson points to the terse nature of this small, elegant language as a fault saying, "that cryptic, incomprehensible code becomes easy to write," and that K&R "give some examples (with a tone of approval!) of atrocious C programming technique."

Is C *more* cryptic than assembly? Is it difficult to write "atrocious" assembly code? Is K&R the Devil's programming guide? Well, isn't that special. I think Mr. Isaacson is not a C-literate programmer.

In his summary, he states that lack of a debugger, short symbol names, and inefficient code make C a poor choice for large efforts. I use Microsoft

C with CodeView, a fine debugger. MSC sports 31-character symbol names, and an excellent optimizer with automatic register allocation, loop optimization, intrinsic code for more common functions (such as strlen) as well as the more common optimizations that Mr. Isaacson finds so trivial.

Well-written assembly code can be more efficient, and certainly more fun to write. But portability, availability of maintenance talent, and client paranoia often preclude its use—not without good reason.

William E. Weinman
1800 S. Robertson Blvd., Ste. 206
Los Angeles, CA 90035

Editor's note: Thanks Bill. And, thanks for agreeing to speak at SOG (a talk entitled "C Isn't The Devil's Own Language").

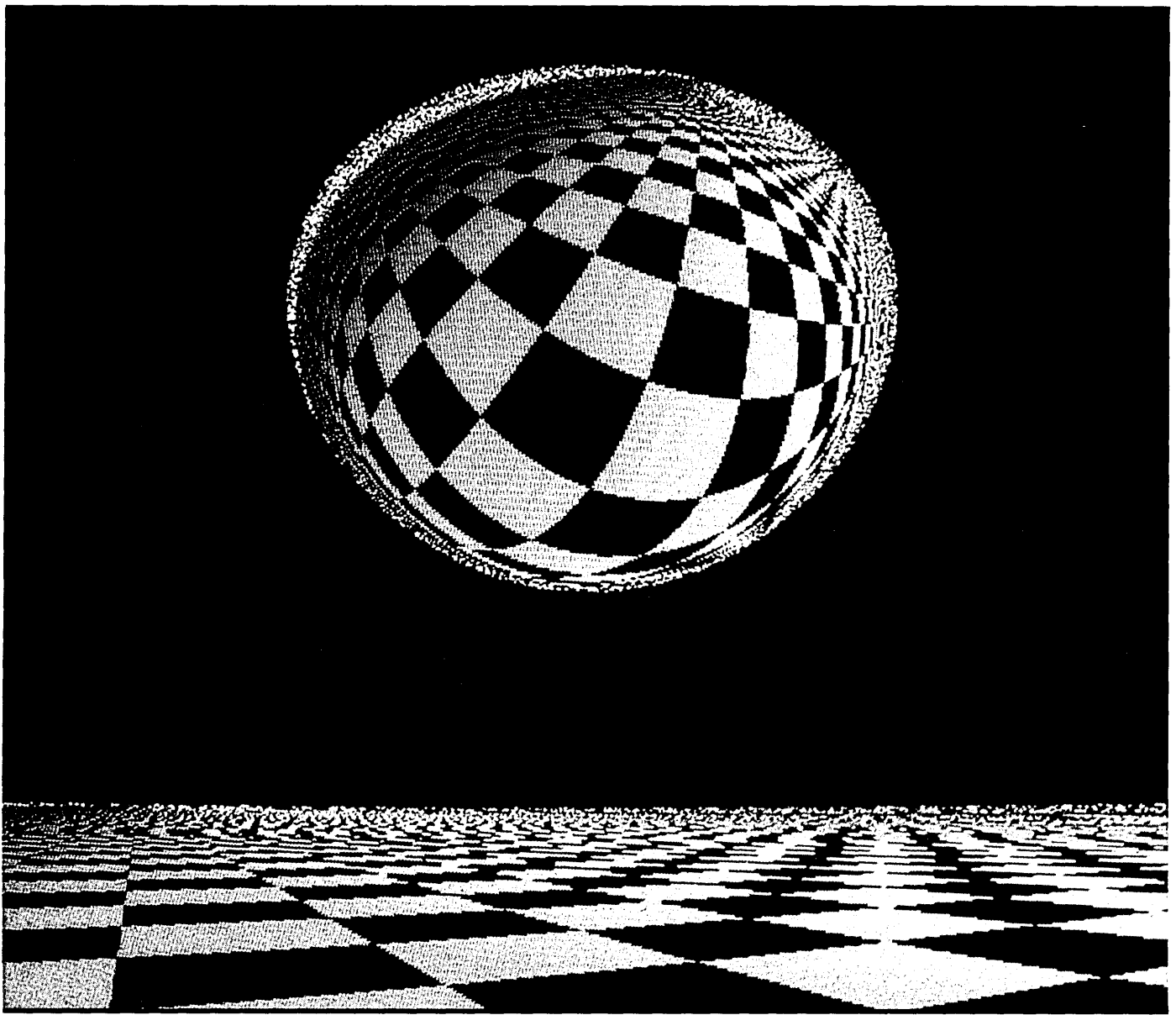
C Vs. Assembly Language - Two

Mr. Isaacson raises a few good points in his article, but I must point out a few incorrect and unduly harsh items.

First let me say that C compilers, especially for PCs, have been greatly improved in the last couple of years. The code generated by Mr. Isaacson's compiler doesn't look like the code that comes out of Microsoft C vs. 5.0. MSC doesn't save SI or DI unless they're used in the function. It also leaves values in registers if convenient, which happens often with pointers that are used more than once.

It may be true that Mr. Isaacson's compiler limited him to seven-character names. But a Unix port I've used allowed 32 characters while the assembler had an eight character limit.

(Continued on page 70)



Graphics In 3-D: *Ray Tracing On The 34010 - Part 1*

The 34010 is the fastest, smartest, most gung ho video controller ever devised (at least by TI). Since it's so smart and fast it may as well help with the chores—chores like creating 3-D images. Here's part one of a very significant two-part series.

While ray tracing provides the motivation and examples, the principles that follow apply to any three-dimensional computer graphic program.

I'll supply 34010 assembly language code which implements the geometric principles. Some if-then conditions will help you navigate through the paragraphs.

If you're not at all interested in programming, you can easily skip the programming sections. I've separated each topic into theory and program-

ming parts, and the programming sections can be skipped without losing the continuity of the theory section.

If you are a programmer, you can use the 34010 examples as suggestions for creating a program on your favorite CPU or in your favorite language. I assume you have no prior knowledge of the 34010 and explain each 34010 instruction I use in the examples.

This isn't a 34010 primer! The most interesting programming instructions, such as pixblt operations, XY addressing, and window violation interrupts are not used here. See the TMS34010 user's guide if you're serious about learning 34010 programming.

If you're an experienced 34010 programmer, I don't want to hear about bugs in the code, or better ways to implement these ideas. Part of my job is to convince my editors I'm an authority in this area. I don't want any wisepersons messing up that myth.

And don't give up on this article if you find the start too elementary. That will change before the end. Everyone, regardless of skill background, should find something of significance in here.

Ray Tracing

Imagine an eye floating in space. If you draw a line from it, the first object it meets is the first thing you'd see if you looked along that line.

The idea is so simple it's embarrassing. But it's important enough to stimulate many interesting computer pictures.

In our computer implementation, the eye will be a point and the computer screen a rectangle in space. The computer universe will consist of a group of simple mathematical objects—spheres, planes, and rectangles.

After you've learned the ropes, you can add other kinds of objects to the program (if you know the mathematical equation describing the object).

So let's get to it.

We can represent an arbitrarily complex object as a collection of polygons, and assign each object a color.

Pick a pixel (or dot) on the screen and draw a line from the eye to that pixel. This type of line, defined by a starting point and a direction, is a ray.

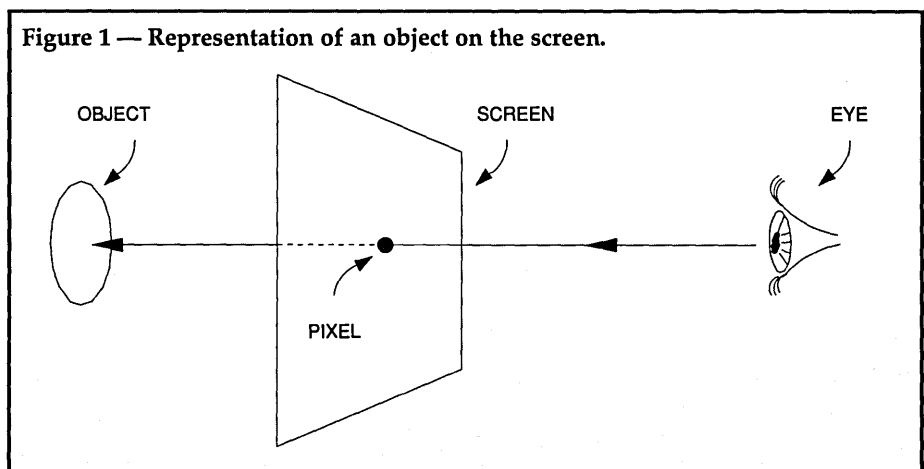
Extend the ray through the screen into space. Determine the first object that the ray intersects (see Figure 1). Assign the pixel the same color as that object. Do this for every pixel on the screen. The resulting image will show one view of the imaginary computer universe.

To create different views of the

the obvious choice for this project, but I prefer 32-bit signed fixed point. Fixed point arithmetic is much faster than floating point with the same number of bits. Accuracy in either case depends on the number of bits.

Fixed point won't, however, let us work on just any scale (floating point will), so the fixed point implementation I'm using here will only give valid results if all values stay within a 32-bit range. But 32 bits will be plenty.

Select a screen size which enables each pixel to be about 1000 times larger than the smallest number we can represent. This will guarantee that any round off error is far smaller than anything we



universe, move the eye and screen around. To make things more interesting, you can move and alter objects or make them reflective, so that rays bounce off and continue on in other directions. The resulting game of light ray billiards generates even more fascinating pictures. (Editors note: Earl generated the graphic on the facing page using PC Tech's 34010 board.)

Arithmetic

Floating point arithmetic may seem

can display on the screen.

We place objects so that the typical distance between them is on the order of 1000 pixels. This means there are plenty of pixels to resolve the object, and the scene will fill the screen.

So the size of the objects and the distances between them are about 1/1000th of the largest number that can be represented. And there's enough room to avoid overflow errors.

Fixed Point

In fixed point representation, we use regular integers, but imagine a decimal point (binary point?) at some fixed location. The fixed point will be set to the right of bit 16 here.

Thus we can represent numbers from -65535 to +65535 (-2^{16} to 2^{16}) with a resolution of 0.00002 (2^{-16}).

Let "A" be a real number, and suppose "a" is the integer representation of "A" in the computer. Then "A" and "a" are related by the simple formula—

$$A = a * 2^{-16}$$

Suppose "B" and "b" are similarly related, a representation of (A + B) becomes—

$$(A + B) = (a * 2^{-16}) + (b * 2^{-16}) \\ = (a + b) * 2^{-16}$$

So (a + b) is the fixed point representation of (A + B). In other words, fixed point addition is just simple integer addition. Subtraction works the same way.

For multiplication observe that—

$$(A * B) = (a * 2^{-16}) * (b * 2^{-16}) \\ = (a * b * 2^{-16}) * 2^{-16}$$

Thus, the representation for (A * B) is just (a * b) shifted right 16 bits.

For division—

$$(A / B) = (a * 2^{-16}) / (b * 2^{-16}) \\ = (a / b) \\ = ((a / b) * 2^{16}) * 2^{-16}$$

So fixed point division is integer division followed by a left shift of 16 bits.

34010 Programming

This will be the first time many readers see 34010 program code, so I'll briefly summarize the 34010.

The 34010 has 30 general purpose 32 bit registers, which are denoted A0 - A14 and B0 - B14.

The stack pointer can be denoted by SP, A15 or B15.

The B registers have special uses in the advanced graphic instructions. For example, in the PIXBLT (pixel block transfer) instruction, the B registers contain the source and destination start address, the block size, and window dimensions for clipping.

When XY addresses are used, one of the B registers specifies the screen origin. And when the source is a binary array, the foreground and background colors are held in the B registers.

Instructions with two operands have

the form "Operation Source, Destination." The source comes first then the destination. For example "add a0, a1" adds a0 to a1 and stores the result in a1. This requires some adjustment for programmers used to Intel processors.

We use double registers in the multiply and divide instructions. Double registers are even and odd pairs with the even register holding the most significant bits. For example "mpys a1, a0" multiplies a1 and a0 and puts the top 32 bits in a0 and the bottom 32 bits in a1.

Figure 2 is the implementation of 32-bit signed fixed point multiplication; 32-bit integer multiplication is one machine instruction on the 34010. The result is 64 bits wide and is put into two 32-bit registers.

$$x = (x + a / x) / 2$$

If we run this long enough, x will become the square root of a. Figure 3 computes the square root.

The push instruction has a strange name, mmtm. The first instruction in Figure 3 is "mmtm sp, a4, a5, a6." This instruction says to push registers a4, a5, and a6. mmtm stands for "move multiple to memory." The pop instruction is mmfm for "move multiple from memory."

"bts" is the bit test instruction. "dsjne" is the name TI gives to the loop instruction; it stands for "decrement skip jump not equal."

The instruction "dsjne a6, fs_01" near the end of Figure 3 will be skipped

Figure 2 - 34010 Fixed Point Multiplication and Division.

```
fxPt      equ 16      ;define fixed point position.
One       equ 10000H  ;our representation of the number one.

fxMultiply:
;Out:     a0 = a0 * a1
mpys      a1, a0      ;signed integer multiplication
sla       32-fxPt, a0 ;line up high order bits
srl       fxPt, a1    ;line up low order bits
or        a1, a0      ;combine them
rets

fxDivide:
;Out:     a0 = a0 / a1
move      a1, a2      ;save divisor
move      a0, a1      ;build dividend in 64 bit register,
                    ;with fxPt shift
sra       32-fxPt, a0 ;high order bits
sll       fxPt, a1    ;low order bits
divs      a2, a0      ;signed integer division
rets
```

END OF LISTING

To get the fixed point result, the integer result must be right shifted 16 bits. The 34010 doesn't have a double register shift instruction, so separate shifts are done on each register. Then the results are combined with an OR. The shifts used provide 0 fill, so the unused bits are effectively masked off.

The division instruction in the 34010 takes a 64 bit double register and divides it by a 32 bit divisor. The quotient goes into a 32 bit register.

To avoid losing precision in the fixed point division routines, the shift is done first, then the division. The dividend is left shifted 16 bits in the 64-bit register pair. The "sra" instruction is a sign extending right shift. The register pair is then divided by the 32-bit divisor.

Since that went so well, let's try a square root, which we can compute with the iterative formula—

if the result of the previous operation was zero. Otherwise it will decrement the a6 register and jump to fs_01 if a6 isn't zero. There are also "dsj" and "dsjeq" instructions.

The first time I wrote 34010 code, I found myself drowning in registers. I was used to Intel CPUs, which have a relative shortage of registers. Some kind of convention on register usage has to be adopted to successfully write a 34010 program.

The subroutines in this article use the following conventions.

B registers are only used for the special graphic operations. Subroutines return values in the low A registers, A0 - A3. Arguments which can be destroyed are sometimes passed in the low registers. These registers are undefined when not used to return values.

Registers A4 - A7 are used as local

variables within a subroutine. They're always saved, and never used for passing arguments.

A8 - A11 are used to pass arguments to a subroutine. The values in A8 - A11 are always preserved by the subroutine. A12 - A14 are used to pass arguments which may change. The subroutine may change the value of these registers, but not the meaning.

For example, if a pointer is passed in A12 to a subroutine, then the subroutine

are examples of vectors.

You may have heard the expression that vectors have both length and direction. At first a point appears to have neither length or direction. But the point gets a length and direction if we think of it as being defined by the vector from the origin to that point.

We can add or subtract two vectors by adding or subtracting their coordinates—

$$P + Q = (Px + Qx, Py + Qy, Pz + Qz)$$

$$P \cdot Q = Px * Qx + Py * Qy + Pz * Qz$$

The length of a vector, denoted $|P|$, is defined by—

$$|P|^2 = P \cdot P = Px * Px + Py * Py + Pz * Pz$$

In other words, the Pathagorean Theorem.

The dot product can be used to compute the angle between two vectors—

$$\text{The cos of angle between P and Q} = (P \cdot Q) / (|P| * |Q|)$$

As a consequence of this, we see that two vectors are perpendicular (at right angles) when $P \cdot Q = 0$.

Informally, we represent direction by a vector. Sometimes mathematics and computers require us to be exact about such things. In particular, since our direction vector can have any length there are an infinite number of vectors which represent a given direction. When forced to choose a particular one, we'll choose the vector which has length one. We call vectors with length one "unit vectors."

An arbitrary vector can be converted to a unit vector by "normalizing"—

$$P = P / |P|$$

Given two vectors, P and Q, it's sometimes advantageous to break P into two components, one perpendicular to Q and one parallel to Q.

We say the parallel component is the projection of P onto Q. The condition of perpendicularity is the same as the dot product being equal to zero. Two vectors are parallel if one is a scalar multiple of the other.

Mathematically,

$$(1) P = P1 + P2$$

where

$$(2) P1 \cdot Q = 0$$

P1 is the perpendicular component and

$$(3) P2 = c * Q$$

for some number c. P2 is parallel to Q.

The number c can be easily calculated, dot both sides of (1) with Q—

$$P \cdot Q = P1 \cdot Q + P2 \cdot Q \\ = 0 + (c * Q) \cdot Q \\ = c * (Q \cdot Q)$$

So—

$$c = (P \cdot Q) / (Q \cdot Q)$$

Figure 3 - Fixed Point Square Root

```

fxSqRoot:
;compute square root of a by iterating x = (x + a / x) / 2
;out:      a0 = sqrt( a0 )
mmtm      sp, a4, a5, a6      ;save registers which hold intermediate
values.
btst      31, a0              ;test for negative argument
jrnz      fs_00              ;if negative, refuse to compute square
root.
movi      10, a6              ;Iteration limit, we'll be
                             ;done long before this.
move      a0, a4              ;Starting interation value
move      a0, a5              ;Remember what number we're square
rooting
fs_01:
move      a5, a0              ;Get a
move      a4, a1              ;and current x
callr     fxDivide           ;a0 = a / x
add       a0, a4              ;a4 = x + a / x
sra      1, a4                ;a4 = ( x + a / x ) / 2
sub       a4, a0              ;compare with old x (sort of)
sra      1, a0                ;give or take a bit
dsjne    a6, fs_01           ;if not stable repeat
move      a4, a0              ;result to a0
fs_00:
mmtm      sp, a4, a5, a6      ;restore some registers.
rets

```

END OF LISTING

may change the value of the pointer, but the new value will still be a valid pointer.

These conventions are personal rules only. I present them here only as an explanation of the register usage in this article. You should adopt whatever conventions fit your style of programming and the program you're writing.

Three Dimensional Geometry

The simplest object in geometry is a point. An arbitrary point in the three dimensional universe (called 3-space for short) is specified by three coordinates. We write $P = (Px, Py, Pz)$ where P is the point and Px, Py, and Pz are respectively the x, y, z coordinates of P.

We have to specify directions in 3-space. We can describe a direction by $D = (Dx, Dy, Dz)$, where Dx, Dy, and Dz are the directions with respect to each axis.

Point and direction defined this way

$$P - Q = (Px - Qx, Py - Qy, Pz - Qz)$$

If P and Q are points then P - Q is the vector which starts at Q and goes to P.

Two types of multiplication are useful in vector arithmetic—one for multiplying a vector by a number, the other for multiplying two vectors.

The first is called multiplication by a scalar. Each coordinate of the vector is multiplied by the number.

$$c * P = (c * Px, c * Py, c * Pz)$$

Geometrically, this corresponds to stretching or shrinking the vector. The length of the vector changes but the direction does not.

The second type of multiplication is called the inner or dot product. Here two vectors are multiplied and the result is a scalar or number. The corresponding coordinates are multiplied and the products are summed—



Finally —
Affordable Intelligence.

TINY EINSTEIN

The Expert System Shell

- Create your own expert systems in minutes.
- With pulldown menus and windows
- Context-sensitive online help
- Free example expert systems
- Tutorial
- Interactive full-screen text editor
- DOS access from shell
- Turbo Fast execution
- Cluster, Trace, Explain
- For Diagnosing . . .
Simulating . . .
Predicting . . .
Planning . . .
Classifying . . .
Training . . .
and Monitoring systems.

Only \$49.95! (Plus \$5 S/H)

Designed & implemented by
Gary Entsminger & Larry Fogg



ACQUIRED INTELLIGENCE
P.O. BOX 2091 • DAVIS, CA 95617 • (916) 753-4704

Reader Service Number 72

Thus, if P and Q are given, c can be computed and so—

$$P2 = c * Q$$

and

$$P1 = P - c * Q$$

The number c which arises in the above computations has an importance of its own. If Q is one of the standard unit vectors (1, 0, 0), (0, 1, 0), or (0, 0, 1) then c is just the x, y or z coordinate of P.

A situation may arise where you need to change your frame of reference. If Q is one of the axis vectors in a non-standard coordinate system, the value c is the Q coordinate of P. I like to think of Q as a measuring stick. I call the process of extracting the number c, calibrating P with respect to Q.

Turning These Ideas Into A Program

Since I chose a 32-bit fixed point representation for numbers, we'll represent vectors by three such numbers.

I chose not to pass vectors around in registers. Instead vectors are always in memory and I pass memory pointers. A vector in memory consists of three 32-bit numbers in consecutive memory locations. The pointer to the vector points to the first number.

In the 34010, memory addressing is by bits. So if a pointer is pointing to the x coordinate, then the y coordinate is at offset 32, and the z coordinate is at offset 64 from the base. The following definitions identify the individual components—

```
;Point
px equ 0000H
py equ 0020H
pz equ 0040H
```

Memory references in the 34010 can be made to any bit boundary, and can be for 1 to 32 consecutive bits. The 34010 status register has two field size fields. The field size fields tell the 34010 how many bits to read or write during memory accesses.

When a memory value of less than 32 bits is loaded into a 32-bit register, the 34010 will either zero fill the upper bits or sign extend the value through the upper bits. The SETF instruction sets the field size.

Syntax is—

```
SETF <FS>, <FE>, <F>
```

where—

FS = field size, 1 - 32
FE = extend option, 0 = zero fill,
1 = sign extend
F = field number, 0 or 1.

I like to set the field sizes as follows:

```
SETF 16, 1, 0  
SETF 32, 0, 1
```

Field 0 is 16 bit, sign extended; field 1 is 32 bit. All the subroutines here assume these settings. Whenever the field size is changed, it's changed back to the original setting as soon as possible.

The 34010 move instruction has the form—

```
MOVE <src>, <dst>, <F>
```

where—

<src> is the source operand
<dst> is the destination operand
<F> is the field operand

F is either 0 or 1, if it's omitted zero is assumed. Most of the move instruc-

Figure 4 - Basic Vector Arithmetic

```
.data
;for intermediate values
vTemp0 .space 60H
.text
vMove:
;Out:
move a9 = a8
move *a8(px), a0, 1
move a0, *a9(px), 1
move a8(py), a0, 1
move a0, *a9(py), 1
move *a8(pz), a0, 1
move a0, *a9(pz), 1
rets

vAdd:
;Out:
a10 = a8 + a9
move *a8(px), a0, 1
move *a9(px), a1, 1
add a1, a0
move a0, *a10(px), 1
move *a8(py), a0, 1
move *a9(py), a1, 1
add a1, a0
move a0, *a10(py), 1
move *a8(pz), a0, 1
move *a9(pz), a1, 1
add a1, a0
move a0, *a10(pz), 1
rets

vSub:
;Out:
a10 = a8 - a9
move *a8(px), a0, 1
move *a9(px), a1, 1
sub a1, a0
move a0, *a10(px), 1
move *a8(py), a0, 1
move *a9(py), a1, 1
sub a1, a0
move a0, *a10(py), 1
move *a8(pz), a0, 1
move *a9(pz), a1, 1
sub a1, a0
move a0, *a10(pz), 1
rets

vDot:
;Out:
a0 = a8 . a9
mmtm sp, a4
```

tions in the accompanying code involve 32-bit operations and so they end with ",1". Many addressing modes are possible—

```
R      register
*R     indirect
*R+   postincrement indirect
-*R   predecrement indirect
Addr  absolute
*R(dis) indirect with displacement
```

Here R is a register name, Addr is an absolute memory address, and disp is a 16-bit signed displacement. Nearly any combination of source and destination addressing modes is possible.

Figure 4 shows the 34010 implementation of basic vector arithmetic. The routine vMove makes a copy of a the vector pointed to by a8 at the vector pointed to by a9.

vAdds adds a8 to a9 and stores the results in a10. vSub subtracts a9 from a8 and stores the result in a10. Each coordinate in a9 is added or subtracted from the matching coordinate in a8, and the result is put into a10.

vDot returns the dot product of a8 and a9 in a0. a8 and a9 point to vectors and a0 is a number. vScale stores the scalar multiplication of a9 and a8 in a10. a9 is a scalar, and a8 points to a vector.

vCalibrate returns the size of the projection of a8 onto a9 as a multiple of the length of a9. The result is returned in a0. vProjection puts the projection of a8 onto a9 into the vector a10. vPerpendicular puts the component of a8 perpendicular to a9 into a10. vNormalize returns a normalized copy of a8 in a9.

Compound Objects

Now let's use vectors to build our descriptions of more interesting objects.

A ray is half of a line; we define it by a starting point and a direction—

```
Start      S = (Sx, Sy, Sz)
Direction  D = (Dx, Dy, Dz)
```

I like to imagine a point starting at S and traveling out in direction D (see Figure 5). The position of this traveling point at time t is given by—

$$P = S + t * D$$

where

t >= 0 is time.

In a strict mathematical setting, t is just a number, and we say that t parameterizes the points on the ray. By the way, if we also allow t < 0 then the above formula represents a line.

Figure 4—Basic Vector Arithmetic, continued. . .

```

move    *a8, a0, 1      ;Multiply x components
move    *a9, a1, 1
calla   fxMultiply
move    a0, a4          ;start accumulating sum
move    *a8(py), a0, 1
move    *a9(py), a1, 1
calla   fxMultiply     ;Multiply y components
add     a0, a4          ;add to sum
move    *a8(pz), a0, 1
move    *a9(pz), a1, 1
calla   fxMultiply     ;Multiply z components
add     a4, a0          ;return sum of products
mmfm   sp, a4
rets

vScale:
;Out:   a10 = a9 * a8 (a9=scalar, a8=vector)
move    *a8(px), a0, 1
move    a9, a1
calla   fxMultiply     ;scale x component
move    a0, *a10(px), 1
move    *a8(py), a0, 1
move    a9, a1
calla   fxMultiply     ;scale y component
move    a0, *a10(py), 1
move    *a8(pz), a0, 1
move    a9, a1
calla   fxMultiply     ;scale z component
move    a0, *a10(pz), 1
rets

vCalibrate:
;Out:   a0 = a8 projected onto and calibrate by the unit vector a9
mmtm   sp, a4, a8
callr   vDot           ;a9.a8
move    a0, a4
move    a9, a8
callr   vDot           ;a8.a8
move    a0, a1
move    a4, a0
calla   fxDivide       ;a0=(a9.a8)/(a8.a8)
mmfm   sp, a4, a8
rets

vProjection:
;Out:   a10 = projection of a8 onto a9
mmtm   sp, a8, a9
callr   vCalibrate     ;get projection multiplier
move    a9, a8
move    a0, a9
callr   vScale         ;scale a9
mmfm   sp, a8, a9
rets

vPerpendicular:
;Out:   a10 = component of a8 perpendicular to a9
mmtm   sp, a4, a8, a9, a10
move    a10, a4
movi    vTemp0, a10
callr   vProjection    ;temp = projection
move    a10, a9
move    a4, a10
callr   vSub           ;subtract projection to get perpendicular
mmfm   sp, a4, a8, a9, a10
rets

vNormalize:
;Out:   a9 = normalized a8
mmtm   sp, a8, a9, a10
move    a9, a10
move    a8, a9
callr   vDot           ;a8.a8 = square of size
calla   fxSqrt         ;size to a0
move    a0, a1
movi    One, a0
calla   fxDivide       ;invert size
move    a0, a9
callr   vScale         ;then scale
mmfm   sp, a8, a9, a10
rets

```

END OF LISTING

A Plane

An arbitrary plane in space can be specified by designating a point on the plane, and a vector perpendicular to the plane. The point is the origin and is denoted $O = (O_x, O_y, O_z)$. The perpendicular vector is called the normal and is denoted $N = (N_x, N_y, N_z)$.

An arbitrary point $P = (P_x, P_y, P_z)$ lies on the plane if the vector, which starts at O and goes to P , is perpendicular to N .

The dot product can be used to determine when two vectors are perpendicular. A point P lies on the plane determined by the origin O , and normal N if—

$$(P - O) \cdot N = 0$$

A little algebra here will save us some time later on. The above formula will be the basis of a computerized test of whether a point lies on a particular plane. Consider the following equivalent forms—

$$\begin{aligned} (P - O) \cdot N &= 0 \\ P \cdot N - O \cdot N &= 0 \\ P \cdot N &= O \cdot N \end{aligned}$$

To the mathematician, any one of these is as good as any other. The programmer, however, will prefer the last form.

solving for t —

$$t = (c - S \cdot N) / (D \cdot N)$$

The questions of intersection will be answered by "when" rather than "where." The time parameter t will be

Imagine a rectangle in space in a standard orientation (see Figure 6), and suppose the edges are parallel to the X

Figure 5—Ray Definition

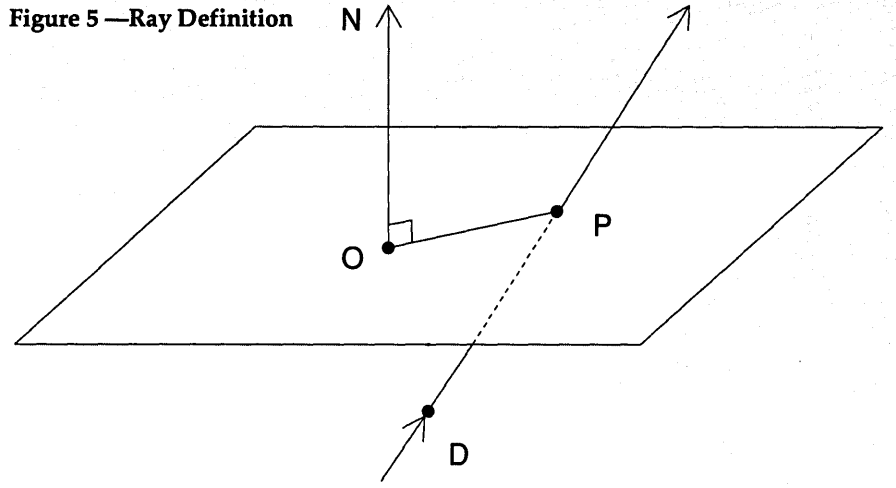
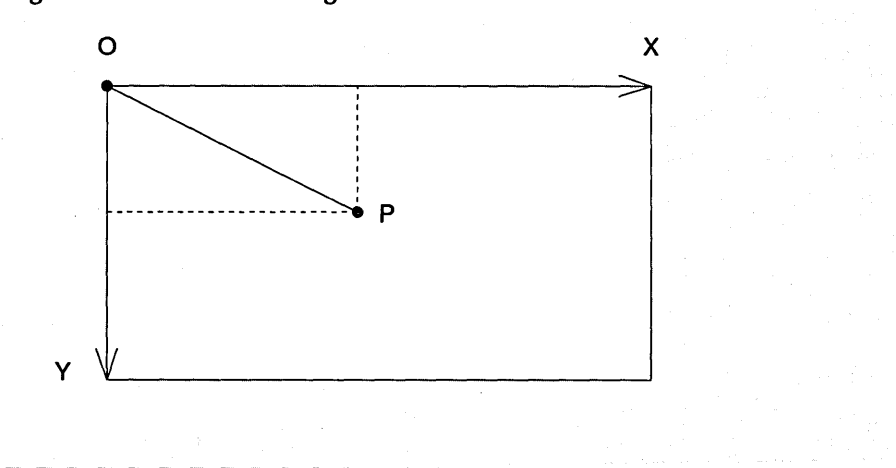


Figure 6—Plane and Rectangle Definition



O and N are part of the definition of the plane. So the quantity $O \cdot N$ is fixed for any given plane. Set $c = O \cdot N$, and then compute c when the plane is defined.

The test for whether a point lies on a plane then becomes—

$$P \cdot N = c$$

The next question we ask is where does an arbitrary ray intersect the plane. Recall that a ray is determined by a starting point S and a direction D , and the ray consists of all points P where—

$$P = S + t \cdot D$$

for some number $t \geq 0$. Thus we see a ray intersects a plane when—

$$(S + t \cdot D) \cdot N = c$$

used to identify intersection.

In ray tracing we want to know what object our ray intersects first. The time parameter answers this, so we often don't need the actual point of intersection.

Of course when we want to know the point of intersection, we can easily compute it from t . Just use the equation for a point on a ray—

$$P = S + t \cdot D$$

A Rectangle

A plane is often too large an object. Usually our flat objects will be rectangles. A rectangle is embedded in a plane, so we add boundary information to the plane information to specify a rectangle.

68000

and 6800/2/8/9

SOFTWARE

SK*DOS - a powerful DOS for the 6809 (\$75) or the 68000 (\$140, incl. an editor, assembler, Basic, utilities, code for a boot ROM, etc.)

HUMBUG - a monitor/boot ROM, \$50 - \$75.


OTHER SYSTEM SOFTWARE including assemblers, text formatters, editors, spell checkers, languages, etc., all very reasonable.

HARDWARE

A wide selection of single-board computers and systems, from \$275.

COMBINATIONS

Package deals of fast and powerful computer plus DOS and more, from \$350.



STAR-K
SOFTWARE SYSTEMS CORP.
BOX 209 - MT. KISCO, NY 10549
914/241-0287

Reader Service Number 40

and Y axes. Define the plane which embeds the rectangle.

When we define a plane, the origin is arbitrary. Any point on the plane can serve as well as any other for the origin. This time pick the upper left corner of the rectangle as the origin. Put a vector on the top edge of the rectangle by starting at the origin and going to the top right side. Call this vector the X extent and define the Y extent along the left side in a similar fashion.

A point, P, lies on the rectangle if it's to the right of the origin but not farther to the right than the X extent, and below the origin and but not further below than the Y extent.

We can test this by verifying that the numbers which come from calibrating the vector from the upper left corner to P with respect to both the X extent and the Y extent are between 0 and 1.

A general rectangle isn't restricted to orientation with respect to the coordinate axes. We can expand the description of a rectangle in a standard position in order to describe an arbitrary rectangle.

First define the plane which embeds the rectangle. We can no longer say which side of the rectangle is the top, so just pick any corner to be the origin. Connect two sides of the rectangle to the point we picked for the origin; denote one the X extent and the other the Y extent.

It doesn't matter which side you call what, just remember to define the extents to start at the origin and go to the other corner on that side. The notion of calibrating one vector with respect to another can be used to test whether a point extends beyond the X or Y extents.

In summary, we define a rectangle by adding two vectors, the X extent and Y extent—

N, the normal vector,
O, the origin,
c, the plane constant ($O \cdot N$)
X, the X extent, and
Y, the Y extent

(The astute reader will observe that N and c are redundant, in the sense that the rectangle could be fully defined from O, X and Y. N and c are carried along to simplify calculations.)

A point, P, lies in a rectangle if three conditions hold. First the point must lie in the plane of the rectangle—

$$(1) \quad P \cdot N = c$$

Second, the projection of the point onto the X extent must lie on the "top" side of the rectangle—

$$(2) \quad 0 \leq ((P - O) \cdot X) / (X \cdot X) \leq 1$$

The same applies to the Y extent—

$$(3) \quad 0 \leq ((P - O) \cdot Y) / (Y \cdot Y) \leq 1$$

To find out whether a ray intersects a rectangle, first determine the point where the ray intersects the plane which contains the rectangle. Then ask whether the point of intersection lies on the rectangle.

A Sphere

Representing a sphere is very straightforward. A sphere is given by a center point, C, and a radius, r. We can use the dot product to measure distance. So a point P is on a sphere if—

$$(P - C) \cdot (P - C) = r^2$$

By recalling the formula for a point on a ray we see that a ray intersects a sphere at time t where—

$$(S + t \cdot D - C) \cdot (S + t \cdot D - C) = r^2$$

I'll spare you the details of the algebra, but this can be put into the standard form for quadratic equations—

$$a \cdot t^2 + b \cdot t + c = 0$$

where—

$$\begin{aligned} a &= D \cdot D \\ b &= 2 \cdot (D \cdot (P - C)) \\ c &= (P - C) \cdot (P - C) - r^2 \end{aligned}$$

so,

$$t = -b \pm \sqrt{b^2 - 4ac}^{1/2}$$

Part 2

That's it for theory. In part two I'll get into the details of implementing these ideas on the 34010. If you have questions that won't wait until next time, drop me a line (or two).



Locate C Bugs before they Bite with PC-lint

PC-lint will analyze your C programs (one or many modules) and uncover glitches, bugs, quirks, and inconsistencies. It will catch subtle errors before they catch you. By examining multiple modules, PC-lint enjoys a perspective your compiler does not have.

"... a remarkable well thought-out product which will check for just about every conceivable coding error ... Its value increases with frequent use ... we confidently recommend PC-lint."

Andrew Binstock, The C Gazette

"PC-lint has everything going for it: flexibility, speed, good documentation, and a reasonable price. I exercised the product daily on a large, working, project to see if I could include it in my development tools. The answer is a definite YES."

*Stephen D. Cooper, Blue Notes
San Francisco PC Users Group*

"... friendliness is a prime consideration in PC-lint. Gimpel has provided ways to make Lint shut up about all those errors you either know or don't care about. If Unix-Lint was implemented as well, I would use it more."

Don Malpass, IEEE Software

Gimpel Software

3207 Hogarth Lane
Collegeville PA 19426
(215)584-4261

PRICE: \$139.00 first copy, \$100 each additional, MC, VISA, COD, PA residents add 6% sales tax, Outside USA add \$15.

Runs on MS-DOS, works with any C compiler - direct support for 12 major C compilers including Microsoft 5.0, Turbo, C86+, Lattice, Datalight, Desmet
PC-lint is a trademark of Gimpel Software.

Neural Networks

Learning Systems Based On The Brain

During the edit I found myself talking with Diane Ingraham, one of the authors. And during that discussion, talk turned to the usefulness of the neural models.

"We're using the model to help our robot see."

"I see."

"For instance, the robot views a whole desktop. We put an egg on the desk, then replace the egg with a walnut. We tell the robot what's unique about each scene.

"Then we put the robot into a whole new environment, put one of the objects somewhere in the new scene, then ask it whether the new scene is more likely to contain the egg or the walnut."

That's exciting. It's not that I'm especially into eggs or walnuts, but think about the possibilities for teaching machines to recognize objects, despite the fact that the objects may lie in new surroundings, or may make up only a tiny part of a scene. This is front edge stuff, folks. (Now, where's that egg illustration? Egg??? I think this computer's got its neurons on backwards.)

Each of us has a brain, a sophisticated information processing system, which reads and interprets sights, sounds, smells, and feelings. Some computer scientists are trying to recreate this sophisticated "thinking" system by using brain cells (or neurons) to model devices.

In our brain, billions of interconnected neurons communicate with each other via excitatory and inhibitory electrochemical signals. This circuitry, (a network of seemingly simple cells) helps us see, hear, speak, learn, think, create, remember, and act.

Although a human brain is a little slow (from a computer's perspective), it consumes little power and can process information in parallel. By parallel processing, our brains can draw con-

clusions by associating many kinds of information.

A run-of-the-mill computer processes serially, yet more accurately and faster than a human being. It roars through calculations at (often) amazing speeds while brains plod through simple math at a snail's pace.

Though computers store knowledge in megabytes, they can't draw on it without specific instructions (i.e., a program). Meanwhile, given a few simple clues, a five-year-old child can recall relevant events quickly (by associating).

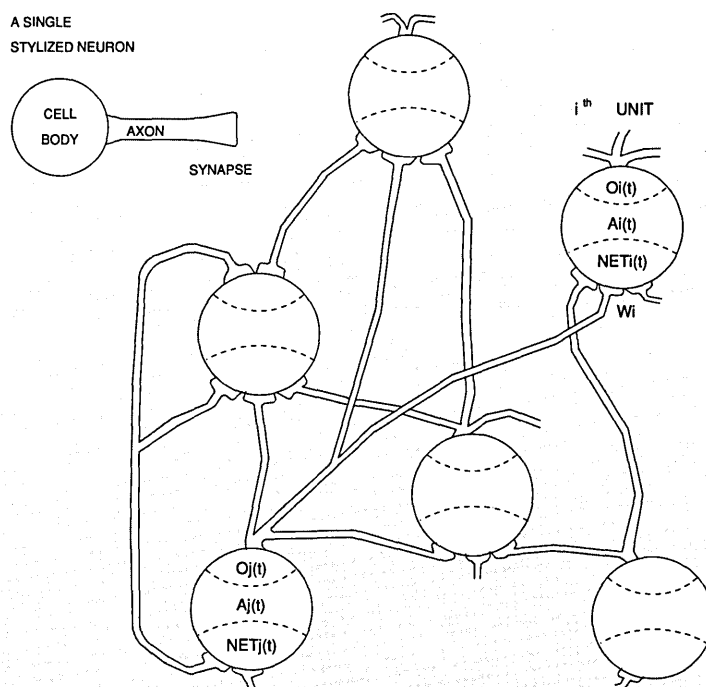
Human brain power—speech recognition, perception, vision, language

processing, language acquisition, sequential thought processing, and consciousness—far exceeds that of any known computer.

What we've been talking about is a model, a simplified biological model, wherein the brain is a system of many interacting parts whose overall state evolves (or changes) continuously. The way it evolves depends on the connections between neurons. Not all neurons affect all other neurons, and not all neurons which affect each other, do so equally.

Based on this model, work began in the 40's and 50's to model computing devices around the neural network theories. That work stalled in the mid-

Figure 1 - Small Neural Network & Neuron



60's because of hardware and software limitations.

Computers simply weren't powerful enough to test the neural network theories. (And, AI gurus Minsky and Papert were attacking the theories.)

Then, in 1982, John Hopfield showed that important problems could indeed be solved by neural networks. This led to a revival in neural network research and in the last three years there's been an explosion of academic and commercial interest. Many companies are now funding attempts to create neural networks in hardware as well as software.

In this article, we'll briefly describe how a neural network works, then go into the details of two useful models.

Neural Networks

Neural networks are self-organizing systems of simple, interconnected processing units (See Figure 1).

One way to look at these units is to consider each as a representation of a hypothesis about an input domain. The hypothesis could be anything; but in its simplest form, you might think in terms of a two-state digital input/output system (1s and 0s). Neurons firing or not.

The strength or weight of the connection between any pair of units could represent the correlation between the two hypotheses.

For example, if two neurons are connected (can influence each other), we give them a value of 1. If they aren't connected, we give them a value of 0.

Obviously, our brains are a bit more complicated than this, but even at this level things can heat up fast. Fifteen neurons would require over 100 connections. Fifty neurons—well over a thousand connections.

And our brain has billions of neurons! Even a Cray wouldn't have a chance in a brain-size network. So, naturally enough, researchers imagine small "artificial" net-

working systems based on the simplest form of a brain's neural network.

In these artificial networks (or simulations) knowledge is stored in a pattern of connections and weights which the network uses for learning, processing, and generalizing from problems.

The nodes in an "artificial" neural network represent processing elements, and each can accept multiple inputs. The operation of the nodes is determined by differential equations (transfer functions) which specify the change in output (with time) based on the input signals. The coefficients of these transfer functions can be modified following a "learning" rule.

Although neural net models found in the literature differ in the details of their structures, applications, and training procedures, they all share the following:

- A set of processing units or neurons
- An activation state, a_i , on or off, (high or low) of the system at time (t) determined by the states of all processing units
- An output rule to determine output, O_i , of a unit
- A pattern of connectivity among all units showing both the connection and its associated weight, W_{ij}
- An activation rule to find the new activation level for a neuron based on its current state and the input it receives, Net_{ij}
- A learning rule for modifying weights
- An operating environment

Each neuron or processing unit can also be "labelled" according to the role it plays:

- Visible units—those that interface with the environment
- Hidden units—those that are unknown to the environment

How Do They Learn?

Just like small children, neural nets

must learn by being taught. We can group the learning into two classes—associative learning and regularity discovery.

(1) Associative Learning—in which the states of all visible units in a system are represented by a list of components (a pattern or vector) corresponding to the output of a unique visible unit.

We train the system on a set of such patterns by fixing or clamping the visible units to represent each pattern in turn.

In the testing mode, we "show" part of a pattern to the neural net, and it completes the pattern by itself. For example, suppose we teach the machine the pattern "10101" along with some other patterns. Then in the testing mode, it will reproduce the pattern, 10101, from a pattern that uniquely defines it (for example, "111").

We can further classify associative learning into two categories depending on which part of the pattern we show the network in the testing phase:

(i) Auto-association—we show the system any sufficiently large set of arbitrarily chosen components of the pattern. Each pattern is then associated with itself, hence the name, "auto-association."

For example, we could train a system using a set of patterns representing photographs of our friends.

Then suppose that during the testing mode we show the system a picture of one of our friends now wearing sunglasses. Ideally, the system will react, filling in all the missing features of the picture (those covered by the sunglasses) and come up with the right identification.

(ii) Pattern Association—where each pattern is divided into two separate patterns: the input pattern and the output pattern.

The goal is to train the system to associate input patterns with output patterns. Then when we present an input pattern to the system in the testing mode,

Unbelievable!

SOURCER™

Creates commented source code and listings from memory, COM or EXE files.

- **CLARIFY UNDOCUMENTED CODE**
- **EASILY MODIFY PROGRAMS**

SOURCER™ creates detailed commented listings and source code directly suitable for assembly. Built in data analyzer and simulator resolves multiple data segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines all necessary assembler directives. Complete support for 8088 through 80286, V20/V30, 8087 and 80287 instruction sets. No other product comes close to the output quality of SOURCER.

BIOS SOURCE

PS/2 • AT • XT • PC • Clones

- **CHANGE & ADD FEATURES**
- **CLARIFIES BIOS INTERFACES**
- **SPECIFIC TO YOUR MACHINE**

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

SOURCER \$ 99.95*
SOURCER

w/BIOS Pre-Processor \$139.95*

(*OUTSIDE USA, ADD \$15 SHIPPING; CA RES. ADD SALES TAX)

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

800-538-8157 x 811 800-672-3470 x 811
(outside Calif.) (inside Calif.)

V COMMUNICATIONS

3031 Tisch Way, Suite 200, Dept. MC
San Jose, CA 95128
(408) 296-4224

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

Reader Service Number 62

it should generate the right output pattern.

(2) Regularity Discovery—where a machine learns regularities in the input/output patterns. By building an internal model of the environment, the network is able to recognize underlying structures.

Models

Now that we've described neural networks generally, we'll move into the details of two simple neural net models that can be programmed on personal computers.

The first is a pattern associator which can be trained to associate input patterns with output patterns. The second is a Boltzmann Machine.

Pattern Associator

The neural network in Figure 2 has an input layer, a hidden layer, and an output layer. Each input neuron is connected to every hidden neuron and to every output neuron. Each hidden neuron is connected to every output neuron.

In this version of the neural net

During the first phase, we supply an input pattern (for example, 110) to the input units, causing some output pattern to appear at the output units. Unless your neural net is a genius, the output likely won't be the target pattern you want.

So, in the second phase, we calculate the difference between the target pattern and the output. (Using network jargon, we say "calculation propagates back from the output layer through the hidden layer to the input layer, modifying weights if necessary.")

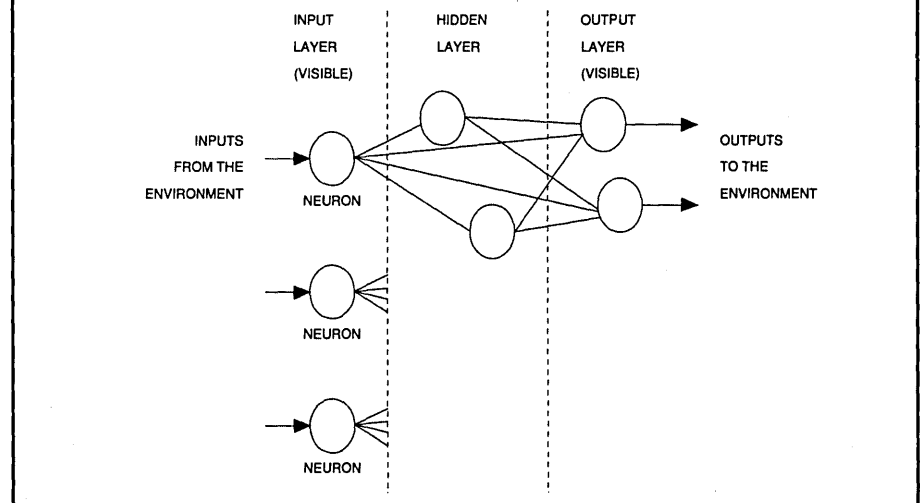
The learning procedure alternates between the first phase and the second phase until the system learns to associate the input pattern to the target pattern. (Hopefully, the thing converges.)

We can test the knowledge of the system by looking at the input units of an input pattern and verifying the output produced.

Units in any layer are updated synchronously. The equations we use during the first phase to calculate the net input, the activation, and the output of each unit are as follows:

The net input into an input unit i is

Figure 2 - A Three-layer Neural Net Used As A Pattern Associator.



model, there are no interconnections within any layer. (Though the theory doesn't exclude these connections.)

The hidden neurons try to capture "higher-order" features characteristic of the inputs and outputs. These characteristics cannot be extracted using simple pairwise connections between inputs and outputs.

A Learning Algorithm

The training rule for this model works in two phases.

given by—

$$Net_i(t) = I_i(t) + B_i$$

where $I_i(t)$ is the i^{th} component of the input pattern, and B_i is the bias of the i^{th} input unit.

The total input into any hidden unit and/or output unit U_j is—

$$Net_j(t) = \sum_j (W_{ji} * O_j(t)) + B_j$$

where W_{ji} is the weight of the connection

from unit i to unit j , and $O_j(t)$ is the actual output produced by unit j at a time t when a particular input pattern is shown to the net.

The activation value of any unit is calculated by the logistic function—

$$a_i(t) = \frac{1}{1 + \exp[-\text{Net}_i(t)]}$$

$(0 < a_i(t) < 1)$

The output function is simply a threshold function—

$$O_i(t) = 1 \text{ if } a_i(t) > 0.5$$

$$= -1 \text{ otherwise}$$

The net input into any hidden or output unit is found from the following equation—

$$\text{Net}_i(t) = W_{ij}(t) * O_j(t) + B_i$$

If the output $O(t)$ isn't equal to the target $T(t)$, then the difference $T(t) - O(t)$ propagates through the system according to the following equations during the second phase.

The error signal $\delta_i(t)$ for each of output signal is determined by—

$$\delta_i(t) = [T_i(t) - O_i(t)] * a_i(t) * [1 - a_i(t)]$$

where $T_i(t)$ is i^{th} component of the target pattern $T(t)$.

The weight for each connection into output unit i is changed by—

$$W_{ij}(t) = f * \delta_i(t) * O_j(t)$$

where f is the learning rate (typically $0 < f < 1$).

The error signal for a middle unit i is found from—

$$\delta_i(t) = a_i(t) * [1 - a_i(t)] * B_{ki} * W_{ki}(t)$$

where δ varies for all output units.

The weights for all connections into middle unit i are changed by—

$$W_{ij}(t) = f * \delta_i(t) * O_j(t)$$

Boltzmann Machine

The first neural net pattern associator is an interesting model, but it is not of much use unless you know all the elements of the input and output patterns.

What happens if some of the elements are missing? That is, if you have an input

pattern, say made up of three elements, 101 for example. Then the model is helpless if you input only 10_ or _01 or 1_1.

One of the beauties of our own brains is that we can make inferences about missing information. The Boltzmann Machine is a version of neural net model which can recognize patterns even though the data is faulty or incomplete. It can "guess" what the missing elements are! In fact, it can even be trained using incomplete data. (Similar to human training.)

Background

Some background first: In 1982, John Hopfield proved that a network of symmetrically interconnected units would operate by the same mathematical rules as a ball rolling around on a bumpy surface. The network tends to settle into one of many stable energy states (where the potential energy function has a local minimum) in much the same way as the ball would settle into a dimple on the surface, refusing to move until someone or something gives it a poke.

This network is essentially a model of associative memory where each local minimum in the energy function corresponds to an association being stored in the network.

A Boltzmann Machine is nothing more than one of these Hopfield networks that uses a special kind of poke called "Simulated annealing" to get out of local minima in order to settle into the globally minimum state, (or close to it).

The trick is in how the interconnection weights are updated.

To program a Boltzmann Machine you must set up an "Annealing Schedule" and then run through the Learning Procedure.

First the Annealing Schedule:

The Annealing Schedule is a set of "temperatures" T and time units t as shown in the box below.

Annealing Schedule

Run the model for t_i time units at

T_1	T_2	T_3	...	T_i	...	T_q	(decreasing)
t_1	t_2	t_3	...	t_i	...	t_q	(increasing)

temperature T_i . The T_q is the equilibrium temperature. Initially the schedule might look like the one shown below.

As the learning progresses, the

9.2	5.3	3.2	1.8	0.5
2	2	4	6	10

temperatures in the schedule can be changed after every phase⁺ using—

$$T_i = \frac{[\text{energy of unclamped units in phase}^+]}{[\# \text{ unclamped units in phase}^+][(3/4)^i - (3/20)]}$$

(a relation that we find works).

Learning Procedure:

1. Set the Annealing Schedule
2. PHASE⁻ - The Teaching Phase
 - (a) Clamp the input elements to the input neurons and the output elements to the output neurons.
 - (b) Randomize the states of all the unclamped (hidden) neurons.
 - (c) For each temperature T_i (not the equilibrium temperature) in the Annealing Schedule, run the Boltzmann machine for t time units. After every time unit, update the state of each unclamped neuron k using the probability decision rule:

$$P_k = \frac{1}{[1 + \exp(-E_k/T_i)]}$$

(Because we assumed that Boltzmann's energy distribution applies to our neural network behavior.)

At low temperatures, the system tends to converge into the global minimum, but the time required to stabilize is very long. On the other hand, at high temperatures it is more willing to make uphill jumps so it reaches a stable state faster.

The best compromise is to start at a high temperature and gradually reduce it as the machine approaches "thermal equilibrium," a process known as simulated annealing. (Annealing comes from the art of crystallizing metal—to get the best crystallization you alternately heat and cool the metal until it hardens. Here, you're alternately increasing and decreasing the activity of the model, trying to reach the global minimum.)

(d) Now run the machine at the equilibrium temperature T_q and update the units as in (c) above, except this time modify the weights after each update using:

$$\Delta w_{ij} = \delta s_{is} s_j$$

where s_i is the state of unit i , and δ is a "small" positive real number, for example $\delta = 0.005$.

(e) Repeat 2 for the next pair of inputs and outputs (sometimes called the "vector").

3. PHASE⁺ - The Testing Phase

(a) Clamp a part of the vector onto the

input units.

(b) Same as 2(b)

(c) Same as 2(c)

(d) Same as 2(d) except now the weight change is:

$$\Delta w_{ij} = -\delta s_i s_j$$

4. Repeat from 1. until the machine learns; that is, when the output units possess the expected states for several time units in phase⁺.

The process will alternate between the "+" and "-" phases until the network has found the global minimum.

(For those hardy souls who are interested, the Appendix, pp.315-317, at the end of Chapter 7 in Rumelhart, et al, 1986, actually derives the learning rule for the Boltzmann Machine.)

Some interesting questions can be investigated in programming a Boltzmann Machine:

(1) What should the temperature be at which annealing begins?

(2) How gradually is the temperature to be reduced?

(3) What should the temperature

(equilibrium temperature) be at which annealing is terminated?

(4) Can the annealing schedule be modified after each learning cycle?

Potential Applications

Like a new wonder drug, neural nets have been suggested for solving an amazing array of problems, including, but not limited to:

- Exploration in space—planetary rover autonomous path selection, recognition of signals from extraterrestrials, teleoperator controls for building things in space, space flight assistance, design of large space structures like space stations, pattern recognition, image compression (vector quantization), atmospheric modeling, fast threat assessment, damage-tolerant space-based computers
- Military applications—battle management, target identification and recognition, self-guided torpedoes, submarine communications, autonomous spacecraft, cruise missile navigation, surveillance
- Civilian applications—vision, robotic control, adaptive control, prosthesis, fingerprint identification, voice recognition, telecommunications, pattern recognition, man-machine interfaces, associative memory, robust or fault tolerant computers, complex simulations of physical, chemical, biological systems, language, optical computing, data processing

We don't know yet whether neural nets simulated in software or built-in hardware actually think, but something awesome occurs when you watch one work.

Fun Reading Suggestions

Amari, S., & M.A. Arbib (eds.), *Competition and Cooperation in Neural Nets, Lecture Notes in Biomathematics*, Springer-Verlag, Berlin, 1982

Forsyth, R., & R. Rada, *Machine Learning Applications in Expert Systems and Information Retrieval*, Ellis Horwood, J.Wiley & Sons, 1986

Hinton, G., *Learning in Parallel Networks*, BYTE, April 1985

Hinton, G., T. Sejnowski, & D. Ackley, *Boltzmann Machines: Constraint Satisfaction Networks that Learn*, Carnegie-Mellon Technical Report: CMU-CS-84-119, 1984

Hinton, G., T. Sejnowski, & D. Ackley, *A Learning Algorithm for Boltzmann Machines*, *Cognitive Science*, Vol. 9, pp 147-169, 1985

Hoppensteadt, F.C., *An Introduction to the Mathematics of Neurons*, Cambridge University Press, Cambridge, 1986

Lindsay, P. H., & D. A. Norman, *Human Information Processing*, Academic Press, New York 1977

McClelland, J.L., D.E. Rumelhart & the PDP Research Group, *Parallel Distributed Processing Explorations in the Microstructure of Cognition Volume 2: Psychological and Biological Models*, A Bradford Book, MIT Press, Cambridge, MA, 1986

McCorduck, P., *Machines Who Think*, W.H. Freeman & Co., New York, 1979

Minsky, M. & S. Papert, *Perceptrons*, MIT Press, Cambridge, MA 1968

Rumelhart, D.E., J.L. McClelland, & the PDP Research Group, *Parallel Distributed Processing Explorations in the Microstructure of Cognition Volume 1: Foundations*, A Bradford Book, MIT Press, Cambridge, MA, 1986

Torras i Genis, C., *Temporal-Pattern Learning in Neural Models, Lecture Notes in Biomathematics*, Springer-Verlag, Berlin, 1985

♦ ♦ ♦

NEURAL NET



Netzwerk models a neural net in a tutorial level program with source you can change. It demonstrates an associative memory finding nearest match to input word. Limit of 1000 neurons max.

Netzwerk includes a complete copy of PL/D, the compiler language it is written in. PL/D is easy to learn, offers control not in C, and is faster.

Netzwerk \$79.95

Neural Net emulator with source, PL/D compiler, and printed manuals. Add \$75 for source of PL/D. PC, XT, AT compatible with 256K.

DAIR Computer Systems
3440 Kenneth Drive
Palo Alto, CA 94303
(415) 494-7081

Reader Service Number 90

Come See Micro C

(At The West Coast Computer Faire)

We're having a Micro Cornucopia booth at the West Coast Computer Faire April 7 - 10 at the Moscone Center in San Francisco. Stop by (we're in #951) and say hi to Dave Thompson, Larry Fogg, Gary Entsminger, and Bruce Eckel. We'll be glad to talk about article ideas, and anything else.

Note: We won't be offering technical phone help between April 5 and April 11.

HALTED

SERVING NORTHERN CALIFORNIA
SINCE 1963!

SPECIALTIES

DISPLAY PAGING RECEIVER

Designed for use in a nation-wide paging system

- Dual conv. superhet 450 MHz crystal controlled receiver module (plug in module!)
- Twenty character alpha-numeric LED display (ASCII Encoded, ANSI character set)
- RCA CDP 1802 Processor based display/mode controller Piezoelectric "beeper" unit

APPLICATIONS:

- Receiver module may be removed for other uses...
- Receiver could be recrystaled for Ham Radio Use...
- Possible to use for original application on Ham Bands...
- Terrific source of parts...

UNITS ARE COMPLETE BUT UNTESTED & SOLD "AS IS"

\$19⁹⁵ INCLUDING SHIPPING

PC-LabCard FOR IBM AT/XT

Multi-Lab Card..... \$295.-
(12 bit) A/D + D/A + DIO + Counter

Super-Lab Card..... \$495.-
(14 bit) A/D + D/A + DIO + Counter

Digital I/O & Counter Card..... \$175.-

Relay Actuator & Isolated D/I Card.. \$239.-

Prototype Development Card.. \$74.-

NEW

RS-232 BREAK OUT BOX

- SWITCHABLE LINES
- LED INDICATORS
- PATCH TERMINALS & JUMPERS
- COMPACT SIZE

\$31⁹⁵

COMPUTER JOYSTICK
(IBM/APPLE COMPATIBLE)

\$19⁹⁵

LITTON TRACKBALL

- 250 CYCLES PER REVOLUTION
- 5 VDC OPERATION
- SPEC. SHEET INCLUDED

\$39.95

XEBEC 1410 SASI HARD DISK CONTROLLER.

\$ 89.00

PROJECT BOX W/FAN

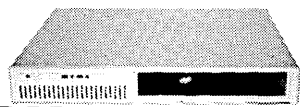
- 3 1/2" x 7 1/2" x 9"
- 3" Fan
- Perfect for mounting power supplies or building projects
- Black Wrinkle Finish

\$9.95

W/Fan

\$1.95

W/O Fan



HALF HEIGHT EXTERNAL DRIVE ENCLOSURE

- Attractive Low Profile Chassis 19x15x2 1/4
- Fits nicely directly under PC
- Standard IBM Colors
- Bezel for 5 1/4" and 3 1/2" Drive

\$29⁹⁵

AT/XT 4 SLOT MOTHER BOARD EXPANDER

INCLUDES:

- 4 SLOT XT MOTHERBOARD
- OPTIONAL AT CONNECTORS
- CABLING
- CASE NOT INCLUDED
- GREAT FOR R & D

\$69⁹⁵

IEEE-488 HARD DRIVE CONTROLLER

- ST506/412 Compatible
- Manufactured by Xebec
- Great for the Experimenter

\$29⁹⁵

3 CONVENIENT LOCATIONS:

HSC Electronic Supply of Santa Rosa
6819 S. Santa Rosa Ave.
Cotati, CA
(707) 792-2357

HSC Electronic Supply
5549 Hemlock St.
Sacramento, CA
(916) 338-2545

3060 COPPER RD., SANTA CLARA 95051

WE SHIP C.O.D.

MAIL ORDERS

Call NOW! (408)732-1573

Store Hours:
Mon-Fri 8:00-7:00
Saturday 9:00-5:00

TERMS: Minimum order \$10. California residents add 7% sales tax. Prepaid orders send freight C.O.D. or call for charges. Shipping will be added to credit card and C.O.D. orders. Prepaid orders over \$100 use money order or certified check. Please do not send cash. Some items limited to stock on hand. Prices subject to change.

Reader Service Number 11

The Logic Of Programming Languages

Proving The Compiler

Paul's an academic. He needs to prove that something will work before he creates it. After he carefully made the point that he could prove, mathematically, that Trilog works, I asked him to do an article on proving languages in general. Here it is.

In a broad sense, a program is a sequence of symbols used to control a machine. The symbols can be characters on a sheet of paper, holes punched into cardboard, bits in a computer, a sequence of keystrokes, etc.

The computing "machine" can be a human performing computations on a sheet of paper, an automatic loom controlled by a Jacquard card, an electronic computer, or a hand-held calculator.

We can use a mathematical conjecture called the Church Thesis to capture the meaning of programs for a specific computing machine by means of a computable relation—

$Eval(p, i, o)$

where p is a suitable encoding of a program, i is an input to the machine, and o is the output.

The relation $Eval(p, i, o)$ holds if and only if the machine obeys the instructions of p by consuming the data i and halting with the output o .

Our programming languages are based on variables, which are used in expressions, which are part of statements, which are grouped into subroutines.

The relation $Eval$ defining the meaning of a program can be directly implemented in a computer in the form of an interpreter, or in a decidable relation—

$Compile(p, q)$

which holds whenever the object program q is obtained from the source program p by a process called compilation.

The computer interprets the object program q in a way specified by the relation $Eval$.

This behavior can be captured in terms of source programs by the relation $Eval$, which we can define as— $Eval(p, i, o)$ holds if and only if we have $Compile(p, q)$ and $Eval1(q, i, o)$ for the object program q

Classical Procedural Programs

Now I'll formally describe a simple procedural language called L , which contains assignment, if, and while statements. Figure 1 shows the syntax of the language, L , in the so-called Backus normal form.

We're not building a parser for L here so we can ignore the fact that our grammar is ambiguous. We've also omitted the obvious definitions of variables and numbers.

Consider the simple program L in Figure 2. Figure 3 gives the result of tracing the execution of the program with input—

$x := 2; n := 5;$

To trace the execution of the program is simple enough, but it's much harder to deduce that the program computes the value x^n into the variable y provided $n \geq 0$. The underlying algorithm is called a logarithmic exponentiation, and is the fastest way of computing integer powers.

The relation $Eval$ (see Figure 4) specifies the formal semantics of L in the notation used first by Smullyan and later adopted for the programming language Prolog.

It goes like this. You can read a clause—

$A \leftarrow B, C, D$

as "in order to establish the relation A , establish the relations B , C , and D ." Prolog programmers should have no trouble understanding the definition of $Eval$.

Although you can visualize the definition as a Prolog program, you really need know nothing about Prolog's computa-

Figure 1 - The Syntax of the Language L in BNF

```
Expr ::= Variable | Number | Expr Oper Expr | ( Expr )
Oper ::= + | - | * | / | < | =
Stats ::= Empty | Stat Stats
Stat ::= Variable := Expr; |
        if Expr then Stats end; |
        if Expr then Stats else Stats end; |
        while Expr do Stats end;
```

END OF LISTING

Figure 2 - A Program to Compute the Function Exp

```
y := 1;
while n > 0 do
  n1 := n / 2;
  if 2*n1 < n then
    y := y*x;
  end;
  n := n1; x := x*x;
end;
```

END OF LISTING

tion method (called SLD-resolution). In short—we instantiate a rule by replacing all of its variables by constants before

using it in the process of establishing relationships.

The Eval(s,bi,bo) relation holds iff (if

and only if) the program, "s," executed under the variable bindings given by "bi" stops with, possibly new, variable bindings, "bo."

We use a string representation of programs to emphasize that a program is a sequence of symbols. Strings are decomposed with the help of "high-powered" but obvious string pattern-matching constructs. Alternatively, we could have used lists to encode programs in the so-called abstract syntax.

Variable bindings are lists of variables coupled with their current values.

For example—

[('x',2), ('y',1), ('n',5)]

Editor's note: The symbols, [and] enclose a list.

The relation Select(b,var,val) holds iff the variable var is associated with the value val in the bindings b. The relation Newb(bi,var,val,bo) holds iff the bindings in bo are like bi but with the value val associated with the variable var. This relation obviously captures the effect of the assignment var := val.

The relation Val(exp,b,val) holds iff the value of expression exp computed under bindings b is val.

The meaning of programs in L is given indirectly by the relation Eval. Nevertheless, this is a perfectly valid approach and the programs are amenable to mathematical analysis. We can prove facts about the relation Eval.

For example, let's denote by P the while loop of the program in Figure 2.

By using induction on the variable n we can easily prove that—

If $n \geq 0$ and p is the smallest number of the form $p = 2^k$ such that $p \geq n$ then,

Eval (& 'P',
 & [('y',y), ('x',x), ('n',n)],

Figure 3 - A Trace of Program Exp Execution

whiles executed	& y	& x	& n	& n1
1	& 1	& 2	& 5	& -
2	& 2	& 4	& 2	& 2
3	& 2	& 16	& 1	& 1
4	& 32	& 256	& 0	& 0

END OF LISTING

Figure 4 -Eval. Definition

```

Val('N',b,v) <- Number('N',v)
Val('V',b,v) <- Select(b,'V',v)
Val('E1 + E2',b,v1+v2) <- Val('E1',b,v1), Val('E2',b,v2)
Val('E1 - E2',b,v1-v2) <- Val('E1',b,v1), Val('E2',b,v2)
Val('E1 * E2',b,v1*v2) <- Val('E1',b,v1), Val('E2',b,v2)
Val('E1 / E2',b,v1/v2) <- Val('E1',b,v1), Val('E2',b,v2)
Val('E1 = E2',b,1) <- Val('E1',b,v1), Val('E2',b,v2), v1 = v2
Val('E1 = E2',b,0) <- Val('E1',b,v1), Val('E2',b,v2), v1 v2
Val('E1 < E2',b,1) <- Val('E1',b,v1), Val('E2',b,v2), v1 v2
Val('E1 < E2',b,0) <- Val('E1',b,v1), Val('E2',b,v2), v1 = v2

Number('D',v) <- Dig('D',v)
Number('ND',10*v+d) <- Dig('D',d), Number('N',v)
Dig('0',0) <-
Dig('1',1) <-
...
Dig('9',9) <-

Select( [('V',v)|t], 'V',v) <-
Select( [('W',w)|t], 'V',v) <- 'V' <> 'W', Select(t,'V',v)

Eval('V := E; S',bi,bo) <-
Val('E',bi,v), Newb(bi,'V',v,bb), Eval('S',bb,bo)
Eval('if E then S1 else S2 end; S',bi,bo) <-
Val('E',bi,0), Eval('S2',bi,bb), Eval('S',bb,bo)
Eval('if E then S1 else S2 end; S',bi,bo) <-
Val('E',bi,1), Eval('S1',bi,bb), Eval('S',bb,bo)
Eval('if E then S1 end; S',bi,bo) <-
Eval('if E then S1 else end; S',bi,bo)
Eval('while E do S1 end; S',bi,bo) <-
Eval('if E then S1 while E do S1 end; end; S',bi,bo)

Newb([('V',w)|t], 'V', v, [('V',v)|tt]) <-
Newb([('W',w)|t], 'V', v, [('W',w)|tt]) <- 'W' <> 'V',
Newb(t,'V',v,tt)

```

END OF LISTING

```
('n1',n1) ],
  & [ ('y',y \times x^{n}),
 ('x',x^{p}), ('n',0), ('n1',0)] ]'
```

Thus the program $y:=1; P$ computes $y = x^n$.

Schools Of Semantics

Our definition of semantics for L is called definition by an abstract machine giving the operational semantics.

A school of semantics called the denotational semantics was very influential a decade ago and claimed that operational semantics isn't a proper semantics because it tells us only what a program does without telling us what the program actually is.

Declarative Languages

We can assign meaning to programs independent of a computation machine only if we restrict ourselves to a special kind of programming language called a declarative language. We can interpret declarative programs as functions or as relations.

In the former case we obtain functional languages (for instance Lisp), in the latter case we obtain logic languages (for instance Prolog, Trilog). Functions and relations are legal mathematical objects treatable by methods of mathematical logic. Declarative programs obtain their meaning via the standard logical method of interpretations.

Declarative languages have a great

logic and efficiency, say at the level of Pascal. The programming language C will probably stay with us as a system implementation language, but we're convinced that most applications can be written in declarative style.

Trilog is a new declarative programming language in which we've combined our theoretical research experience in the area of programming language semantics with the practical experience of compiler writing.

Trilog is a language firmly based on predicate calculus. Its programs are collections of predicates (relations). Trilog combines the areas of procedural, database, and logic programming within a simple language of only eight constructs.

To illustrate the procedural part of Trilog, let's consider the predicate $Exp1$ (see Figure 5).

$Exp1$ computes the exponential function by the logarithmic method. The arguments x , n , and a are declared as input integer arguments, n is restricted to the non-negative interval.

When we call the predicate $Exp1$, the input arguments will have the appropriate values. The value of the integer output argument y will be computed by the call. Note the "iff" word saying that the predicate $Exp1(x,n,a,y)$ holds if and only if the formula in its body holds.

From the definition of $Exp1$, we can easily prove the theorems in Figure 6.

By a straightforward induction on n , we then obtain:

$$Exp1(x,n,a,y) \leftrightarrow y = x^{\{n\}} * a$$

Hence the call to $Exp1$ with the third argument set to 1 computes the exponential function for a non-negative n :

$$Exp1(x,n,1,y) \leftrightarrow y = x^{\{n\}}$$

There's even a simpler method of introducing Exp into the Trilog code given in Figure 7. Both Exp and $Exp1$ use the same fast logarithmic exponentiation.

The predicate Exp requires a true recursion whereas the recursion in $Exp1$ is a "tail recursion." Tail recursion is recognized by the Trilog compiler and is automatically optimized to a loop, as in Figure 2.

Logic Programming

We've seen that we can introduce $Exp1$ as a predicate. It's interesting to note that this predicate can be simulated in Pascal by defining a procedure with the following heading—

Figure 5 - Tail Recursive Exponentiation in Trilog

```
proc Exp1(x:<I,n:<[0..],a:<I,y:>I) iff {comm: n >= 0 & y = x^n * a }
  if n = 0 then
    y = a
  else
    if n mod 2 = 1 then a1 = x*a else a1 = a end &
    Exp1(x*x,n/2,a1,y)
  end
END OF LISTING
```

Figure 6 - Proving Theorems

```
Exp1(x,0,y,y)
  n >= 0 & Exp1(x * x,n,x * a,y) -> Exp1(x,2 * n+1,a,y)
  n >= 0 & Exp1(x * x,n,a,y) -> Exp1(x,2 * n,a,y)
END OF LISTING
```

Denotational semanticists were certainly right. By giving the evaluation predicate (or relation) $Eval$, we specify the computations. A program is mere data to the computing machine and generally doesn't have a meaning per se.

However, after criticizing the machine approach most of the denotationalists proceeded to define a so-called meaning function mapping programs into a computational mechanism known as lambda calculus. True, the lambda functions possessed perfect meaning in the so-called models of lambda calculus, but the meaning function was nothing but the relation $Eval$ in disguise.

The mistake of denotationalists was their belief that computations can be described by models instead of machines. They have somehow forgotten that the notion of computability can be defined only through computing machines, be it by the schemas of recursive functions or by Turing machines.

advantage over procedural languages in that they're free of side effects. The communication between programs is through parameters only. As a consequence, the programs can be relatively easy to prove correct. Since the programs don't depend on any hidden assumptions, they're easier to maintain and more rugged.

The great computer scientist Tony Hoare once observed that "Algol was an improvement over all programming languages succeeding it." We can paraphrase him and say that— The semantics of logic is a real improvement over the denotational semantics which came fifty years later.

The declarative languages used to have quite inefficient implementations. The mainstream programmers could thus claim that they are just academic toys and that real programmers use C. Fortunately, declarative languages have come of age (although not all C programmers have).

It's now clear that we can have sound

ADD TO THE POWER OF YOUR PROGRAMS WHILE YOU SAVE TIME AND MONEY!

CBTREE does it all! Your best value in a B+tree source!

Save programming time and effort.

You can develop exciting file access programs quickly and easily because CBTREE provides a simple but powerful program interface to all B+tree operations. Every aspect of CBTREE is covered thoroughly in the 70 page Users Manual with complete examples. Sample programs are provided on disk.

Gain flexibility in designing your applications.

CBTREE lets you use multiple keys, variable key lengths, concatenated keys, and any data record size and record length. You can customize the B+tree parameters using utilities provided.

Your programs will be using the most efficient searching techniques. CBTREE provides the fastest keyed file access performance, with multiple indexes in a single file and crash recovery utilities. CBTREE is a full function implementation of the industry standard B+tree access method and is proven in applications since 1984.

Access any record or group of records by:

- Get first
- Get previous
- Get less than
- Get greater than
- Get sequential block
- Get all partial matches
- Insert key and record
- Delete key and record
- Change record location
- Get last
- Get next
- Get less than or equal
- Get greater than or equal
- Get partial key match
- Get all keys and locations
- Insert key
- Delete key

Increase your implementation productivity.

CBTREE is over 6,000 lines of tightly written, commented C source code. The driver module is only 20K and links into your programs.

Port your applications to other machine environments.

The C source code that you receive can be compiled on all popular C compilers for the IBM PC and also under Unix, Xenix, and AmigaDos! No royalties on your applications that use CBTREE. CBTREE supports multi-user and network applications.

CBTREE IS TROUBLE-FREE, BUT IF YOU NEED HELP WE PROVIDE FREE PHONE SUPPORT.
ONE CALL GETS YOU THE ANSWER TO ANY QUESTION!

CBTREE compares favorably with other software selling at 2,3 and 4 times our price.

Sold on unconditional money-back guarantee.

YOU PAY ONLY \$159.00 - A MONEY-SAVING PRICE!

TO ORDER OR FOR ADDITIONAL INFORMATION

CALL (703) 356-7029 or (703) 847-1743

OR WRITE



Peacock Systems, Inc., 2108-C Gallows Road, Vienna, VA 22180

Reader Service Number 20

TRILOGY

A Powerful Procedural, Database, and Declarative Language.

SPEED — Where Prolog must backtrack, Trilogly can often solve the problem logically. Trilogly takes advantage of logic constraints (they constrain the search to possible solutions) which either eliminate backtracking or reduce millions of backtracks to a very few.

SYNTAX — Trilogly uses an intuitive, Pascal-like, program structure.

INTEGRATION — Trilogly is complete. It's the only language you need for writing Pascal-style routines, database handlers, and Prolog-style programs.

MODULARITY — Trilogly is modular language, very similar to Modula-2.

ENVIRONMENT — A complete programming environment, you get editor, library, linker, loader, error handling, automatic make, and contextual help. Plus, you get modules for: math, string handling, file manipulation, windows . . .

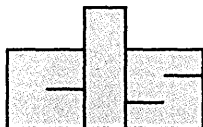
A TRUE COMPILER — Trilogly is an interactive compiler which produces native code for the 8086/8087.

LOGICAL PURITY — Trilogly was designed from scratch as a simple language with a completely logical foundation. Trilogly's speed results from its design, not from added commands. (Prolog's assert, cut, var, and retract, are not logical parts of that language. They were added to improve performance.)

DATABASE SUPPORT — Trilogly supports: variable size records, records with arbitrary values (lists, recursive trees); plus record insertion, deletion, and modification. (Anywhere in the file.) Files are relations and can be queried from within the language.

PRICE — Only \$99.95 postpaid, U.S. funds. Plus \$5.00 shipping & handling. Or \$12.00 shipping & handling outside North America. Check, money order or VISA accepted.

Order From:



COMPLETE
LOGIC
SYSTEMS

741 Blueridge Ave.
North Vancouver BC Canada V7R 2J5
(604) 986-3234

ONLY \$ 99⁹⁵

```
procedure Expl(x:integer; n:integer;
              a:integer; var y:integer); ...
```

and by replacing the ampersand "&" by a semicolon ";".

At the level of logic programming there is no direct translation into the procedural Pascal. At this level Trilogy should be compared with Prolog. Trilogy

persons such that—

- (1) no two persons share the same initial,
- (2) nobody has the initials BF,
- (3) the last name of A is before the last name of B.

There are 3! (3 factorial = 6) possible combinations.

Notice that the only initials satisfying

which seems to us slightly counterintuitive.

The types of the first and last name initials are defined in Trilogy as enumerated types. The predicate Names is satisfied by an array n correlating the first names (indices) to the last names (values).

The double arrow "->>" says that the values of the array are all different. Trilogy derives the solution a = D, b = E, c = F without a single backtrack just by finding the unknown array n satisfying the constraints.

The "initials" problem is of course an artificially simple one. A medium-size logic problem has typically a search space of about 1.5 million candidates. A declarative Prolog program must generate all candidates before finding the ones satisfying the constraints. Such a program typically executes for about 90 minutes (Turbo-Prolog on an XT).

The same Trilogy program is shorter because the clues can be directly translated almost word by word from English. What is even more important, the Trilogy program will generally execute under two seconds with only about five backtracks.

Really big problems (billions of candidates) cannot be even contemplated in Prolog by the generate and test method. Trilogy can do them within minutes.

Figure 7 - Truly Recursive Exponentiation in Trilogy

```
proc Exp(x:<I,n:<[0..],y:>I) iff { n >= 0 & y = x^n }
  if n = 0 then
    y = 1
  else
    Exp(x,n/2,a) & { a = x^(n/2) }
    if n mod 2 = 1 then y = a*a*x else y = a*a end
  end
END OF LISTING
```

Figure 8 - Generate & Test in Prolog: A=d, B=e, C=f

```
names(A,B,C) <-
  permute([d,e,f], [A,B,C]),
  B <> f, less(A,B).

less(d,e). less(d,f). less(d,f).

permute([], []).
permute(T, [H|T2]) <- delete(H,T,T1), permute(T1,T2).

delete(H, [H|T], T).
delete(H, [H1|T], [H1|T1]) <- delete(H,T,T1).

<- names(A,B,C).
END OF LISTING
```

Figure 9 - Constraints in Trilogy. The query is all Names(a,b,c)

```
First = A | B | C
Last = D | E | F

pred Names(n::First->>Last) iff
  n(B) <> F & n(A) < n(B)
END OF LISTING
```

can deal with the same data structures as Prolog and can also perform backtracking to compute non-deterministic predicates.

But unlike Prolog, Trilogy is the only available logic programming language offering constraint satisfaction. Constraints eliminate blind backtracking by solving systems of equations and inequalities.

I can illustrate the constraints of Trilogy with an example which I have intentionally simplified to the bare essentials.

A Small Puzzle

Given three first name initials "A", "B", "C", and three last name initials "D", "E", "F", find the initials of three

the constraints are AD, BE, and CF. The Prolog program (see Figure 8) must generate all 6 permutations before it can find the correct one by a satisfaction test.

Note that the Prolog test B<>f cannot be executed until the variable B has become instantiated by the predicate permute.

In Trilogy there are no such restrictions, the above is simply remembered as a constraint on the solution. You can see from the program in Figure 9 that there's no need for explicit generation of candidates for solutions.

Note that variables in Trilogy start with small letters, whereas constants, being proper names, are capitalized. Prolog uses just the opposite convention

Prolog As A Procedural Language

The implementers of Prolog were quite aware of the exponential blow out of the generate and test method and introduced a number of new "features" such as cuts, asserts, retracts, vars, etc. The features were introduced after implementers realized that they could cleverly utilize the insides of a Prolog engine to shortcut the search. That they also shortcut the logic didn't matter.

We now solve the puzzle of initials using Prolog constructs called var and nonvar. The Prolog engine always knows whether a variable has a value or not. The implementers made this visible to the programmer. The construct var(X) succeeds if X has no value. It fails otherwise. nonvar is the negation of var.

The constructs are called extra-logical since X=2, var(X) fails whereas var(X), X=2 succeeds. Hence, there is no way to give a logical meaning to var(X).

Nevertheless, programmers quickly realized that by using var they could draw the same inferences as humans do when solving this kind of problem.

Imagine a 3 by 3 matrix of zeros and ones with first initials as rows and last

Performance and versatility For your CP/M or MS-DOS computer

QP/M

QP/M by MICROCode Consulting

Fed up with the message "BDOS error: R/O"? With QP/M, you'll never lose another file because you changed a diskette. QP/M offers full CP/M 2.2 compatibility with outstanding performance and more commands WITHOUT eating up precious program space. Get such features as automatic disk relogging, simple drive/user selection using either a colon or semi-colon, 31 user areas, drive search path, multiple program command line, archive bit maintenance, and transparent time/date stamping; all in the same space as CP/M 2.2. Installs from a convenient customization menu, no software assembly required. Bootable disks available with CBIOS for Kaypro, Xerox (8" or 5 1/4", -1 or -2), & BBI.

QP/M Operating System, bootable - specify system . . . \$ 64.95
QP/M without CBIOS (installs on any Z80 system) . . . \$ 49.95

Networks

QP/M Network File System by MICROCode Consulting

QP/M Network File System is an efficient local area network allowing up to seven CP/M computers to share peripherals and data resources.

- Transparent operation at speeds up to 11,000 bytes/second in synchronous mode
- Speeds of up to 1,920 bytes/second in asynchronous mode
- Local/remote disk drive and printer support
- Remote peripheral support for modems and real-time clocks
- All stations need not be on the network even though connected
- Local drive access protection and control
- Simple menu oriented configuration utility
- Extended DOS calls are provided for addition of custom network utilities.

Works with interrupt driven Z80 systems such as Xerox 820, Kaypro (KayPLUS & Advent ROMs), Eagle, and other computers running QP/M, or CP/M 2.2

QP/M Network File System \$ 39.95

Hard Disks

Need more speed and storage on your system?

Improve the productivity of your Z80 computer with a hard disk.

HDS Host Board

This daughter board provides a convenient interface for connecting a Western Digital WD1002-05 hard disk controller to your computer.

- Plugs into the Z80 socket, no other wiring required
- 40 pin interface for a WD1002-05 (or HDO) controller board
- Switch selectable I/O port addressing
- Comes as bare board or assembled & tested
- Kaypro '84 host board also available

Winchester Connection by MICROCode Consulting

The most simple and comprehensive hard disk software package available for CP/M.

- Designed for use with the WD1002-05 controller board
- Works with one or two hard disks - 5 to 64 meg
- Menu installed, no software to assemble
- Complete hardware tests and error handling
- Automatic swap, for warm boots from hard drive
- Software drivers install above or below CP/M
- Allows custom partition sizes and mixed drive types
- Independent block and directory sizes on each partition
- Includes manual, format, test, park, and swap utilities

Winchester Connection Software only \$ 39.95

HDS Board with Winchester Connection Software . . . \$ 79.95

HDS Bare Board with software \$ 59.95

HDS Board, WD1002-05, and software . . . \$245.00

Call or write for other pricing options

WD1002-05 HARD DISK CONTROLLER BOARD by Western Digital

- Standard ST506 drive interface
- Same size as standard 5 1/4" drive
- 40 pin interface to host computer
- WD2797 floppy disk controller interface on board
- Can control up to three hard drives
- Direct replacement for Kaypro 10 controller

WD1002-05 Controller Board \$185.00

Other Western Digital boards available

Prices subject to change without notice. VISA and Mastercard accepted. Include \$5.00 shipping and handling, \$7.50 for COD, UPS-Blue or RED Label additional according to weight. Please include your phone number with all correspondence.

Kaypro

KayPLUS ROM Set by MICROCode Consulting

Want more performance and flexibility from your Kaypro? With the KayPLUS ROM set you can have the advantages of a Kaypro 4 or 10, even on your Kaypro 2.

- Install up to four floppies and two hard drives
- Boots from floppy or hard disk
- Supports 96 TPI and 3 1/2" disk drives
- Can use any ST506 type hard drive - 5 to 64 Meg
- 32 character type-ahead keyboard buffer
- Automatic screen blanking (not avail. on 83 series)
- 12 disk formats built-in, unlimited configurable
- Full automatic disk relogging with QP/M
- Internal real-time clock support
- No software assembly required

Includes manual, format, configuration, diagnostics, sysgen, diskette customization utility, AND hard disk utilities. Available for '83 and '84 series Kaypros.

KayPLUS ROM Set, specify model \$ 69.95

KayPLUS ROM Set with QP/M \$125.00

Parts and accessories for the Kaypro

Kaypro 2X Real-time Clock parts kit \$ 29.00
 Kaypro 2X Hard disk interface parts kit \$ 16.00
 Kaypro 10 or '84 series Hard Disk host board . . \$ 49.00
 Kaypro four drive floppy decoder board \$ 35.00
 Complete parts and repair services available

Xerox 820

PLUS2 ROM and X120 Double Density Board by MICROCode Consulting and Emerald Microware

About had it with single density diskettes on your Xerox 820-1? Get unsurpassed versatility with our X120 Board and PLUS2 ROM package.

- Run up to four floppy disk drives at once
- Mix 8" and 5 1/4" at the same time
- Software compatible with Kaypro and Xerox 820
- Built in drivers for most serial and parallel printers
- Get mini-monitor functions and auto-boot capability
- 19 built in disk formats, including Xerox and Kaypro
- Includes custom disk format definition program
- Banked ROM BIOS for more space in your TPA
- Composite video adaptor on X120 board
- Runs 48 TPI diskettes on 96 TPI drives
- Supports real time clock from Z80-CTC
- Works on the Xerox 820-1 and Big Board I
- Both ROM and X120 board are required for operation

PLUS2 ROM Set and X120 Board A&T \$114.95

PLUS2 ROM Set and X120 Bare Board \$ 49.95

PLUS2 ROM Set only \$ 39.95

120 Bare Board only \$ 15.00

*** Special *** 2 boards for \$25, 5 for \$50

Other kits, parts, and packages available

Parts and accessories for the Xerox 820

Xerox 820-2 CPU Board - new \$ 75.00
 Xerox 820-2 Floppy Controller board - new \$ 65.00
 Xerox 820-2 CPU board w/ Floppy Controller . . \$125.00
 Xerox 820-1 CPU board - new \$ 75.00
 Xerox 820 complete high profile keyboard . . . \$ 65.00
 Xerox 820 bare high profile keyboard - new . . \$ 25.00
 Xerox 820 5 1/4" drive cable \$ 9.00
 Xerox internal video cable w/brightness control . \$ 9.00
 Xerox 820 power supply \$ 35.00
 Power connector, specify board or cable . . . \$ 2.50
 Xerox parallel printer interface cable \$ 35.00
 Dual Half Height 5 1/4" Disk Drives - DSDD, in cabinet with standard Xerox cable \$265.00
 Complete parts and repair services available



P.O. Box 1726, Beaverton, OR 97075



(503) 641-0347



30 day money back guarantee on all products.

Reader Service Number 10

IBM PC

CP/M, NorthStar, Macintosh, Apple II, MS-DOS, and PS/2 - Don't let incompatible diskette formats get you down, read them all with your PC.

UniForm-PC by MicroSolutions

How often have you wished you could use your CP/M diskettes on your PC? Now you can access your CP/M disks and programs on your MS-DOS computer just as you would a standard MS-DOS diskette. Once the UniForm driver is installed, you can use standard DOS commands and programs right on your original diskette without modifying or copying your files. UniForm-PC allows you to read, write, format, and copy diskettes from over 275 CP/M and MS-DOS computers on your PC, XT, or AT. With UniForm-PC and the Compaticard, you can use 5 1/4" high density, 96TPI, dual format 3 1/2" (720k/1.44 meg.- PS/2), and even 8" drives.

UniForm-PC by MicroSolutions \$ 64.95
 UniForm for Kaypro and other machines \$ 64.95

Compaticard by MicroSolutions

Meet the Compaticard, THE universal disk drive controller card. This half card will let you run up to 16 disk drives (4 per Compaticard) on your PC or XT, including standard 360K, 96 TPI, high density (1.2 meg, dual speed), 8" single or double sided (SD or DD), and dual format 3 1/2" drives (720K/1.44 - PS/2). The combinations are almost unlimited. Comes with its own MS-DOS driver and format program for high density and 3 1/2" diskettes. Use it with UniForm-PC for maximum versatility. 8" adaptor and additional cabling available.

Compaticard Board \$169.95
 Compaticard with UniForm-PC. *** Special *** \$225.00
 Compaticard with UniFORM-PC & high density or 3 1/2" drive *** Special *** \$350.00

MatchPoint-PC by MicroSolutions

The MatchPoint-PC board for the PC/XT/AT works with your standard controller card to let you read and write to NorthStar hard sector and Apple II diskettes on your PC. INCLUDES a copy of the UniForm-PC program, as well as utilities to format disks, copy, delete, and view files on Apple DOS, PRODOS, and Apple CP/M diskettes.

MatchPoint-PC Board \$169.95

MatchMaker by MicroSolutions

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

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

Frustrated because your PC can't speak CP/M?

UniDOS by Micro Solutions

Run CP/M programs on your PC? Of course. UniDOS is a memory resident program that can use the NEC V20 CPU chip to actually RUN your favorite 8080 programs. Use UniDOS with UniForm-PC, and automatically switch to CP/M mode as you log on your CP/M diskette. Switch to emulation mode to run Z80 code programs or for systems without a V20. UniDOS directly converts video and keyboard emulation for Kaypro, Xerox 820, Morrow, Osborne, VT100, and eight other displays. All standard CP/M system calls are supported. Note: The NEC V20 CPU is a fast, low power, CMOS replacement for the 8088 CPU chip that includes a full 8080 instruction set as well as the standard 8088 set. Systems using an 8086 may substitute a V30 chip.

UniDOS by MicroSolutions \$ 64.95
 UniDOS w/UniForm and V20-8 chip \$135.00

UniDOS Z80 Coprocessor Board by MicroSolutions

This 8 Mhz. Z80H half-card will run your Z80 and 8080 code programs at LIGHTNING speed on your PC or AT. Functions just like the UniDOS program, except NO V20 or emulation mode is required to run your programs. Now includes UniForm-PC!

UniDOS Z80 Coprocessor Card \$169.95

initials as columns. The element BE of the matrix is set to one only if there is a person with such initials.

Obviously, if a value in the matrix is set to one, all other values in the same row and column must be zeros. Vice versa, if two values in the same row or column are zeros the third value must be a one.

Figure 10 shows a Prolog program using var. The query names(Mat) is answered by finding the matrix Mat. Note the help given to Prolog by encoding the third clue. It says that either AD is an initial or else AE must be (and then BD is not).

After the constraints have been set, the "predicate" infer is repeatedly called as long as there is at least one row or column inference. This is achieved with the help of another extra-logical feature called cut, "!".

Where there's a cut, the program will not backtrack past the cut to find alternate solutions. The row and column inferences are done in the "predicate" rcinf. When the inferences saturate the matrix, we must still check that all initials have been set.

This almost unreadable program seems to be a terrible overkill for such a simple problem. Unfortunately, the use of var and nonvar to simulate constraints is the only feasible way of solving large

We think the hacks shouldn't have happened. The designers should have insisted on the purity of both languages. But perhaps it couldn't have been helped. After all Lisp was designed al-

Figure 10 - Extralogical Use of Prolog

```
names([AD,AE,AF],[BD,BE,BF],[CD,CE,CF]) <-
  BF = 0,                                     %second clue
  (AD = 1; AE = 1, BD = 0),                   %third clue
  infer([AD,AE,AF],[BD,BE,BF],[CD,CE,CF]).  %draw all inferences
novar(AD),novar(AE),novar(AF),novar(BD),novar(BE),novar(BF),
novar(CD),novar(CE),novar(BF).              %see that all is set

infer([AD,AE,AF],[BD,BE,BF],[CD,CE,CF]) <-
  ( rcinf(AD,AE,AF,A); rcinf(BD,BE,BF,A); rcinf(CD,CE,CF,A);
  rcinf(AD,BD,CD,A); rcinf(AE,BE,CE,A); rcinf(AF,BF,CF,A); A=ok),!,
  ( A=inferred, infer([AD,AE,AF],[BD,BE,BF],[CD,CE,CF]);
  A=incons, fail).

rcinf(X,Y,Z,A) <- var(X),!,inf(X,Y,Z,A).
rcinf(X,Y,Z,A) <- var(Y),!,inf(Y,X,Z,A).
rcinf(X,Y,Z,A) <- var(Z),!,inf(Z,X,Y,A).
rcinf(X,Y,Z,incons) <- X+Y+Z<1.

inf(1,Y,Z,inferred) <- nonvar(Y), nonvar(Z), Y=0, Z=0, !.
inf(0,Y,0,inferred) <- nonvar(Y), Y=1, !.
inf(0,0,Y,inferred) <- nonvar(Y), Y=1, !.
inf(_ ,Y,Z,incons) <- nonvar(Y), nonvar(Z), Y=1, Z=1.

END OF LISTING
```

problems with millions of possibilities in Prolog.

Not only are the programs unreadable, but any connection to logic is lost. Such programs can be explained only in a procedural way by a sequence of changes to bindings of variables. Once again a relation Eval is the only way to assign meaning to Prolog programs.

Why The Logic Was Lost

Both Lisp and Prolog were originally designed as nice and tidy languages standing firmly on mathematical foundations. Lisp was designed by McCarthy at MIT. Prolog was designed by Colmerauer at Marseilles. Unfortunately, neither of the designers watched closely over the implementations.

In a dubious quest for quick efficiency, the implementers at MIT (Lisp), and in Marseilles as well as in Edinburgh (Prolog), added "features" to both languages.

The features increased the efficiency but unfortunately added the (euphemistically called) meta-logical features. In plain English, the logic of Lisp and Prolog is gone. Both languages are mere procedural languages where the bindings to the variables must be done in an exact sequence.

The extra-logical features of Lisp are the dynamic binding, the association lists, rplaca, setq...

most 30 years ago and Prolog more than 15 years ago. Perhaps it was then impossible to achieve efficiency without sacrificing logic.

Today, however, there is no justification for hacks in declarative languages. Efforts to standardize Lisp and Prolog by including features violating logic are misguided.

The programming language Trilogy is uncompromising in its logic. Trilogy is very efficient but it obtains the efficiency without violating the logic. This was achieved by a tight control over the implementation. Whenever we had an implementation problem or whenever we have seen that we can improve the efficiency, we have always asked the question of how to do it within the logic.

The efficiency of Trilogy in non-deterministic cases is achieved by constraints which eliminate blind backtracking. The use of types and modes by the Trilogy compiler helps to generate Pascal-like code quality in deterministic cases.

◆ ◆ ◆

OVERSTOCK & DEMOS COMPUTERS ACCESSORIES

KAYPRO 2, 2X & 4
\$150 - \$200

Z-80 CO-PROCESSOR
BOARDS

BABY BLUE \$50
TURBO SLAVE \$100

USED 3M DC 300 XL
TAPE CARTRIDGES
\$5.00

All plus shipping charges

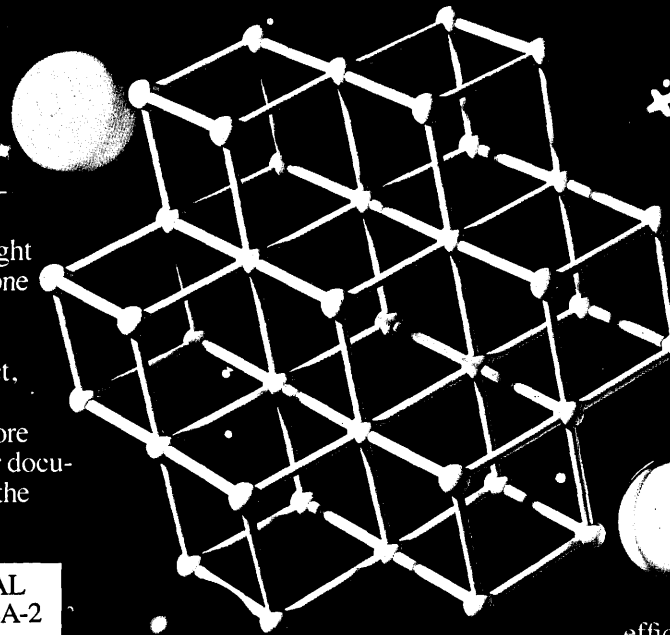
**PROJECT DATA
SYSTEMS, INC.**

1050 Northgate Dr. #200
San Rafael, CA 94903
(415) 492-1840

SOFTWARE ENGINEERING COMES OF AGE.

ANNOUNCING LOGITECH MODULA-2 VERSION 3.0

Modula-2 is the language of choice for modern software engineering, and LOGITECH Modula-2 is the most powerful implementation available for the PC. The right language and the right tools have come together in one superior product. Whether you're working on a small program or a complex project, with LOGITECH Modula-2 Version 3.0 you can write more reliable, maintainable, better documented code in a fraction of the time at a fraction of the cost.



**FREE TURBO PASCAL
TO LOGITECH MODULA-2
TRANSLATOR**

NEW, IMPROVED DEBUGGERS

Time gained with a fast compiler can be lost at debug time without the right debugging tools. With the powerful Logitech Modula-2 Debuggers you can debug your code *fast*, and dramatically improve your overall project throughput. The Post Mortem Debugger analyzes the status of a program after it has terminated while the dynamic, Run Time Debugger monitors the execution of a program with user-defined break points. With their new, mouse based, multiple-window user interface these powerful debugging tools are a pleasure to use.

NEW, INTELLIGENT LINKER

Links only those routines from a particular module that you need, so you eliminate unreferenced routines and produce smaller, more compact executable files.

NEW, IMPROVED COMPILER

Faster and more flexible. Now its DOS linker compatible object files (.OBJ) can be linked with existing libraries in C, PASCAL, FORTRAN and ASSEMBLER — so you can build on previous development and put the power of LOGITECH Modula-2 to work for you right now. Fully supports Wirth's latest language definition, including LONGINT and LONGSET, which provides large set support including SET, of CHAR. Provides optimization for tighter, more efficient code generation.

NEW EDITOR

Our new, mouse based editor is fully integrated, easy to learn, fast and easy to use, and very customizable. Its multiple, overlapping windows and color support make it easy to manage parts of one file or several files on the screen at one time. You'll love using it — with or without a mouse.

Call for information about our VAX/VMS version, Site License, University Discounts, Dealer & Distributor pricing.

To place an order call toll-free:

800-231-7717

In California:

800-552-8885

- LOGITECH Modula-2 V. 3.0 Compiler Pack **\$99**
Compiler in overlay and fully linked form, Linkable Library, Post Mortem Debugger, Point Editor
- LOGITECH Modula-2 V. 3.0 Toolkit **\$169**
Library sources, Linker, Run Time Debugger, MAKE, Decoder, Version, XRef, Formatter
- LOGITECH Modula-2 V. 3.0 Development System **\$249**
Compiler Pack plus Toolkit
- Turbo Pascal to Modula-2 Translator **FREE**
With Compiler Pack or Development System
- Window Package **\$49**
Build true windowing into your Modula-2 code.
- Upgrade Package
Call LOGITECH for information or to receive an order form.

Add \$6.50 for shipping and handling. California residents add applicable sales tax. Prices valid in U.S. only. Total Enclosed \$ _____

VISA MasterCard Check Enclosed

Card Number _____ Expiration Date _____

Signature _____

Name _____

Address _____

City _____ State _____

Zip _____ Phone _____

LOGITECH

LOGITECH, Inc.

6505 Kaiser Drive, Fremont, CA 94555

Tel: 415-795-8500

In Europe: LOGITECH, Switzerland

Tel: 41-21-87-9656 Telex 458 217 Tech Ch

In the United Kingdom: LOGITECH, U.K.

Tel: 44908-368071 Fax: 44908-71751

The Mysteries Of RS-232

Success In This Field Requires Connections

Asynchronous serial communication is definitely one of the weirder parts of personal computing. Although RS-232 is a "standard" interface, you can't just plug things together and expect them to work.

In this article, I'll discuss my strategy for cabling problems, talk about RS-232 on the IBM PC and its clones, try to dispel some of the mystery of communications protocols, and otherwise entertain you.

What's Serial?

In serial communication, you force all the information through a single wire, one bit at a time. Sounds strange, doesn't it? Instead of each bit (in a byte) having its own wire, as on a parallel printer port, each bit has to wait its turn on a single wire.

There are two methods for communicating serially, synchronous and asynchronous.

In synchronous communication (see "Controlling Synchronous Serial Chips

With A Parallel Port" in *Micro C* #39) there are two signals: data and clock. Because of the clock (which tells the receiving end when the next bit is valid) there's no need to pause between bytes to reset both the sender and receiver. (Thus there are no stop or start bits.)

In asynchronous communication (by far the most common mode) only the data is sent. First, both ends have to agree on how fast the data is sent (the baud rate). Then, there's a pause between characters (the start bit at the beginning of the character and 1, 1.5, or 2 stop bits at the end).

By keeping track of the start and stop bits, and by knowing how fast data is being sent, the receiver knows when to look for each of the 8 (or 7) data bits that come between.

Figure 1 shows how a byte is transferred. The ones and zeros are "logical"; they don't reflect the voltage values on the RS-232 lines (more about that later). When quiet, the line is in a marking condition (it's marking time). The first edge of the signal (transition from mark to space) notifies the receiver that a new

byte is beginning.

To insure it wasn't just a noise glitch, the receiving device (the 8250 on a PC) rechecks the signal by sampling it several times.

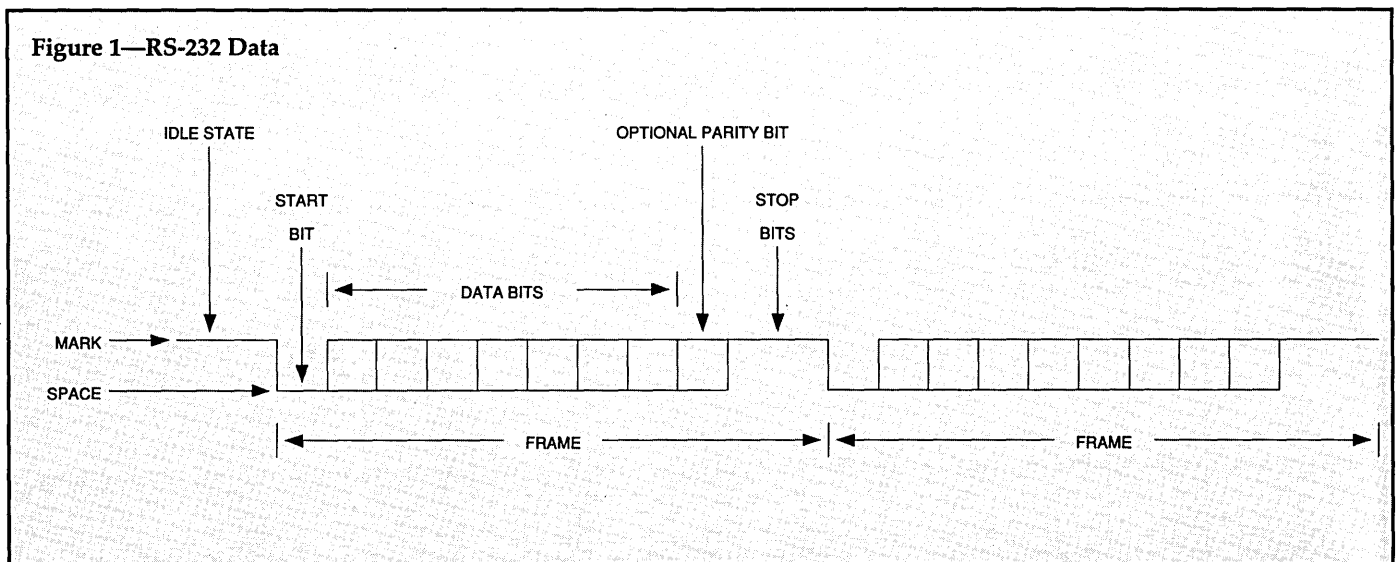
After the last data bit, some extra information arrives. If you have parity turned on, then the 8250 adds a 1 or 0 to make the number of 1 bits in the character even or odd. (Even or odd parity.)

If one of the bits was received incorrectly then the receiving end would notice that the number of ones was no longer correct. If two bits changed, then parity checking wouldn't catch the problem.

The stop bits arrive last; you can use 1, 1.5, or 2. The stop bits return the line to its marking state. They allow the receiving system some time to take care of things before the next byte starts.

The most common configuration you'll probably use is: eight bits, no parity, one stop bit. This means ten bits are transmitted for each byte. So, if you divide your bit rate (also known as baud rate) by ten, the result will tell you approximately how many bytes per second

Figure 1—RS-232 Data



you will move.

For example: at 9600 bits/sec (9600 baud), you'll move approximately 960 bytes per second. This number is only approximate, since most file transfer protocols (described later) add bytes to the beginning and end of each data block.

After a byte has arrived, the 8250 places it in a temporary storage register and sets some flags. If there were an unread byte in the register when the new one came in (destroying the old one), the overrun flag is set. If parity is on and parity isn't right (indicating a noise glitch on the line), the 8250 sets the parity error flag.

If you've set up the system for interrupts (this is performed in software by your communications program), a received byte forces an interrupt. Then the interrupt service routine must grab the byte before it's overrun.

Lotsa Wires

There are only three *really* important wires in serial communication: a line to transmit from you to the other guy, a line to receive information back, and a ground line. You can transmit and receive at the same time; the lines are completely independent.

You may notice there are usually 25 pins on your serial connector. That's more than three. To understand why, some history is in order.

History

The EIA RS-232C (Electronic Industries Association Recommended Standard 232C) was developed solely to connect Data Terminal Equipment (DTEs are computers, terminals, printers and other like machines) to Data Communication Equipment (DCE; i.e., a modem). It consists of the definition of a set of lines through which this communication takes place, and the voltage levels for ones and

zeros through these lines.

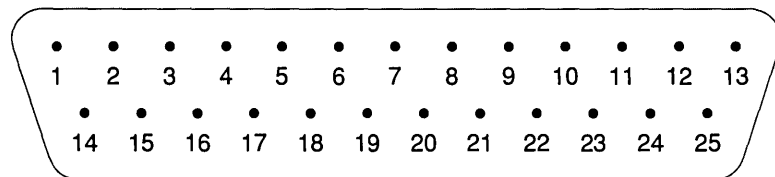
The physical connection is most often made through either a DB-25 (traditional) or DB-9 connector (on some PC adapters). Check out the pinout and line names in Figure 2.

On RS-232A transmit and receive lines (pins 2 and 3; Tx and Rx), a logical zero

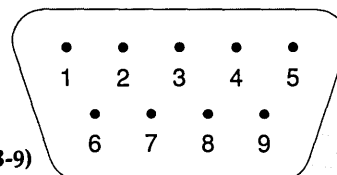
is represented by a signal between +5 and +15 Volts, a logical one is represented by a signal between -5 and -15 volts. (As you might guess, there is also an RS-232B. It uses voltages between +25 and -25V).

It's worth noting that the standard RS-232 driver chip (1488) and receiver chip

Figure 2—RS-232 Pinout (DB-25)



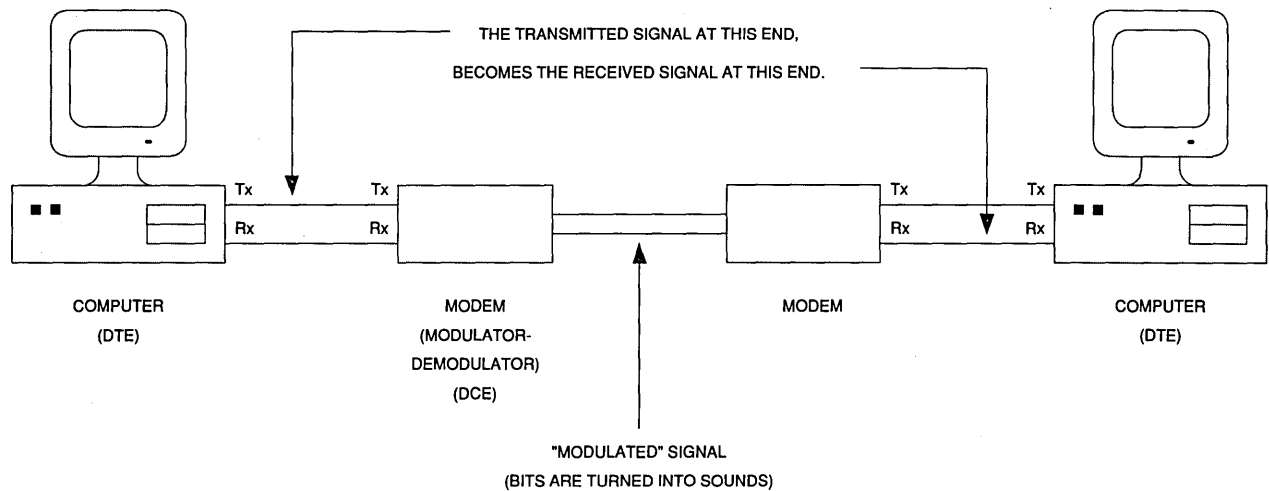
PIN	SIGNAL NAME	PIN	SIGNAL NAME
1	EARTH GROUND	14	SECONDARY TRANSMITTED DATA
2	TRANSMITTED DATA (Tx)	15	TRANSMIT CLOCK
3	RECEIVED DATA (Rx)	16	SECONDARY RECEIVED DATA
4	REQUEST TO SEND	17	RECEIVER CLOCK
5	CLEAR TO SEND	18	UNASSIGNED
6	DATA SET READY	19	SECONDARY REQUEST TO SEND
7	LOGIC GROUND	20	DATA TERMINAL READY
8	CARRIER DETECT	21	SIGNAL QUALITY DETECT
9	RESERVED	22	RING DETECT
10	RESERVED	23	DATA RATE SELECT
11	UNASSIGNED	24	TRANSMIT CLOCK
12	SECONDARY CARRIER DETECT	25	UNASSIGNED
13	SECONDARY CLEAR TO SEND		



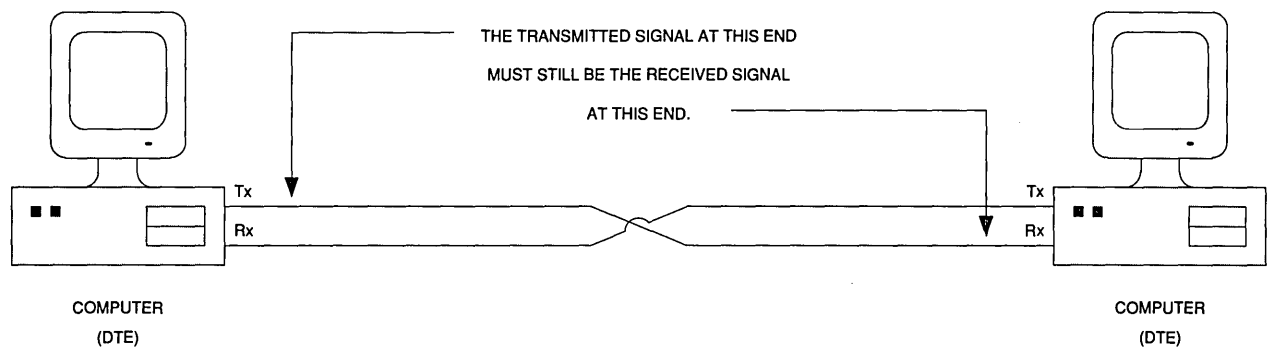
RS-232 Pinout (DB-9)

PIN	SIGNAL NAME	PIN	SIGNAL NAME
1	CARRIER DETECT	6	DATA SET READY
2	RECEIVED DATA	7	REQUEST TO SEND
3	TRANSMITTED DATA	8	CLEAR TO SEND
4	DATA TERMINAL READY	9	RING INDICATE
5	SIGNAL GROUND		

Figure 3—Connecting Devices using RS-232



WHEN YOU CONNECT TWO COMPUTERS DIRECTLY, WITH NO ("NULL") MODEM:



WHEN YOU CONNECT A COMPUTER TO SOMETHING ELSE:

IF THE OTHER DEVICE IS:	USE A:
DCE	STRAIGHT CABLE
DTE	CROSSED CABLE

(1489) don't work reliably at + or -5 volts. (The Commodore 64 contains simple transistor level shifters in place of the 1488 and 1489. The Commodore's RS-232 interface doesn't work with all serial devices.)

The other (handshake and clock) lines on the RS-232 interface use the same voltage levels as the data lines but their sense is backwards. On these lines, a positive voltage represents a logic one and a negative voltage represents a logic zero.

Signal generators assume they'll be driving 3K-7K ohms. Thus they have to supply between 2 and 5 mA. Most line drivers and receivers are capable of supplying between 10 and 20 mA.

The maximum recommended cable length for RS-232C is 50 feet (without repeaters). The maximum standard sig-

naling rate is 20K bits per second (this is why 19.2K baud is a common limit).

Editor's note: In the last couple of years we've seen the maximum rate move to 56K baud (and higher).

The NON-Standard

It's a tribute to human ingenuity to see how many non-modem things people found to hang on to an RS-232C port. Trouble was, as soon as one released the constraint of the modem, there was no reason to use the lines as if they were talking to a modem. Since no standard existed on how to use the lines for non-modem purposes, designers used them as they saw fit, and the RS-232C NON-standard was born.

So, when you connect a device to your computer's RS-232 port, chances are

it won't work. Either your computer or your device (or both) do not use the standard as it was originally intended. The biggie is this: the transmit line of the terminal was meant to be connected to the transmit line of the modem, and the receive line to the modem receive line.

When the terminal became a PC and the modem was replaced by, say, a printer, some bright printer designer realized that the printer was actually receiving the data which the computer was sending, so the TRANSMIT line of the computer should go to the RECEIVE line of the printer, and the TRANSMIT line of the printer should go to the RECEIVE line of the computer.

Data wise, the standard computer-to-modem cable wouldn't work.

Hardware Handshaking

Another insidious line is Clear To Send (CTS). Printers and other devices which are much slower than a computer need some way of saying: "wait, I've received enough, I'll let you know when I need more."

This pause can be handled in hardware or in software.

The hardware method usually uses the CTS line. When the printer pulls this line low (off), the computer is supposed to stop and wait until the printer raises the line.

This is a good idea, and there are no wires to cross (as with Rx and Tx). But there are two problems.

First, there is no guarantee the computer has the hardware to sense the signal (or the sense to sense the signal). Fortunately, PCs do. The only thing you can *always* assume about an RS-232C port is that it will have Tx and Rx.

You can't even assume that pin 1 (frame ground) will work as a signal ground. Pin 7 (signal ground) is the only safe bet.

Second, even if the circuits are in the computer, software must watch for the signal (or the 8250 must generate an interrupt). This isn't always simple, especially if the programmer is using BASIC or isn't an engineer.

In the end, if you're designing a device which you want to interface to *any* computer, it's best to avoid using the CTS line. (In the PC world, CTS is the standard handshake.)

Software Handshaking

The second solution to the "flow control" problem is via software. Two special characters are defined: with XON-XOFF flow control, these are CONTROL-S to stop the flow, and CONTROL-Q to restart it.

When the computer receives the STOP character (via its Rx pin), it stops sending information until it receives the START character. The computer can also send out STOP and START characters to control data coming from a remote device.

The advantage of software flow control is that you make no assumptions about hardware.

Designers also use many of the other lines for hardware handshaking, often with frustrating results.

For a simple-to-use, generic interface, it's best if you use only the Tx, Rx, and ground lines, with XON-XOFF flow control.

Universal RS-232 Cable (Ha Ha)

Figure 4 shows the solution I usually use when I make up an RS-232 cable. It's hard to test whether you need a crossed cable; it's easiest to try both ways and see which one works. To save hassles, I just wire in a double-pole, double-throw (DPDT) switch so I can pop it back and forth to cross and uncross Rx and Tx. The other wiring shown will satisfy any hardware handshaking.

I've used a cable like this to transfer files from one computer to another using a public domain communication program (almost any one will work, as long as the same protocol is used on both ends).

RS-232 On The PC

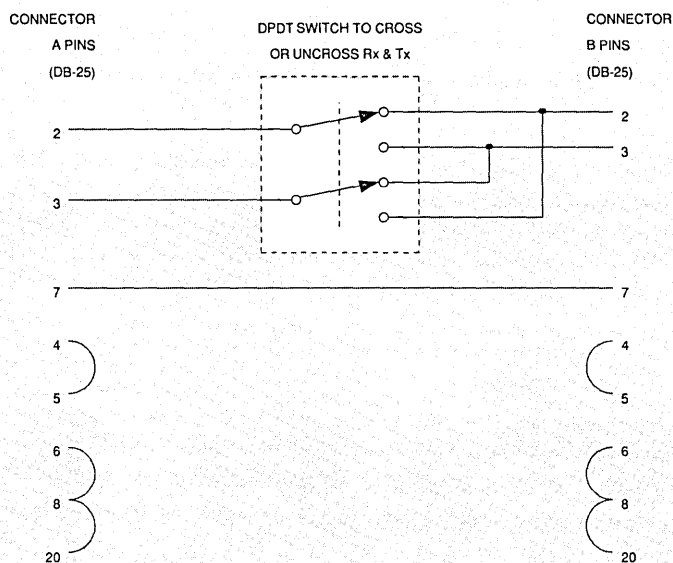
Clones support two serial ports, generally referred to as COM1 and COM2. Each of these ports has its own interrupt.

You may find, when configuring your serial card (via jumpers or dip switches), that you can also make the ports COM3 and COM4. Selecting these options places the ports in an unused I/O area. There is no interrupt or operating system support for COM3 or COM4 and the lack of interrupts severely limits their usefulness.

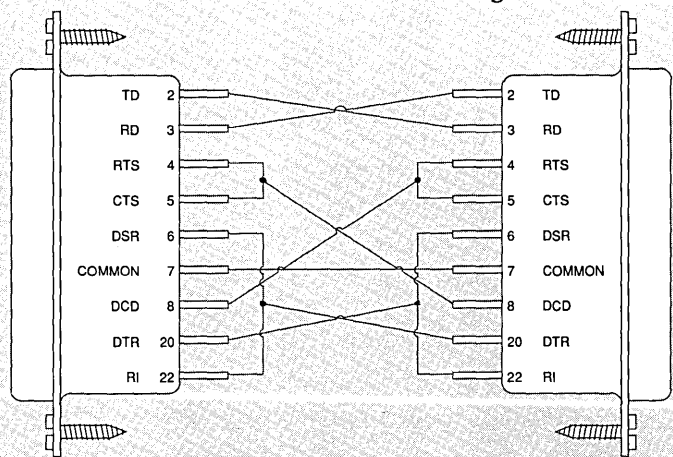
Editor's note: On PCs, COM1 lies at 3F8h - 3FFh (data at 3F8h), it uses interrupt 0Ch. COM2 lies at 2F8h - 2FFh (data at 2F8h), it uses interrupt 0Bh.

Programs access the serial ports by

Figure 4 — Universal RS 232 Cable



RS-232 Cable with Handshaking



CLONE SYSTEMS

(One YEAR guarantee on system)

Turbo Mother Board 4.77 and 10 MHz
 640 K Ram installed on board
 Serial, Parallel, Game Ports
 Clock/Calendar AT Style Keyboard
 Color (CGA) or Monochrome Video Board
 150 Watt Power Supply Flip Top Case
 ABOVE WITH 2 FLOPPY DISK DRIVES \$ 599.00
 WITH 1 FLOPPY AND 20 MEG \$ 859.00
 WITH 2 FLOPPY AND 20 MEG \$ 899.00
 Assembled and Tested for 24 Hours

AT TURBO SYSTEM

AT COMPATIBLE MOTHER BOARD WITH BIOS
 8 MEG AND 12 MEG SWITCHABLE SPEED
 512K RAM INSTALLED UP TO 1024 ON BOARD
 WA2 HARD DISK/FLOPPY DISK CONTROLLER
 MONOCHROME GRAPHICS VIDEO WITH PRINTER
 1.2 MEG OR 360 K FLOPPY
 220 WATT POWER SUPPLY AT CASE
 AT KEYBOARD SET UP DISK
 ONE YEAR WARRANTY ON SYSTEM \$1195.00

EGA UPGRADE FOR ABOVE \$ 75.00
 512K UPGRADE (1024 INSTALLED) \$ 130.00
 5339 KEYBOARD UPGRADE \$ 30.00

HARD DRIVES FOR XT AND AT
 ST-225 KIT FOR XT (20 MEG) \$ 259.00
 ST-238 KIT FOR XT (RLL 30 MEG) \$ 279.00
 ST-251 FOR AT (40 MEG) \$ 405.00

MONITORS

Color Monitor RGB (CGA) \$ 255.00
 Color Monitor RGB (EGA) \$ 355.00
 Monochrome TTL (Green) \$ 95.00
 Monochrome TTL (Amber) \$ 95.00
 EGA Color Video Card \$ 129.00

CITIZEN PRINTERS

MODEL 120D 120 CPS 9" \$ 179.00
 MODEL 180D 180 CPS 9" \$ 199.00
 MODEL MSP-15E 160 CPS 15" \$ 359.00
 MODEL MSP-40 240 CPS 9" \$ 319.00
 MODEL MSP-45 240 CPS 15" \$ 439.00
 MODEL MSP-50 300 CPS 9" \$ 419.00
 MODEL MSP-55 300 CPS 15" \$ 499.00

CASCADE ELECTRONICS, INC.

ROUTE 1 BOX 8
 RANDOLPH, MN 55065
 507-645-7997

Please ADD Shipping on all Orders
 COD Add \$3.00 Credit Cards ADD 5%
 Limited to Stock on Hand Subject to change

Reader Service Number 15

reading and writing locations in the I/O space. The PC can access, via the microprocessor's address lines, memory cells or I/O (input/output) cells. I/O addressing has different timing and handshaking than memory addressing. I/O devices generally respond more slowly than memory (as far as I can recall).

Not all microprocessors do this. The Motorola 68000 family, used in Macs, Ataris, Amigas, and many Unix machines, puts all of its I/O in the memory space (this is called "memory-mapped I/O"). The 68000 has special handshake lines for slow memory or peripherals.

The status pins on the 80X86 or 8088 microprocessor tell the rest of the system whether the next access will be memory or I/O.

High-level languages also have some method of specifying I/O. BASIC uses the IN and OUT commands. Turbo Pascal uses the port array. Turbo Prolog uses the portbyte predicate. Microsoft C uses inp() and outp().

Turbo C uses inport()/outport() for word I/O which allows an I/O instruction to use the maximum word size of the processor's data bus (8 bits on the 8088, 16 bits on the 8086/80286, and 32 bits on the 80386) or inportb()/outportb() for byte I/O.

The 8250 chip occupies a block of eight addresses in the I/O space. Switches on the serial card determine the base (starting) I/O address.

Well-behaved programs find out which I/O base address is supposed to be COM1 and which is COM2 by reading the BIOS data area. Base:offset 0040:0000 holds COM1 and 0040:0002 holds COM2. This way you can switch the ports without rewiring; simply change the data. Many programs don't bother reading this data, however, so it isn't very useful.

Communication Protocols

When a communication program attempts to transfer a large amount of data (for instance, a file) over a serial link, it runs a risk.

It's not unusual for noise or other problems to change bits as they sail across the line. For file transfer to be dependable, the transfer program must be able to (1) determine whether there's an error and if so, (2) correct it.

Most file transfer protocols (FTP's) break files up into blocks or packets. This way, you don't need to retransmit an entire file if there's an error.

Data is usually surrounded by infor-

mation which says, "a packet's coming, this is it's number, here's the data." Finally, some sort of check value is sent.

One of the earliest microcomputer protocols is Ward Christensen's XMODEM. XMODEM divides a file into packets containing (1) the SOH (Start-Of-Header) character (an ASCII 01) to indicate the beginning of a packet, (2) the sequence number of the packet, both normal and in one's complement, (3) 128 bytes of file data, and (4) one byte of checksum or the checkvalue, which is calculated by simply adding all the data bytes together as they come in (throwing away the carries).

Although by no means perfect, XMODEM is probably the most commonly used protocol. An early flaw (a checksum can miss errors) has been rectified with the addition of a cyclic-redundancy check (CRC) for the checkvalue. CRCs are more complicated but much more reliable. But you must often specify XMODEM checksum or XMODEM CRC.

The stand-alone XMODEM program only transfers one file at a time. This was improved in the MODEM7 program, which will move groups of files. MODEM7 usually generates either checksum or CRC automatically.

The YMODEM transfer has its own protocol, but will use XMODEM if that's what the other end is using. ZMODEM, WXMDEM, and others keep popping up. Usually, what's best is whatever the other machine uses.

I've looked at a number of public domain/shareware communications programs, and I think my favorite is Shareware's Procomm—it's mature, it seems bug-free, it supports most popular file-transfer protocols, it emulates a VT100 terminal (convenient for connecting to minis and mainframes), and it's easy to use. You can find it on most BBS's and on *Micro C* disk #38.

Muppet Protocol

The only problem with most of these protocols is they were developed on microcomputers, so many of them can only be found on microcomputers (although XMODEM is often found on UNIX systems). Columbia University set out to smash the communications protocol bug once and for all with KERMIT (the name of a famous frog, that's all).

KERMIT is designed to work on *all* computers without regard to size. This was an ambitious project, but it seems to work; and it helps that all the source code is public domain.

KERMIT negotiates with the other end to establish the communication parameters. This is clever since it allows extensions to the protocol without disturbing existing installations.

Two extensions have already been made to speed KERMIT up—but it still assumes computers can only handle printable ASCII characters. This is one of the things which make it portable, but it runs like a turtle (not a frog) compared to XMODEM. As a rule of thumb, use XMODEM whenever you can, and KERMIT when you can't.

Someday

There's a lot more to writing a communication program than I can cover here (check out Joe Campbell's superb *C Programmer's Guide to Serial Communications*, Howard W. Sams & Company, 1987), but I plan to do some kind of experiment with it in the coming months. Stay tuned.



Rafting
Barbeque
Computer tech talk

SOG VII

July 14 - July 16

Bend, Oregon

Leave city life behind!



Write A Data Base Program
(end to end)
in 10 minutes in TPascal 4.0

using
TURBO
Programmer

(formerly Turbo GhostWriter)

Perfect for creating complex business applications!

Ideal for BASIC programmers who find TPascal too tough!

Spec your customer in the morning - Show a demo in the afternoon!

Features

- screen editor and painter
- automatic programmer documentation
- automatic data checking/validation
- plenty of "hooks" for customizing
- unlimited technical support

More features

- automatic B-tree indexing
- consistent user interface
- automatic context-sensitive help
- relational model to show customizing
- 30 day money-back guarantee (less \$14 shipping/handling)

No questions to answer. Just draw your screens the way you want them to appear, tell Turbo Programmer how to set up the indexes and that's it... Running 4.0 code in 6 seconds with no programming. Regular price \$450.

Now Only \$289 Orders & Info 800 227-7681
ASCII 3239 Mill Run, Raleigh, NC 27612-4135

Reader Service Number 48

Intel's 8088

A Tour Through The Brains Of The PC

Larry's finally done it. He's traversed his way through the primary players (we're talking 40 pins here) in the IBM XT and its clones. He's finishing up with the kingpin, the heart and soul, the chip without whom there would be no others... (Do I hear a bull.... from the crowd?)

Back in *Micro Cornucopia* Issue #35, I began this series on the PC's innards with a look at the 8253 programmable interval timer. This article caps the series by covering the iAPX 88/10 (or 8088 to the less snooty among us).

I'll also talk briefly about some of the differences between the 8088 and other members of the 8086 family of CPUs—NEC's V20 included. We'll finish up by following a keystroke from keyboard to screen to see how all these smart chips conspire to perform something simple.

All In The Family

In 1978 Intel introduced the father of the 8086 family, named (appropriately enough) the 8086. The following year saw the first of the soon to be ubiquitous 8088s. The two processors were nearly identical—the main difference being the size of their data busses. The 8086 transferred 16 bits of data at a time while the 8088 could deal with only 8 bits.

However, object code compatibility between the two processors was complete and IBM's choice of the 8088 for use in their PC assured Intel of a huge market.

The 8086 and 8088 were followed by the 80186, 80188, 80286, and, most recently, the 80386 CPU—all compatible with the 8086. That means the whole family can execute 8086 code.

The 186 (16-bit data I/O) and 188 (8-bit) added on-board clock, direct memory access (DMA), interrupt control, counter/timer, chip select logic, and

more. Inside that one piece of silicon lay most of the smart chips I've discussed over the past year. Intel also added a number of new instructions.

The 286 took the 8086 family in another direction with the capability to create multi-user and multi-tasking systems. It could generate a 24-bit address (16 MBytes) though it remained a 16-bit processor.

A full 32 bits of address and data and 32-bit internal architecture make the 386 a true 32-bit processor. As such, it *could* deal with its entire address space (four gigabytes) as one contiguous block. No more segments! However, it's quite happy in a segmented configuration like Intel's earlier processors.

The 8088 remains the most prevalent member of the family and we'll be concentrating on it. But some of the discussion will apply, at least loosely, to the other processors.

The 8088

Intel divides the 8088 into two major sections: the execution unit (EU) and the bus interface unit (BIU). These units operate independently with the EU executing instructions while the BIU prefetches the next instructions and transfers data to and from memory and I/O space.

In rough terms, the EU thinks and the BIU talks. The EU is almost completely isolated from the rest of the system. When the 8088 executes a program, the BIU prefetches instructions from memory and loads them into the 8088's 4-byte instruction queue. The EU fetches instructions from the queue and executes them.

During execution of an instruction, the BIU happily prefetches more instructions to fill up the queue. If the current instruction requires port or memory I/O, the BIU finishes up with its prefetch (if any) and does the I/O. It then resumes instruction prefetching (if needed) and the EU

goes back to thinking.

8088/8086 Registers

Both processors make use of the same 14 two-byte registers. The four data registers (AX, BX, CX, and DX) can be accessed in two ways: as a single 16-bit register (AX), or as two separate 8-bit registers (AH and AL). "H" and "L" refer to the high and low bytes in the AX register.

Although you can use most of the registers for several purposes, some registers *must* be used with certain instructions. For example, port I/O requires use of AX for words and AL for bytes.

The four pointer and index registers can only be accessed as words (16 bits), but they have most of the general purpose capabilities of the data registers. While they can be used for math and logic operations, they usually serve other purposes.

The stack pointer (SP) and base pointer (BP) support stack functions. String functions often use the source index (SI) and destination index (DI) for manipulating strings.

When the 8088 needs to find the next instruction of a program, it uses the instruction pointer (IP). A programmer can't get at the IP directly—probably a good thing. The flag register holds information on the status of arithmetic and logic operations, as well as control flags.

And the code segment (CS), data segment (DS), stack segment (SS), and extra segment (ES) registers provide part of the addressing information.

There are, in addition, several "temporary" and "internal communication" registers inaccessible to the outside world. But I've never seen these documented very well so I can't tell you anything about them.

Execution Unit

The EU includes the general purpose

registers, flags, and arithmetic/logic unit (ALU).

The flags live in a single 16-bit register. (See Figure 1.) Six flags contain operation status information—did the operation result in a carry or a borrow, did the result overflow the capacity of its destination, was the result positive or negative, what was the parity of the result, was the result zero?

The other three flags perform the 8088 control functions. Setting the direction flag makes string instructions process strings from high to low addresses. Resetting the flag reverses the processing direction.

Figure 1 - 8088 Flags Register

Bit	Function
0	carry
1	unused
2	parity
3	unused
4	auxiliary carry
5	unused
6	zero
7	sign
8	trap
9	interrupt enable
10	direction
11	overflow
12-15	unused

With the interrupt enable flag set, the 8088 recognizes external interrupt requests. On the PC, external interrupts come in from the 8259 interrupt controller and can be initiated by a real time clock tick, a keypress, or some other hardware event. Resetting the interrupt flag disables these interrupts. The CLI and STI instructions take care of clearing (resetting) and setting this flag.

Finally, the trap flag causes the 8088 to generate an interrupt after execution of each instruction. This flag allows a

debugger to single step through your code in search of those nasty glitches. (See *Micro C's* Issue #40 C Column for more on debuggers.)

The Real Brains

If you want to isolate one part of the 8088 that really has some smarts, it's the ALU. Everything else supports this one section. The ALU performs all arithmetic operations and logical comparisons.

Most of us like to think of assembly programming as the bottom line. But how does the ALU know what to do with an instruction?

Microcode.

Microcode isn't little, tiny code written by stunted programmers. The folks at Intel use microcode to translate the complex set of object instructions into the much smaller set the hardware understands.

Bus Interface Unit

The rest of the registers (segment registers and the instruction pointer), a 20-bit address generator, the instruction queue, and bus control logic live in the BIU.

The BIU has three main tasks. First, it prefetches instructions from memory in an attempt to always have instructions waiting in the instruction queue. Execution speed increases dramatically if the EU doesn't have to wait while its next instruction is fetched.

Ideally, there will always be something waiting in the queue for the EU to digest. No such luck. Two factors decrease the efficiency of the queue. A few fast instructions in a row can empty the queue, but this doesn't happen all that often.

The more common queue problem occurs when a program calls for any kind of branch. The branch could be a procedure call, a return from a procedure, a loop instruction, an interrupt, an inter-

rupt return, or any of the jump instructions.

In each of these situations, sequential prefetch breaks down. The next instruction in memory is not the next instruction to be executed. So the queue gets dumped, the BIU begins to refill it starting at the jump destination, and we've lost some of the queue's benefit. You can see that code which jumps around a lot will be slower than sequential code.

The queue and its maintenance illustrate one of the differences between the 8088 and 8086 processors. The 8088 contains a 4-byte queue which the BIU tries to refill as soon as a single byte has been fetched by the EU. Since the 8088 has an 8-bit data path, it prefetches one byte at a time.

The 8086, on the other hand, has a 6-byte queue. And, since it does 16-bit data I/O, and can therefore prefetch two bytes at a time, the 8086 waits until two bytes of the queue have been fetched by the EU before attempting to prefetch another two bytes from memory.

I/O

The BIU also handles I/O chores for the EU. Whenever an instruction calls for either memory access or port I/O, the EU requests a bus cycle from the BIU. A bus cycle usually consists of four clock cycles, or T-states (unless we're talking to a slow device). More on this later. For now, let's just be aware that any time the EU needs to talk to the outside world, it does so through the BIU.

As it turns out, the BIU ends up spending some of its time idling. This can happen when a coprocessor (perhaps an 8089 I/O coprocessor or the 8087 math chip) takes over the bus, or when the EU has to grind through a complicated instruction.

It only takes four bus cycles, or 16 T-states, to completely fill the queue—assuming no interruptions by the EU for

I/O. (Why does this make me want to say E-I-E-I/O?) But some of the more complicated instructions take well over 150 T-states to execute. So the BIU gets a breather and usually ends up with something waiting in the queue for the EU.

The third task of the BIU has to do with ...

Segmented Memory

"Ooooh, he said those nasty words."

The main complaint most folks have against Intel processors is their use of a segmented memory addressing scheme. Using only its 16-bit registers, the 8088 could directly address only 64 KBytes of memory. We need four more bits for a complete 20-bit address into 1 MByte of RAM.

The BIU contains a dedicated adder which constructs the 20-bit address. It does this by combining the contents of one of the 16-bit segment registers with a 16-bit offset. Think of the segment address as the beginning of a block of

(See *C'ing Clearly* in this issue for more on memory addressing.)

Segmented memory does add to programmers' confusion. But it's really not *that* much of a hassle and it gives us all an excuse for Intel-bashing.

8088 Pinout

The 8088 lives in a standard 40-pin DIP package. (See Figure 2.) Nothing mysterious about A8 through A15. These pins provide eight bits of address which are valid throughout the entire bus cycle.

The rest of the address lines share their space with either data (AD0 through AD7) or status (A16/S3 through A19/S6) signals.

"Time multiplexing" allows this sharing of a pin (and a bus) by two signals. Let's look at AD0.

During the first (T₁) of the four or more T-states in a bus cycle, AD0 holds A0. T₂ gives time to set the direction of the transfer, and the BIU reads or writes a valid D0 for the rest of the bus cycle.

of T₃. If it sees a low on READY, the BIU inserts wait states (T_w) between T₃ and T₄ until the I/O device signals its readiness by releasing the READY line.

Bus Control

How does the rest of the system know whether there's address or data on the multiplexed bus? An 8288 bus controller lets everyone know.

Throughout the bus cycle the processor sends information to the 8288 via S0,

Figure 3 - Status Line Decoding

S0	S1	S2	8088 Status
0	0	0	Int acknowledge
1	0	0	I/O port read
0	1	0	I/O port write
0	0	1	Code access
1	1	0	Halt
1	0	1	Memory read
0	1	1	Memory write
1	1	1	Passive

S1 and S2. The 8288 decodes these status lines according to Figure 3 and generates control signals for the rest of the system.

For example, the following sequence would write a byte to memory. During T₁ the 8088 outputs a full 20 bits of address to the system. At the same time, it drives S0 low and leaves S1 and S2 high. The 8288 decodes this combination of status signals and generates an address latch enable (ALE).

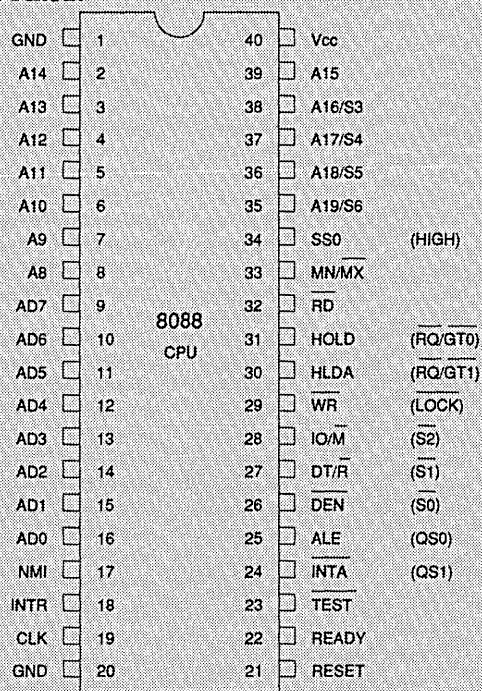
An LS373 takes care of the demultiplexing chores for A0 through A7. It responds to ALE by latching (storing) the lower eight bits of address. Another LS373 latches the upper eight bits of address in response to the same signal.

One more main board signal effects address generation, the address enable (AEN). Several system states can restrict access to the bus. Lock, hold, and wait conditions all hold AEN high. When any of these conditions occur, AEN keeps the 8288 from issuing any signals. Then, when the bus again becomes available, the system sends a low out on AEN, enabling the 8288's command outputs.

AEN also enables the output of both LS373s before ALE triggers the address latching. In addition, it latches A8 through A11 into an LS244. So now we have all 20 bits of address safely out on the address bus.

At the end of the ALE pulse, a new combination of outputs on the 8088's status lines brings another 8288 signal

Figure 2 —8088 Pinout



MAXIMUM MODE PIN FUNCTIONS (e.g., QS1) ARE SHOWN IN PARENTHESES.

memory and the offset as an index into that block.

We get 20 bits by shifting the segment address four bits to the left. Adding the resulting 20-bit number to the offset gives an address within the 1 MByte space.

If the I/O deals with a slow device, the device may pull the 8088's READY line low, telling the processor that the device hasn't had time to place (or read) data on the bus.

The 8088 looks at READY at the end

If You Don't Have WindowDOS 2.0, You're Wasting Time!!

"When Baba Ram Dass said "Be here now, remember," designers of hard disk utilities should have paid heed. A powerful manager like XTREE can track files and subdirectories and execute DOS commands, but it isn't memory resident. Handy pop-up DOS commanders like PopDOS may be here now, but they lack the power of a full-fledged disk manager. After much meditation, the developers of WindowDOS 2.0 have come up with the best answer yet to the guru's paradox.

Until now, the closest thing to a real RAM-resident disk manager was version 1.0 of WindowDOS. It offered a full-screen pop-up menu and could rename, copy, and delete files. But it couldn't move files, format disks, or rename subdirectories—which XTREE can. Now version 2.0 is here, and it's a winner. Its RAM resident (using less than 50K) but offers all the power of a nonresident disk manager."

—Patrick Marshall, WindowDOS 2.0 Product Review, PC World, May, 1987

Once you've experienced the convenience of instant access to DOS commands, you'll never be satisfied with returning to DOS to list files, format disks, or copy, rename, or erase files. Nor will you be happy with a DOS shell, because shell programs are just as inaccessible as DOS when you are using an application program. **Only one program combines memory-residency with the power of a full-featured disk manager: WindowDOS Version 2.0.**

Features Not Found In DOS

- ◆ Sort directories in 8 ways--or not at all
- ◆ Copy, erase, and move groups of files
- ◆ Find any file in seconds
- ◆ Display default directory of any drive with a single keystroke
- ◆ Display graphic tree
- ◆ Global copy & erase commands
- ◆ Copy function prompts you to insert another disk when necessary
- ◆ Display hidden files and subdirectories
- ◆ Display file contents in various formats and page forward/backward
- ◆ Display Wordstar files in readable format
- ◆ Unique RAM Environment function shows name, size, location, and interrupts of every program in memory
- ◆ Rename subdirectories for instant reorganization
- ◆ Hide and unhide subdirectories
- ◆ See and change file attributes
- ◆ Send control codes to printer
- ◆ Switch default printer
- ◆ Password "lock" your system
- ◆ Set AT Real-Time Clock
- ◆ 5-minute screen-blanking function
- ◆ Input response macros

Enhances These Functions

- ◆ Format disks (faster than DOS)
- ◆ Make and erase subdirectories
- ◆ Copy, rename, and erase files
- ◆ Copy files to printer or COM ports
- ◆ Display disk free space and other media information
- ◆ Check and set the time and date

Benefits

- ◆ **Saves Time**—No waiting to exit or reload programs. Instant access to DOS functions whatever your current task. Easily saves 10 or more minutes a day.
- ◆ **Comprehensive**—Broad range of commands, including many not supplied by DOS. Satisfies the needs of both new and advanced users.
- ◆ **Simplifies DOS**—No need to remember exact DOS commands. Intuitive interface and "point and shoot" design saves keystrokes and prevents mistakes. Group file "tagging" avoids the drudgery of repetitive commands.
- ◆ **Security**—Capability to hide/unhide subdirectories, password "lock" a computer, and check for unwanted programs in RAM helps secure data and prevent unauthorized access.

WindowDOS 2.0 Addresses "RAM Cram" Like No Other Program!!!

1. Designed specifically to be loaded first, unlike most memory-resident programs that insist on being loaded last.
2. Uses a hot key combination that does not have an associated ASCII value—prevents key conflicts with other programs.
3. Unique RAM Environment function lets you monitor the locations, memory costs, and interactions of all programs in memory, including the currently running program. Great for power users/developers.

Other Information

- ◆ Not copy protected
- ◆ Uses only 51K of memory
- ◆ Supports EGA & Hercules
- ◆ Runs memory-resident or as a stand alone program
- ◆ Uninstall command
- ◆ PC/XT/AT/100% Compatibles
- ◆ **Order Today--Only \$49.95**

WindowDOS Associates • Box 300488-C • Arlington, Tx 76010 • 817-467-4103

into play. Data transmit/receive (DT/R) controls the direction of an LS245 octal bus transceiver.

The LS245 provides the multiplexing/demultiplexing and I/O functions between the 8088 and the data bus. The 8288 is now sending a high on DT/R telling the LS245 that this will be a write operation, and we're into T₂.

We have an address and we know we'll be writing, but where and what? At the beginning of T₂, S0-S2 cause the 8288 to send a memory write (MEMW) command. And, finally, the processor initiates a data enable (DEN), again through the 8288. DEN enables the LS245 and D0 through D7 chug out onto the data bus.

For most of the rest of the bus cycle we have a valid address, valid data, and an active MEMW. Voila, a complete memory write.

More Pins

On the 8088, A16 through A19 are multiplexed with four more status signals. The PC ignores S3 through S6, but they deserve at least a mention. S3 and S4 provide information on which segment register is currently in use for data access. S5 shows the status of the interrupt enable flag and S6 does nothing.

Situations arise where a programmer needs to ensure that the bus won't be grabbed by another processor in the system. For example, in a multiprocessor system, several processors may share common resources—data, buffers, perhaps a printer. In order to avoid contention, each processor checks a "semaphore" before accessing the resource.

However, since the exchange (XCHG) instruction used to inspect and update a semaphore does two memory accesses, it would be possible for another processor to horn in between the two accesses and wreak havoc. To guard against this problem, we lock other processors out of the bus.

An assembly instruction prefix of "lock" forces the 8088 to pull its LOCK output low for the duration of the instruction. LOCK can then be used by the system to restrict bus access during semaphore updates. But the PC doesn't need semaphores and uses LOCK in a different manner.

During a locked instruction, the PC refuses to acknowledge hold requests from the DMA controller. I'm not sure what purpose this serves. If anyone has made use of the lock prefix in 8088 assembly code, I'd like to hear from you.

8087 Coordination

The queue status outputs (QS0 and QS1) provide information about use of the instruction queue. (See Figure 4.) If your system has an 8087, it uses this information to help route instructions to the coprocessor.

Two request/grant pins (RQ/GT0 and RQ/GT1) let coprocessors request use of the bus. The PC implements only the second of these for communication between the 8088 and the 8087. Both the request and grant are active low pulses. When the 8087 needs the bus, it yanks RQ/GT1 low for one clock cycle.

Assuming a LOCK condition doesn't exist, we're not in the middle of accessing

enable flag is set, the interrupt sequence begins. If not, the 8088 continues executing code until interrupts are reenabled.

Nonmaskable interrupts (NMI) are caused by any of three error conditions: memory parity errors, I/O bus errors, or interrupts from the 8087. In each case, the processor automatically executes interrupt 2, the NMI handler. This ROM code tries to locate the source of error, prints a semi-informative message on the screen, and halts the system.

Actually, "nonmaskable" is a lie. System configuration switches can be set to disable both parity and I/O error messages. Or you can totally disable NMIs by resetting bit 7 of the NMI mask

Figure 4 - Instruction Queue Status

<u>QS0</u>	<u>QS1</u>	<u>Status</u>
0	0	Nothing removed from the queue
0	1	First byte of an instruction removed
1	1	Subsequent byte of instruction removed

a word of memory, and we're not at the beginning of an interrupt acknowledge sequence, the 8088 finishes its current bus cycle (if any) and responds with another one clock pulse on RQ/GT1. This grant pulse tells the 8087 that the 8088 has released the bus.

When the 8087 finishes using the bus, it sends a one-clock pulse to the 8088 telling the processor that it can reclaim the bus on the next clock cycle. During the 8087's bus activity, the BIU naps. It goes into an inactive state (T_i). When it wakes up, it goes to T₄ and we're off and running again.

Another important pin in 8087 systems is the TEST pin. The 8087 and 8088 can operate concurrently. But often we want the 8088 to wait for the results of an 8087 operation before continuing. The assembly instruction "wait" forces the 8088 into an idling condition. It idles until it sees a low on TEST. This pin connects directly to the 8087's BUSY output. So, as soon as the 8087's done, the 8088 can continue.

"Wait" instructions also show up before most 8087 instructions. This guarantees that the 8087 will be finished with its last task before getting a new one.

Interrupts, Etc.

A high on the interrupt request (IREQ) line during T₄ means a hardware interrupt needs service. If the interrupt

register (port 0A0h) to zero. (Nonmaskable interrupt mask sounds government inspired—like the Department of Redundancy Department.)

To reset the 8088 the RESET pin must be held high for at least four clock cycles. After four clocks a falling RESET signal triggers the reset sequence. During the space of about seven clock cycles, the 8088 zeros the DS, ES, SS, and IP registers and sets the CS register to 0ffffh.

With reset accomplished, the 8088 comes back to life and begins executing code at address 0ffff0h. In the PC this location lies at the end of the ROM BIOS. It holds a jump to RESET (the routine which does power on self test) and we're back in action (we hope).

Finally, the 8088 needs a heartbeat and some power. The heartbeat comes in the form of an asymmetric square wave from the 8284 clock generator. The two-thirds low, one-third high nature of CLK optimizes the 8088's internal timing. And the power is a healthy 2.5 Watts of 5 Volts DC (at 0.5 Amps).

Minimum/Maximum Modes

This discussion has dealt only with the 8088 in a maximum configuration. The MN/MX input alters the functions of nine processor pins and provides two modes of operation for the 8088.

Maximum means a large, complex configuration where the 8088 needs help with system control. It also needs to be

able to handle coprocessors.

In a minimum configuration, the 8088 generates its own bus control signals instead of using S0-S2 as discussed above. These extra control signals take the place of the processor status, queue status, lock, and request/grant lines necessary for coprocessor control. So a system using the minimum configuration, like the PCjr, cannot have coprocessors.

The V20

A few years back, NEC came out with an 8088 replacement called the V20. The chip has proved to be very popular for a couple of reasons.

First, it executes the 8080 instruction set. Quite exciting for the CP/M world at first, but interest in this feature seems to have dwindled. And for good reason: why cripple your clone with foreign software when an abundance of good software exists that takes advantage of the native system.

A better reason exists for using the V20. It's fast. Fast because of:

- Use of two data busses in the EU.
- Faster address generation in the BIU.
- Two extra 16-bit registers for multiplication/division and shift/rotate instructions.
- Special prefetch pointer. This pointer shows the location to be accessed next and reduces the loss of time due to jumps in code.
- New instructions for bit manipulation and BCD operations. Interesting, but not that useful. I doubt that many folks take advantage of these because the software would be incompatible with the 8088.

Use of CMOS technology gives substantial power savings over the 8088. The V20 uses 0.5 Watts to the 8088's 2.5. The V20 also enters a low power standby mode whenever it sees a halt instruction.

Tying It All Together

Over the past year we've talked about the PC's five smart chips, one at a time. But the PC is a system. So let's try to make some sense of the interrelationships which make the PC function. We'll keep it simple and just watch how a keystroke ends up as a character on the screen.

Assuming an idling state with the cursor hanging out next to the DOS prompt, a keypress leads to the following events. First, the scan code generated by the keypress gets loaded into port A of the

programmable peripheral interface (PPI) and an interrupt request 1 goes out to the interrupt controller (PIC).

If there aren't any higher priority interrupts waiting to be serviced, the PIC raises the 8088's INT pin requesting interrupt service for the keystroke. If the 8088's interrupt flag is set (enabling interrupts), it finishes its current task and pulls all of its status lines low.

This causes the 8288 bus controller to generate an interrupt acknowledge for the PIC. When the PIC sees that it has the processor's attention, it loads an 8-bit pointer onto the data bus. The 8088 reads this pointer and uses it to index into a table of 256 4-byte vectors in the bottom of the PC's address space.

These vectors point to the interrupt handler code for each interrupt. For the keyboard interrupt, the 8088 gets a pointer of 9 from the PIC. So the 8088 jumps to the address contained in the ninth 4-byte vector. If no memory resident programs are out there gobbling up keystrokes, the jump will be into the ROM BIOS. Otherwise some protected area of RAM will contain the interrupt handler.

In either case, the 8088 now processes the keystroke according to the code in the interrupt handler and returns to whatever it was doing before the keystroke.

Interrupted Interrupts

The 8088 won't be able to deal with the keystroke without interruption. It has to contend with memory refresh, real time clock ticks, plus any memory resident code that's watching the clock.

Every 15 microseconds the DMA controller sees a pulse from counter 1 of the 8253 timer chip on its highest priority DMA request input. The DMA controller requests a hold from the 8088 which finishes up with its current task, grants the hold, and sits back while the DMA controller refreshes part of RAM.

The other periodic interrupt also originates in the 8253. Counter 0 gooses the PIC's highest priority interrupt request line about 18.2 times a second. These "ticks" cause the 8088 to execute the time of day interrupt handler and keep the time updated.

The End

Well, that's it. We've looked at all the smarts in the PC.

There's an amazing amount going on in the 8088's little head. Although we've gone through a lot of detail, other levels of obscurity exist below this one. And

there are plenty of topics we didn't even discuss. How about the instruction set? And what does microcode really look like?

I could easily have filled this whole issue but Dave (and our advertisers) would have me strung up. If you're hankerin' for more, take a wade through some of the following.

Sources

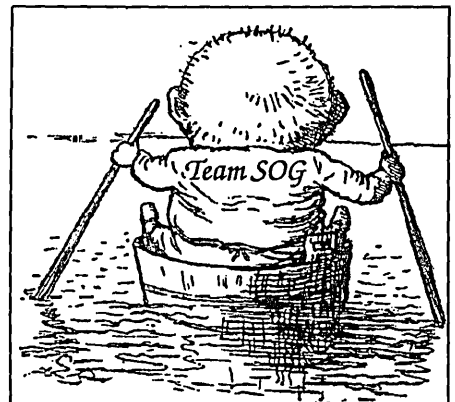
This series in *Micro Cornucopia* includes issues:

- #35—8253 Counter/Timer
- #36—8259 Interrupt Controller
- #37—8237 DMA Controller
- #40—8255 Programmable Peripheral Interface

And a bit more light reading:

- IBM's *Technical Reference* for the PC-XT.
- Intel's *8086 Family User's Manual*, *Component Data Catalog*, and *Microsystem Components Handbook*.
- Murray Sargent and Richard L. Shoemaker's *The IBM Personal Computer From the Inside Out*.
- David C. Willen and Jeffrey I. Krantz's *8088 Assembler Language Programming: The IBM PC*.

◆ ◆ ◆



*Rafting
Barbeque
Computer tech talk
SOG VII
July 14 - July 16
Bend, Oregon
See you there!!*

How Much Information Does A Message Contain?

An Introduction To Information Theory

Who the heck cares about information theory? If we wanted theoretical stuff, we'd call ourselves Micro Theoretician. Micro C is for practical solutions to real problems: like the following. Enjoy.

The mathematical theory of communication developed by Claude E. Shannon in 1948 is regarded as one of the intellectual landmarks of this century. Like many other great theories (e.g., Newton's Three Laws Of Motion or Einstein's principle that no material body can travel faster than light), it's based on seemingly simple but profound insights.

At the heart of Shannon's theory is the idea that in all communication processes, messages are sent to resolve *uncertainty*. Each message (received) is one out of a set of possible messages which *might* have been sent. Let's look in detail at the implications of this insight.

Uncertainty

Consider a situation where any of several outcomes have the same possibility. For example, take a horse race with no favorite, no dark horse, and an equal number of sorry nags, each with an identical chance of losing.

We "feel" intuitively that the larger the number of possibilities, the greater the uncertainty of the outcome. That the uncertainty of a tossup race of 32 horses is greater than the uncertainty of a tossup race of only 8 horses.

Let's examine this "general case" (where the alternative possibilities are all identical) in some detail. Afterwards, we'll explore the modifications we need to extend our analysis to cases where some outcomes are more probable than others.

So What's In A Message?

When we receive a message (or a confirmation that one of the possible messages has materialized—let's say at a modem), we can say we've received some information.

It also seems reasonable to say that the higher the uncertainty before we got the message, the higher the "quantity of information" that has been conveyed by the message. In other words, we can measure the amount of information by the amount of uncertainty resolved by the message.

But how do we measure it?

Take the extreme case where there's only one possibility. The outcome is certain, so the uncertainty is zero. And any message confirming this already certain outcome conveys absolutely zero information. But this still doesn't let us measure uncertainty precisely. Recall the race: how much more uncertainty is there in 32 than in 8 outcomes?

If you guessed 4 times as much, you guessed wrong. Is there 8 times as much uncertainty with 8 outcomes as with 1 outcome? No. With 1 outcome, the uncertainty is 0, and clearly it's unsatisfactory to say that with 8 outcomes, the uncertainty is 8 times 0, or 0 as well.

Let's reconsider the race. Instead of a message telling us which of the 32 horses has won, suppose we group the horses into four groups: A, B, C, and D. We then get a first message telling us which of the groups the winner is in, followed by a second message telling us which of the 8 horses in that group is the winner.

The first message resolves our uncertainty among 4 possible outcomes, and the second resolves our uncertainty among 8 possible outcomes. We could say that the information we've received is the sum of the uncertainty associated with 4 outcomes *plus* the uncertainty associated with 8 outcomes.

But we could also say that the total in-

formation we've received is the resolution of uncertainty associated with 32 outcomes. If we use the symbol $H(N)$ for the amount of uncertainty associated with N outcomes, we could describe the race uncertainty this way—

$$H(32) = H(4) + H(8)$$

This argument is perfectly general, so we can say that—

$$H(m * n) = H(m) + H(n)$$

where m and n can be any numbers.

In other words, the quantity of information conveyed by messages is additive.

Logarithms & Uncertainty

Shannon showed that the only mathematical function which has this additive property is the logarithm. (The logarithm of any number A to a particular base B is the power to which the base B should be raised to get the number A .)

For example, since $1000 = 10$ raised to the power 3, the logarithm of 1000 to the base 10 is 3; and similarly since 32 is 2 raised to the power 5, the logarithm of 32 to the base 2 is 5, and so on.

Two other important properties of logarithms are—

- (1) the logarithm of 1 to any base is 0
- (2) $\log(a/b) = \log(a) - \log(b)$

If a message has one possible outcome, there is no uncertainty; so $H(1)$ is 0, and the logarithmic measure fits in with this as well.

The uncertainty H could be measured using any base for the logarithms, but there are considerable advantages in using 2 as the base, as we'll see presently. So from now on we'll use "log" to mean "logarithm to the base 2."

One advantage in using 2 as the base works like this—the uncertainty associated with 8 outcomes is $\log(8) = 3$,

since 2 raised to the power 3 is 8. We could think of this 3 as the number of characters we need for representing 8 different alternatives if we use a character set which has only 2 characters.

For example, if we use only the two characters 0 and 1, we know we can have the 8 alternative codes: 000, 001, 010, 011, 100, 101, 110, and 111. More generally, with a string of K characters, each of which is a 0 or a 1, we can have 2 raised to the power K alternative patterns.

The first character can be a 0 or 1, and corresponding to this, the second can be a 0 or 1, giving us $2 * 2$ possible patterns. With 3, we can have $2 * 2 * 2$ possible patterns, and so on, until we get 2 raised to the power K patterns for K characters.

So the amount of uncertainty with 2 raised to the power N alternatives is N units. Now we can say that the amount of uncertainty in a situation with N alternatives is the minimum number of characters required to code a different message to represent the different alternatives, using a two-character alphabet (0 and 1) for coding.

The word "minimum" above is important. If we have two alternatives, the minimum string length we need for representing the two alternatives is just 1, since we can say that 0 stands for alternative one, and 1 for alternative two. We could be wasteful and use longer strings, and say 00000 stands for the first alternative and 11111 for the second and so on, but we're thinking of the most economical way of distinguishing between the alternatives.

When data is represented in a computer's memory using 0's and 1's, the on/off switches which we use to represent the information are usually described as holding one "bit" of information each. So, we can say that the uncertainty or information content of a message selected from N alternatives is $\log(N)$ bits.

However, we should keep in mind that when we talk bits here, we're thinking of the least number of bits necessary to represent the alternatives. To emphasize this, I'll use the word *s_bits* (for Shannon bits) when referring to the measure of uncertainty. This will distinguish it from the bits used for holding the information in a computer.

The simplest case is 2 alternatives, which we can associate with the uncertainty of $\log(2) = 1$ *s_bit* of information.

Shannon used the name "entropy" to refer to the amount of uncertainty in an outcome, since the logarithmic formula he developed was very similar to one which had been used in the field of thermodynamics. (The thermodynamics formula was used to measure a similar situation where there were a large number of possible configurations.)

In the examples above, I've considered cases where there are 8, 16, or 32 alternatives, all powers of 2. So, when we take the logarithm to base 2, we get whole numbers like 3, 4 or 5. This is the number of 0's and 1's we need to represent the number of alternatives uniquely. But what if the number of alternatives is, say, 5?

If we look in a table of logarithms, we find the logarithm of 5 to the base 2 is 2.32(193). It doesn't seem to make sense to say that we need a minimum of 2.32 0's and 1's to uniquely represent 5 possibilities.

First, we can see in a general way why it comes out as more than 2 but less than 3. With two characters, we can represent only 4 possibilities (which is not enough), and with 3 we can represent 8 possibilities (which is more than we need).

However, we can go further than this. If we try to code a message distinguishing among 5 alternatives, we're forced to use 3 characters, even though it's more than we need.

We think generally of 2 raised to the power 1 or 2 or 3 or 4 as meaning 2 multiplied by itself 1, 2, 3, or 4 times. What then is 2 multiplied by itself 2.32 times? 2 raised to the power 2.32 = 5.

What if we raised 5 to the power 100 (i.e., multiply it by itself 100 times)? That would be the same as multiplying 2 raised to the power 2.32 by itself 100 times. This should give us 2 raised to the power $2.32 + 2.32 + \dots$ a hundred times (i.e., 232, a whole number). (Incidentally, both of these are pretty close to 17 followed by 37 zeros).

Coding Blocks

From the viewpoint of information theory, the implication is: if we code the messages selected from 5 alternatives, we need 3 characters to code each of them. As we saw above, we're wasting some of the capacity of these strings. But if we try to pool sets of such messages, we may be able to cut down the waste.

An Example

Suppose we consider 100 messages with 5 alternatives each.

The number of alternative patterns we have is $5 * 5 * 5 \dots$ or 5 raised to the power 100. However, since 5 raised to the power 100 is the same as 2 raised to the power 232, it means we should be able to code a block of 100 messages using just 232 characters, and not 300 characters as we must if we coded them individually.

We could economize some by using smaller blocks. Suppose we have 3 such messages. The number of alternatives is $5 * 5 * 5$, or 125. If we code them individually, we have to use 3 characters for each, or 9 for the whole set. But if we take the 3 messages as one then we need only 7 characters (0's or 1's) to handle 128 alternatives.

So 7 characters would be sufficient to represent the 125 alternative patterns

we're concerned with. And the block of 3 messages needs only 7 characters (or 2.33 characters per message).

By using even longer blocks, we move down to the limit of 2.32, which the measure of information/uncertainty tells us is the minimum number of characters required.

Practical Examples

I've already talked about a number of alternative messages. Now, in practice, we don't send messages by having a set of codes for each of the possible alternatives. We construct them by using a series of characters from some alphabet.

For example, a message may be made up of letters of the English alphabet, plus punctuation marks, numbers, and so on. But the analysis above can easily handle this.

Think of each character in the message as a "mini-message" selected from a set of alternatives, since we could have chosen any one of the characters of the alphabet.

We know that computers store or transmit information coding the letters of the alphabet using strings of 0's and 1's. The most widely used code is ASCII,

which uses a string of 7 0's and 1's to provide 2 to the power 7, or 128 alternatives to represent the letters, numbers, special symbols, etc.

So each character is a message with about 7 units of uncertainty. And since uncertainty is additive, we can think of the uncertainty in a long piece of text as the sum of the uncertainty in each of the strings of 7 0's and 1's used.

Now let's look at the measure of uncertainty from a slightly different angle, which will help in considering the case where the alternatives aren't identically possible.

We saw that if there are, for example, 8 identical possibilities, the uncertainty associated with the situation is $\log(8)$ s_bits.

So, when we receive a message confirming any one of the 8, the amount of information conveyed is $\log(8)$, and since they're equally possible it makes sense to say that the average amount of information conveyed by the receipt of a series of different messages selected from this set is also $\log(8)$.

Another way of looking at this is to say that since there are 8 equally possible messages, the chances or *probability* as-

sociated with each of the messages is 1/8. From this point of view, we can say that the uncertainty resolved by each message is $\log(1/\text{probability of the message})$. This viewpoint allows us to extend the theory to a case where the messages are not equally possible.

Unlikely Messages

Let's consider a case with two alternatives—"A" which has a probability of 4/5, and "B" which has a probability of 1/5. We can say that the uncertainty resolved by receipt of alternative "A" is—

$$\log(1/(4/5)) = \log(5/4) = .322 \text{ s_bits}$$

and the uncertainty resolved by receipt of alternative "B" is similar—

$$\log(1/(1/5)) = \log(5) = 2.322 \text{ s_bits.}$$

We could also say that, in a long series of messages, on the average "A" would be received 4/5 of the time, and "B" 1/5 of the time. So, the average uncertainty resolved by messages from this set of two messages is a weighted average—

$$4/5 * 0.322 + 1/5 * 2.322 = 0.722 \text{ s_bits.}$$

Now this brings up some interesting issues. What does it mean to say that the uncertainty is less than 1 s_bit, both in the case of the message with probability 4/5 and for the weighted average?

Let's first look at the uncertainty associated with alternative A, which we know is $\log(5/4)$, which (from the properties of logarithms) is the same as $\log(5) - \log(4)$.

From the viewpoint of uncertainty, this tells us that the uncertainty has been reduced from a situation of 5 equally possible alternatives to one of 4 equally possible alternatives, and so is not much of a reduction in uncertainty.

Since "A" has a probability of 4/5 and "B" has a probability of 1/5, we can think of the selection as being from a bag in which there are four pieces of paper marked "A" and one marked "B." So, when "A" materializes, it only means that one of the four A's has been picked.

But even though we are technically "uncertain" which of the four A's was picked, it doesn't really matter. For our purposes the four A's are equivalent. So this residual "uncertainty" of $\log(4)$ isn't important, although it explains why the formal reduction in uncertainty comes

MAILBASE SYSTEM™

Productivity Software

for dBase/Wordstar/Ventura Publisher

- Letters/forms/contracts • Production • Record keeping • Grouped/repeated work, variations • Secretarial or professional use • Meeting management • Desktop input • dBASE file organizer • Develop your own specialized system with no programming • Constant or on-the-fly formatting • Stackware with standard programs

Painless construction of general letters, customized contracts, tabbed tables for Ventura Publishers, etc; from dBASE II or III files. Use any version dBASE & Wordstar/MM. Track meeting participants; contracts; business letters; automatically make action summaries. Over 5 years of practical development. Never again type anything twice. MS/PC-DOS, but also an Apple II CPM-Softcard version (not 7), 5¼ disks.

FEATURES

1. Use any dBASE file & fields up to 214 (characters or numeric);
2. Automatically track outgoing multi-copy letters & variants;
3. Select any fields at run time for letter integration, adjacent fields for block text;
4. Branch to alternate letters in a single mailmerge pass then summarize regional actions;
5. Copy any fields to subsequent records, either old or newly appended, for letter/contract production;
6. Make consistent "structure extended" data dictionaries in dB format for transparent systems management;
7. Produce correctly tabbed 2, 3, or 5 col. tables for Ventura Publisher from any dB file;
8. Other dB file management & production utilities; 2 col. Harvard Publisher tables, (other wp's on request);
9. On-disk documentation: manual; tutorial; examples; letter/contract skeletons. Hardcopy manual \$12 extra;
10. Use to customize invoicing systems, meeting management operations without programming.

- NOT COPY PROTECTED
- Mail order only, \$10 secondary sales rebate
- Money order or personal check (allow ten days to clear).

SEND TO:

HARGER I.N.T.
P.O. Box 20, Grand Central Station
JKT Pouch
New York, New York 10163

ONLY
\$45

dBASE II & dBASE III are trademarks of Ashton Tate. Wordstar & Mailmerge are trademarks of MicroPro. Apple II is a trademark of Apple Computer Inc. Softcard, MS-DOS & Microsoft are trademarks of Microsoft Corporation. CP/M is a trademark of Digital Research Inc. PC DOS is a trademark of International Business Machines Corporation. Ventura Publisher is a trademark of Univation Inc. Harvard Publisher is a trademark of Softward Publishing Corporation.

Reader Service Number 26

out as a low figure when something has a high probability like 4/5.

More interesting is the conclusion we have come to that the average uncertainty is 0.722 s_bits. In terms of our previous interpretation, this means we should on the average need a string of 0's and 1's whose length is less than 1 character. But with two alternatives, how do we use an average length of less than 1 character, when at least 1 character is needed to represent 2 alternatives?

Again, the answer lies in coding blocks of messages, rather than individual ones. Let's consider the messages in blocks of two. There are four possibilities, AA, AB, BA or BB. The relative frequencies with which we will get these sequences are in Figure 1.

Since the entropy calculations tell us that, on the average, the message A should require a string of less than one character, we try to economize on the AA's by using just one character, say 0, to represent them. We want to make the four messages uniquely decodable, so we shouldn't use 0 as the start for any of the

Figure 1 - Relative Frequencies

AA	4/5 x 4/5 = 16/25	or 0.64 prob.
AB	4/5 x 1/5 = 4/25	or 0.16 prob.
BA	1/5 x 4/5 = 4/25	or 0.16 prob.
BB	1/5 x 1/5 = 1/25	or 0.04 prob.

others. So we represent AB by 10, and then for similar reasons, we represent BA by 110 and BB by 111 (see Figure 2).

We see that the average number of 0's and 1's we would use for two messages is:

$$0.64 * 1 + 0.16 * 2 + 0.20 * 3 = 1.56$$

(or 0.78 per message)

There are various theoretical considerations regarding how best to code

Figure 2 - Two-character Strings.

Message	Code	Length	Probability
AA	0	1	0.64
AB	10	2	0.16
BA	110	3	0.16
BB	111	3	0.04

messages to make them uniquely decodable (while trying to reach the minimum limit indicated by entropy calculations). I won't go into this branch of knowledge, known as Coding Theory. However, the economy and efficiency that coding theory has been able to

achieve are a direct result of Shannon's formula.

If we use one symbol for each message, we would actually be using more symbols than we need. A situation like this, in which the uncertainty per symbol is less than the maximum which can be handled (as we know, we can handle 1 s_bit of entropy using a two-character set), is said to exhibit redundancy.

While normally one would want to cut out redundancy, it has important uses. For example, we can use it to counter the distortion or "noise" of a message.

Noise

The most significant contribution of Shannon's work was a result known as Shannon's Second Theorem. (The First Theorem says it should be possible, by using block coding, to use a string of binary characters whose length equals the entropy).

The second theorem says that even if messages get corrupted in transmission, it's always possible to counteract this by sending a somewhat longer message. This is a remarkable result, and is at first sight somewhat counter intuitive. Let's consider a specific example.

Let's say we have a message in which the two symbols 0 and 1 are sent with equal probabilities 0.5. We know that with 2 equally possible alternatives, the uncertainty or entropy is log(2) or 1 s_bit. If the messages are transmitted accurately, the information conveyed is 1 s_bit for each character transmitted.

Now suppose there is random distortion, so that when a 1 is sent, 80% of the time it's received as a 1, but 20% of the time it's corrupted into a 0. Similarly, when a 0 is sent, 80% of the time it's received correctly, but 20% of the time, it becomes a 1.

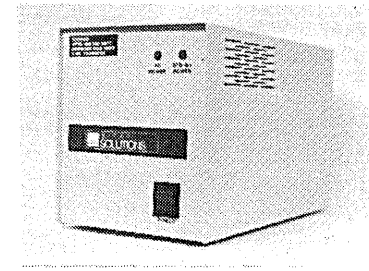
Now, from the receiver's viewpoint, when a 1 is received, there is some residual uncertainty. Was it a 1 sent which has been received correctly, which happens .5 * .8 (or .4 of the time), or was it a corrupted 0 which happens .5 * .2 (or .1 of the time)?

In other words, when a 1 is received, the receiver has 4 to 1 odds, or .8 to .2, that it's a correct 1 or a distorted 1, and could compute the entropy of the message as:

$$.8 * \log(1/.8) + .2 * \log(1/.2) = 0.72$$

The same would hold if a 0 is received, so the average entropy at the receiving end is 0.72. In contrast, with no corruption the entropy at the receiving

UNINTERRUPTABLE POWER SOURCE



MICRO

SOLUTIONS protects your equipment and your data from power outages and brownouts. Our power systems provide the fastest switching speed in the industry (2 ms ± 1).

EMI/RFI filtering and surge/spike protection all in one affordable unit. 1 year warranty on all units. Available in a size to suit your needs -

200 watts	\$290.00
350 watts	\$360.00
550 watts	\$410.00
800 watts	\$610.00
1000 watts	\$710.00

Includes shipping to your door in the continental U.S.. As specialists in overseas systems, we can supply 220 volt units. Call or write for details.

SOFTWARE SPECIAL

BROOKLYN BRIDGE

The **BROOKLYN BRIDGE** supplies the link between the new PS/2 IBM computers or laptops that use 3.5" diskettes and the rest of the MS-DOS world still using 5 1/4" drives. The cable supplied will allow you to transfer files and software between the two computers **FAST**. Simple to use and reliable. Get it now for only **\$99.00**



ASHER TECHNOLOGY PUTS FAX ON YOUR DESKTOP OR YOU CAN TAKE IT WITH YOU ON YOUR LAPTOP FOR LESS THAN \$500. We have found the answer to every small business's need for FAX at a price they can afford, and that works. Get in on this exciting new technology. Call for free demo disk!



P.O. Box 166 Riner, VA 24149
1-800-323-4829 (703) 382-6624
call 24 hours - 7 days a week

Visa MasterCard C.O.D.

We Ship Worldwide Dealers Supported

Reader Service Number 24

end is 0, since the receiver knows that what is received is what was sent. Shannon called the residual entropy the Equivocation in the message.

So, the "noisy channel," instead of resolving the uncertainty of 1, has only resolved $1 - 0.72 = 0.28$. This figure is said to be the Capacity of the channel of transmission.

Reducing Errors To Zero

On the face of it, it looks as if there may not be any recourse for remedying this uncertainty. How can the receiver know whether the 1 or 0 received was sent as such or was received after distortion?

The remarkable and powerful result which Shannon proved is this: it should be possible to find a way of coding the messages by using long blocks rather than coding the messages individually. So the error rate is reduced to 0, provided we don't try to reduce entropy at a rate higher than the capacity.

In other words, if we use an average of $1/0.28 = 3.6$ characters for reducing one unit of uncertainty, instead of the 1 character we would use in a noiseless channel, it would be possible to reduce the equivocation to 0. The price we pay,

of course, is that we're resolving uncertainty at a lower rate, 0.28 per symbol sent instead of 1 per symbol in a noiseless channel.

I won't go into the details of the proof of this theorem, but let's consider why it's important.

Shannon's theorem didn't tell us how to resolve uncertainty. It simply showed us the minimum number of symbols needed to reduce uncertainty to 0. It's up to code theorists to come up with the solutions.

This result is remarkable because it shows that something that seems intuitively impossible, is possible. And this theoretical reassurance has stimulated the development of coding methods which try to get the equivocation as near 0 as possible.

In practice, once the equivocation is very low, there may not be an advantage in lowering the rate through longer blocks. Even if we repeat a message over and over again, we can't be absolutely sure there will be no errors (but based on Shannon's theorem it is possible).

That's It

Though the theory of information was developed in specific relation to the

transmission of messages by telegraph, telephone, and so on, its general scope has led to continuing extensions of its applications (see references below).

In particular, it applies not just to transmission of information, but to storage as well. You can think of storage of information as transmission, with a delay between recording (sending) the message and retrieving it.

By associating the quantity of information in a message with the uncertainty which a message resolves, Shannon was able to develop a simply brilliant field of knowledge which has led to deeper understanding of what information is and how it can be best handled.

Of course, the theory has some limitations in that it considers the number of possible messages as the key quantity. We would say from our human perspective that a set of two messages ("you'll get a million dollars"; "you won't get it") isn't the same as the set ("you'll get one dollar"; "you won't get it"), although each has an entropy of 1 s_bit. We think of the first set as being more "meaningful" or having greater "semantic significance."

Shannon carefully pointed out in his book that the "semantic aspects of communication are irrelevant to the engineering problem."

However, in the years since the theory was developed, there have been continuing efforts to incorporate the semantic angle as well. Reference (3) in particular is an excellent source for an overview of this aspect of the theory of information.

References

- (1) Shannon, C. E. and Weaver, W.: *The Mathematical Theory of Communication* (Urbana, IL: University of Illinois Press, 1964).
- (2) Usher, M. J.: *Information Theory for Technologists* (London, U.K.: Macmillan Publishers, 1984).
- (3) Campbell, J.: *Grammatical Man* (New York, N.Y.: Simon and Schuster, 1982).
- (4) Bharath, R.: *Information Theory* (BYTE, December 1987, pages 291-298)

◆ ◆ ◆

FREE Demo Disk

**I use
Show Me!
all day long.
"...it's great!"**

Show Me!, the pop-up multi-window file viewer, allows you to browse, print, search, and paste from any file. New Version III adds EMS and LAN support, multi-file text search, and more! Our customers swear by Show Me! You will too!

Escape... through the windows of Show Me!

**Call for your FREE demo disk
(800) 634-3122**

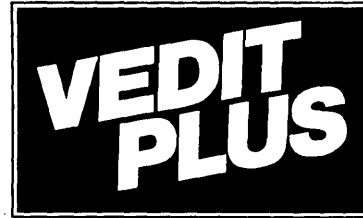
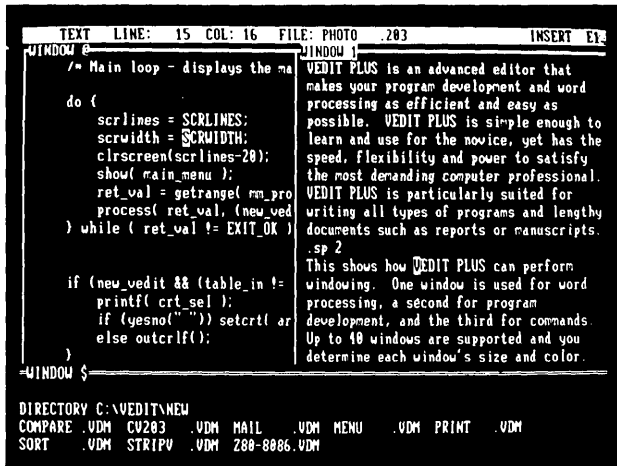
Single user version \$59.95; unlimited user network version \$119.95. Add \$5 shipping & handling. Visa & Mastercard accepted. Unconditional 90 day satisfaction guarantee.

Serengeti Software • P.O. Box 27254 • Austin, Texas 78755

Reader Service Number 27

New, Lower Prices for CP/M

- VEDIT Version 1.40 \$49 (Single file, no windows)
- VEDIT PLUS Version 2.32 \$79 (Multiple file, no windows)
- VEDIT PLUS Version 2.33 \$95 (Current version with windows)



#1 PROGRAMMABLE EDITOR

FREE Fully Functional Demo Disk *

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS defies comparison.

Try A Dazzling Demo Yourself.

The free demo disk is fully functional - you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial - you experiment in one window while another gives instructions.

The powerful 'macro' programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a 'macro' - it shows that no other editor's 'macro' language even comes close.

Go ahead. Call for your free demo today. You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Available for IBM PC, Tandy 2000, DEC Rainbow, MS-DOS, CP/M-86 and CP/M-80. (Yes! We support windows on most CRT terminals, including CRT's connected to an IBM PC.) Order direct or from your dealer. \$185.

Compare features and speed

	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	100 ⁺
Multiple file editing	20 ⁺	2	No	20 ⁺
Windows	20 ⁺	2	No	20 ⁺
Macro execution window	No	No	No	Yes
Trace & Breakpoint macros	No	No	Yes	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Configurable keyboard				
Layout	Hard	No	Hard	Easy
'Cut and paste' buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
On-line calculator	No	No	No	Yes
Manual size / index	250/No	42/No	469/Yes	380/Yes

Benchmarks in 120K File:

2000 replacements	1:15 min 34 sec	1:07 min 6 sec	
Pattern matching search	20 sec	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	11 sec

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc.

* Demo Disk is fully functional, but does not readily write large files.

Reader Service Number 7

Call for 286 / XENIX Version Fully Network Compatible

- Simultaneously edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname support.
- Horizontal scrolling - edit long lines.
- Flexible 'cut and paste' with 36 'scratch-pad' buffers.
- Customization - determine your own keyboard layout, create your own editing functions, support any screen size.
- Optimized for IBM PC/XT/AT. Color windows. 43 line EGA.

EASY TO USE

- Interactive on-line help is user changeable and expandable.
- On-line integer calculator (also algebraic expressions).
- Single key search and global or selective replace.
- Pop-up menus for easy access to many editing functions.
- Keystroke macros speed editing, 'hot keys' for menu functions.

FOR PROGRAMMERS

- Automatic Indent/Undent for 'C', PL/I, PASCAL, etc.
- Match/check nested parentheses, i.e. '{' and '}' for 'C'.
- Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
- Optional 8080 to 8086 source code translator.

FOR WRITERS

- Word Wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Support foreign, graphic and special characters.
- Convert to/from WordStar and mainframe files.
- Print any portion of file; selectable printer margins.

MACRO PROGRAMMING LANGUAGE

- 'If-then-else', looping, testing, branching, user prompts, keyboard input, 17 bit algebraic expressions, variables.
- Flexible windowing - forms entry, select size, color, etc.
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Menu-driven tutorial

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821



A C Compiler Update

And Secrets Of MS-DOS

By Scott Robert Ladd

P.O. Box 61425
Denver, CO 80206
303) 322-7294

Scott updates his C review from last issue and then looks closely at C's near pointers, far pointers, model sizes...

Whew! I'm glad that C compiler review is over with — that was a lot of compiling and linking and compiling and linking and... what? The UPS man? With another compiler? More than ONE?! Well, I guess I'll sign for it...

A current table of benchmark results is available on the Micro Cornucopia BBS, at (503) 382-7643. For example, it now contains figures for Datalight Optimum-C v3.14, Turbo C 1.5, Mark Williams Let's C v4.0.12, and MIX Power C v1.0.0.

Let's start with an update on the compilers reviewed last issue.

OOPS

Somehow, a line got dropped somewhere in my comments about Quick C in issue #40. Quick C DOES use standard libraries (.LIB files). The integrated version uses just the standard Quick Libraries, unless told otherwise in a Program List (used to build a MAKE control file). The Quick Library provided excludes such things as math and graphics functions. To add functions not present in the default Quick Library you must either create a new Quick Library, or include the Medium model library (MLIBCE.LIB) in a Program List.

Microsoft Update

Early versions of Quick C would cause a hard disk crash when used with a specific type of old Western Digital controller. This was a bug in the controller, not in Quick C. You can either upgrade your controller ROM (by contacting Western Digital), or get a modified version of Quick C from Microsoft (call (800) 426-9400).

Microsoft's Optimizing C (which I abbreviate as MSC) version 5.00 will not compile some programs containing complex switch

statements embedded in loops. The compile will fail with an "out of heap space" error. I still haven't figured out why some switch statements are a problem, while others aren't. The problem only occurs in the small and medium models, when the /Oa or /Ox switches are used.

Microsoft is working on a new (5.1) release of MSC. It will have an editor, a new linker, and will work with both MS-DOS and OS/2. It will be possible to create dynamically-linkable libraries with the new version. All known bugs will be fixed. Also coming from Microsoft are new versions of Macro Assembler (MASM), Pascal, Fortran, and a business BASIC compiler. All of these new compilers will work with both MS-DOS and OS/2. A new OS/2 Software Developer's Kit, with a \$350 price tag, will also be announced. My arm is getting sore filling out update forms...

Speaking Of OS/2...

Lattice will soon be releasing a new version of their C compiler which will support OS/2 development. The new compiler will run under both MS-DOS and OS/2. A global optimizer and a symbolic debugger will be included.

Datalight Optimum-C

The folks at Datalight are up to version 3.14 now; it seems almost as though they're running the version-of-the-month club. I appreciate Walter Bright's (the author of Datalight C) attitude: if there's a bug, he wants the users to get the fix as soon as possible. Although this means the compiler constantly changes, it also means a better and better compiler. Datalight has a subscription BBS (for \$60 per year), where you can get the absolutely latest version of the compiler at any time; otherwise, you can update to the current version for \$20.

Performance with Optimum C 3.14 has improved, especially in the area of floating-point calculations. Code size has increased just slightly (approx. 200 bytes) from version 3.12. I still don't recommend this compiler to beginners (poor documentation), but an expert can't go wrong with Optimum-C.

Borland Turbo C v1.5

I mentioned Turbo C v1.5 in the review, pointing out that you would see it by the time the review appeared in print. True enough, Turbo C 1.5 has arrived, and it looks good. An object-module librarian and a GREP utility are now included. The compiler itself hasn't changed much; in fact, it generates slightly larger (800 - 1000 bytes) executables than before, due to an increase in the size of some library modules. One surprising difference is in the speed of disk access, where the time with version 1.5 is 25% more than with 1.0.

All Borland languages (Turbo C, Turbo Pascal, etc.) support a graphics library called the "Borland Graphics Interface", or BGI. It supports nearly every PC graphics adapter I've heard of, including Hercules and EGA cards. It can automatically sense which type of adapter is installed, allowing you to write programs which will run on multiple adapters.

Mix Power C

This compiler is being heavily advertised as a competitor to both Turbo C and MSC. In the past, MIX has produced a non-standard compiler (known as MIX C), and was generally considered an excellent choice for learning C. Power C is an ANSI compatible, optimizing compiler, and sells for the amazing low price of \$19.95. What do you get for such a small amount of money?

Power C has a huge library (400+ functions) which is a superset of the MSC v4.0 library. There seems to be a function for every occasion, including simple graphics and support for CGA, EGA, and Hercules graphics cards. The source for the library is available for \$10.00.

The performance claims on the back cover of the manual look awfully good. Are they true? Well... there are bugs in the release (1.0.0) I received. For instance, even following MIX's suggestions, I couldn't get Power C to compile any programs which contain recursion. I'll wait for the "fixed" version they're sending me. As soon as I can compile all the benchmarks, I'll add Power C to the master charts.

Memory Models and Pointers

If you're going to do intricate or system-level programming under MS-DOS, you'll need to understand how the microprocessor and MS-DOS handle memory. Let's take a quick look.

The 8086/8088 microprocessor segments its memory, whereas chips such as the older Z80 think of memory as being "flat." In a flat memory space, addresses are linear. The Z80 uses 16 bits to indicate an address, which translated into 64K, or 65,536 bytes of memory.

To expand memory addressing beyond 64K, Intel added four more bits of address. That 20 bits meant the 8086 family could access 1 meg (64K X 16). Since twenty bits don't fit into a single 16-bit word (without data compression), Intel used two words to specify a memory location, one called the "segment," the other the "offset."

Editor's note: I used to wonder why Intel didn't just paste the two addresses together as a 32-bit total. Then it could address lots and lots of memory, make life easier for new programmers, and compete with Motorola's 68000. But I suppose that was too obvious.

To generate a 20-bit address, the segment word is shifted 4 bits to the left (multiplied by 16) and then added to the offset. This gives the 20-bit linear

(code) segment, while the DS register contains the current data segment.

A near address is faster because the processor doesn't have to load a new value into a segment register. However, a near address can only range over a 64K area, while a far address reaches any location within a megabyte.

Memory Models

There are several ways of using segmented memory when creating a program, and this is where the concept of "memory models" was born.

Many compilers let you select from six models. Turbo C supports all six models; Turbo Pascal only supports the Large model. Figure 1 shows the various models.

Sometimes data needs a single unbroken space larger than 64K. The huge model pastes together segments to create a larger single space.

Pointers

Most programming languages have a data type called a "pointer" which contains a memory address. The size of a

Figure 1 - Memory Model Sizes

<u>Memory Model</u>	<u>Code Addresses</u>	<u>Data Addresses</u>
Tiny (Compact)	code and data share one segment	
Small	near	near
Medium (Program)	far	near
Compact (Data)	near	far
Large	far	far
Huge	far	huge

pointer depends on the memory model. address. Two different segment and offset pairs can point to the same memory location.

There are 65,536 possible segments, each 16 bytes apart. We call these 16-byte chunks "paragraphs." MS-DOS aligns its memory areas on paragraph boundaries and allocates memory in 16-byte multiples.

Near And Far

Due to the complex nature of memory, addresses come in two types: near and far.

A near address is a single 16-bit word, containing an offset within the current 64K segment. A far address consists of two words, one a segment, the other an offset.

Near addresses don't change the segment (usually stored in one of the processor's segment registers). The CS register contains the current executable

pointer depends on the memory model.

In the Medium model, pointers to procedures and functions (code) are two words long, containing a far address. The data pointer is only one word long, containing a near address. Thus, the Medium model is very efficient for programs with large amounts of code, but relatively small amounts of data. Some compilers offer "mixed model" programming. This lets you declare pointers far and near (or far and wide).

Now that you (hopefully!) understand the concept of memory models and pointers, let's begin dissecting the special (and largely undocumented) ways MS-DOS uses the PC's memory.

Memory Control Blocks

One function of an operating system is to regulate the use of memory. MS-DOS must organize space for its own code, device drivers, TSRs, normal

programs, and data. The Memory Control Block, or MCB, is DOS's primary tool for memory management. An MCB takes up 16 bytes (one paragraph), of which only the first five bytes are significant. These five bytes have the format shown in Figure 2. (Note: I use a trailing "h" to designate hexadecimal values.)

The Block Size specified in the memory control block tells how many paragraphs of memory are in the block controlled by the MCB. An MCB's block of memory follows immediately.

The MCBs are organized as a contiguous chain. If the chain byte is an "M," it's a member of the chain; otherwise, it's the last block in the chain. MS-DOS function 48h allocates a new memory block, function 49h deletes a block, and function 4Ah attempts to change the size of a block.

Undocumented function 52h will return a segment in the ES register and an offset in BX; subtracting 2 from BX will give you the address (in ES:BX) of a word containing the segment of the first MCB. I'm not sure what this function actually points to, but it works for find-

Figure 2 - Format Of The Memory Control Block

Byte Offset	Byte Length	Purpose
0h	1	Chain Indicator
1h	2	The segment of the owning Program's Program Segment Prefix (PSP)
3h	2	Block Size (in 16-byte paragraphs)

Figure 3 - The PSP Format

Byte Offset	Byte Length	Purpose
00h	2	The 8088 instruction INT 20h. A program can jump to this instruction to exit. Calling function 4Ch of INT 21h is considered to be the proper way of exiting, though.
02h	2	Segment address of the top of allocated memory.
04h	1	Unknown. Reserved by MS-DOS.
05h	5	A far call to the MS-DOS function dispatcher, equivalent to executing an INT 21h.
0Ah	4	The previous INT 22h (DOS Terminate) address.
0Eh	4	The previous INT 23h (CTRL-BREAK) address.
12h	4	The previous INT 24h (Critical Error) address.
16h	2	PSP segment of parent program. UNDOCUMENTED.
18h	20	MS-DOS file handle table. UNDOCUMENTED.
2Ch	2	The segment of this program's copy of the environment. UNDOCUMENTED.
2Eh	34	Unknown. Reserved by MS-DOS.
50h	3	INT 21h and far RET instructions. This is definitely NOT a recommended way of accessing INT 21h.
53h	2	Unknown. Reserved by MS-DOS.
55h	16	Default File Control Block (FCB) number 1. MS-DOS parses the first command line parameter into this area, making the assumption that it's a file name.
65h	16	Default File Control Block (FCB) number 2. MS-DOS parses the first command line parameter into this area, making the assumption that it's a file name. Both this and FCB 1 are hold-overs from CP/M.
80h	1	The length of the command line tail. Note that this area is also the beginning of the default Data Transfer Area (DTA).
81h	127	The tail of the command line used when this program was executed. MS-DOS makes some changes to it; see text. Also note that this area gets overwritten as part of the DTA, unless the DTA is moved elsewhere.

END OF LISTING

CIRCUIT BOARDS

PCB-Edit...creates multi-layered PCB's with ease. Included are solder mask and legend ink support, plotter and printer drivers, and one of the fastest CAD artwork layout packages for the IBM

Only..... \$99.95
Demo Disk.... \$10.00

PCB-Shop...will build your double sided, plated thru holes circuit boards from PCB-Edit files, or your artwork for only \$1.00 per square inch in single quantity. No set up charges for PCB-Edit files, \$25.00 set up charge for other artwork.

ANALOGIC...the 32 channel logic analyzer for the IBM PC/XT has a 16 bit trigger word, 80 nano second sample time, and costs only:

ASSEMBLED BARE BOARD
\$399.95 **\$99.95**

Call or write for more information.

ANALOGIC
Phone (602) 458-4065
P.O. Box 3228
Sierra Vista, Arizona 85636

Reader Service Number 38

ing the first MCB in versions 2.11, 3.1, and 3.2x of MS-DOS.

Several programs, most notably the public domain MAPMEM from TurboPower Software, examine the chain of MCBs, using them to discover what is currently in memory. After finding the first MCB, subsequent ones can be located with the following formula:

```
Next MCB segment =  
    16 * (Current MCB Size + 1)
```

If the chain of MCBs is broken, MS-DOS will eventually crash with a "Memory Allocation Error." Only the last block of memory in the chain can be deleted safely; this is why you're in trouble if you remove a TSR from memory when it's followed by other programs and/or TSRs.

Program Segment Prefix (PSP)

Each MCB has a pointer to the Program Segment Prefix, or PSP, of the program which owns it. The PSP is 256 bytes long, and as its name implies, it resides in memory as a prefix to the program it is associated with. The PSP is rather fascinating, containing some very handy and interesting information. Figure 3 shows the format of the PSP, including several undocumented fields.

Many programming languages implement a special, reserved variable which points to the segment address of the PSP. In most C compilers, the variable is called `_psp`, and is declared in the header file `DOS.H`.

Turbo Pascal calls it `PrefixSeg`, and it's part of the DOS unit. Turbo also has a pair of DOS functions which provide the PSP of the currently executing program.

Calling the DOS function dispatcher (INT 21h) with the AH register set to either 62h or 51h will get you the current PSP in the BX register. Why there are two identical functions (in MS-DOS 3.21, they execute *exactly* the same code!), no one knows. Function 62h is documented, while 51h is not. But 62h is only available in DOS 3.x or higher. Function 51h worked in my tests for DOS versions 2.1 and later.

To access the PSP, a program should define a far pointer to a structure (or record in Pascal) which is formatted like the PSP. Then, using the PSP segment and an offset of 0, the pointer can be assigned to address the PSP data area.

Figure 4 shows this technique using Turbo C. It will display the command tail contained in the PSP.

Figure 4 - Accessing The Program Segment Prefix

```
#include "stdio.h"  
#include "dos.h"  
  
extern unsigned _psp; /* declare the _psp variable */  
  
/* define the PSP structure */  
struct PSP  
{  
    char        fill1[10];  
    void far    *old_INT_22;  
    void far    *old_INT_23;  
    void far    *old_INT_24;  
    unsigned int parent_seg;  
    unsigned char handles[20];  
    unsigned int environ_seg;  
    char        fill2[82];  
    unsigned char tail_len;  
    char        cmd_tail[127];  
};  
  
/* declare a pointer to a PSP structure */  
struct PSP far *PSPdata;  
  
main()  
{  
    unsigned char i;  
  
    /* make a far pointer to the PSP */  
    PSPdata = (struct PSP far *)MK_FP(_psp,0);  
  
    /* print the command tail */  
    for (i = 0; i < PSPdata->tail_len; ++i)  
        putchar(PSPdata->cmd_tail[i]);  
  
    /* with a final CR */  
    putchar('\n');  
}
```

As you can see, my structure doesn't define every field in the PSP. Some of the information in the PSP just isn't terribly useful. For example, the hard-coded places which execute interrupts and jumps are holdovers from MS-DOS's early days when it tried to emulate CP/M. The same thing goes for the two FCB blocks; FCBs have been supplanted by I/O functions which use file handles.

The program in Figure 4 accesses what is generally the most useful item in the PSP—the "command tail," beginning at offset 81h. DOS takes the command line which executed the program, strips off the name of the program, and stores the remainder in this area.

I/O redirection and pipe operators (<, >, and |) are also removed from the line before storage. The length of the resulting string is stored in the PSP at offset 80h. A carriage return (not included in the length) terminates the string.

Accessing The Command Line

Most programming languages support functions for accessing the com-

mand line. C, for example, has `argc` (argument count) and `argv` (an array of command line component strings), which are passed to `main()` as parameters. Turbo Pascal has the `ParamCount` and `ParamStr` procedures.

While very useful, these built-in functions assume that items on the command line are separated by spaces and tabs. Sometimes it's necessary to examine the command line directly, parsing it according to your own rules.

A word of warning: certain DOS functions store the data they retrieve in a location known as the Disk Transfer Area, or DTA. The initial DTA address is set to offset 80h within the PSP, right where the command tail is stored.

The FCB file and the directory search functions use the DTA. Very, very few compilers use the FCB functions for file I/O, preferring the more flexible and modern "handle" I/O functions introduced with DOS version 2. If you do use a function which puts its data in the DTA, such as the directory search functions 4Eh and 4Fh, use function 1Ah to set a new DTA address, or save the command tail in your own buffer.



The C Store™

EVERYTHING FOR THE C PROGRAMMER

**PLUS FREE SHIPPING!
FREE SOFTWARE!**

LATTICE C COMPILER Ver 3.2 \$229
The classic DOS development environment.

MICROSOFT C COMPILER Ver. 5.0 \$269
Innovative CodeView™ debugger, "make", more.

TURBO C COMPILER Ver. 1.5 \$69
Fast, full development environment bargain.

INSTANT C INTERPRETER Ver. 3.0 \$379
Instant linking, execution and debugging!
Directly link Microsoft, Lattice libraries.

C-TERP C INTERPRETER Ver. 3.0 \$229
Virtual memory support, versions for all compilers.

PC LINT Ver. 2.10 \$99
Shake out C bugs before compiling; neat!

PANEL PLUS Ver. 1.0 (w/source) \$379
Complete screen I/O development, no royalty.

WINDOWS FOR C Ver. 4.14 \$149
WINDOWS FOR DATA Ver. 2.06 \$229
Flexible, fast, high quality windowing system.

GREENLEAF LIBRARIES:
Functions Ver. 3.10 \$129
Communications Ver. 2.10 \$129
Data Windows Ver. 2.10 \$159
Data Windows/With Source \$269
Seasoned, reliable library leader of the pack.

CTREE Ver. 4.1 \$299
RTREE Ver. 1.1 \$229
CTREE/RTREE Package \$499
One of the fastest B-trees, handles networks.

BTRIEVE Ver. 4.1 \$179
XTRIEVE Ver. 3.02 \$179
XTRIEVE report option Ver. 3.02 \$99
Innovative performer, fault tolerant B-tree.

dbVISTA with Source Ver. 2.21 \$389
dbQUERY with Source Ver. 1.0 \$389
Very fast portable B-tree, SQL query option.

NORTON GUIDE: C Ver. 1.0 \$69
NORTON GUIDE: C/ASM Twin Pack Ver. 1.0 \$110
Online help and reference when you need it.

**THE BEST QUALITY C PRODUCTS
AT THE BEST PRICES!**



ORDER TOLL FREE:
(800) 356-0909

IN NEW YORK CALL:
(800) 341-1950, EXT. 889

- FREE! PC-Write V2.71 complete word processor or Spectacular Two Player, Real-Time SPACEWAR V1.71 with every order!
- FREE! No charge for UPS Ground.

- No surcharge for VISA or Mastercard.
- 24 hour 1200 Baud order line! (914) 241-9324.
- Customer/Technical Support (914) 666-8119.
- No APO, FPO or international orders.
- 30 day money back guarantee on unused items with intact seals.

The C Store, Suite 277
487 East Main Street, Mt. Kisco, NY 10549-0110

Reader Service Number 53

The stored interrupt vectors can be handy, too. These are used by MS-DOS to restore the previous vectors upon program termination, in case the program changes them.

When capturing one of these interrupts in a memory-resident program, you need to change the associated vectors in the PSP to point to your Interrupt Service Routines (ISRs). Do this before executing the Terminate and Stay Resident (TSR) function. Otherwise, DOS will restore the original vectors, and any ISRs you may have installed will no longer be attached.

File Handle Table

The file handle table (once again, undocumented by Microsoft) explains some things about MS-DOS file handling. When using the UNIX-like I/O functions (which appeared with MS-DOS 2), files are referenced by a one byte "handle."

DOS uses this number as an index into the table stored in the PSP, which contains what I call the "internal handle" of the given file. The internal handle is an index into the primary table of DOS file handles, stored deep down inside MS-DOS. The internal table

will have from 8 to 255 entries, depending on the FILES= entry in your CONFIG.SYS configuration file.

When a program starts up, the first five entries in the PSP handle table are automatically assigned (in order) to the values found in Figure 5.

Any value other than FFh in a handle indicates that it is open to a file or device. As you can see, this system gives MS-DOS great flexibility in assigning a program's I/O to files and devices. Since MS-DOS treats a device the same way it treats a file, it can do I/O redirection (the >, <, and | command line operators) merely by changing the internal handle in the program's PSP handle table.

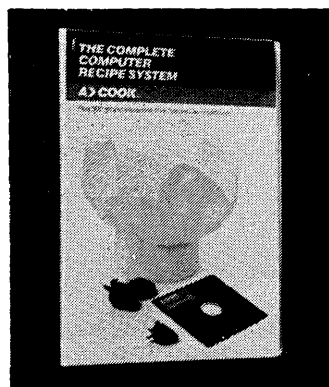
Finally

I hope the information above gives you plenty of room for thought and experimentation. In the next issue, I'll explain how the MS-DOS environment works, and how to manipulate the master copy of the environment. I'll also keep you updated on new developments in C compilers and tools. See ya then!



Figure 5 - Automatic Assignments In The PSP

Table Entry	Handle Name	Value (Device)
0	Standard input	01 (CON- console)
1	Standard output	01 (CON- console)
2	Standard error	01 (CON- console)
3	Standard auxiliary	00 (AUX- serial port)
4	Standard printer	02 (PRN- printer)
5 thru 20	Open handles	FFh



THE COMPLETE COMPUTER RECIPE SYSTEM

Use your personal computer to plan your weekly meals, parties and special Holiday meals. Includes 75 basic recipes and 500 recipe references from 5 popular cookbooks. Easy to use screens to help you set up your own recipes and categories. Add up to 200 of your own recipes, up to 1200 recipe references. Has a unique feature to help you create a cross reference index of recipes, giving the recipe name, book or magazine and page number. Has fast, easy to use software to help you with your recipe selections. Adjust ingredients from (1-99) servings. Print recipes for friends and kitchen use. Prints the shopping list too!

\$39.50

Apple II/Ie/Iic, one disk drive Order 523A
IBM PC/JR/XT/AT & compatibles, 128KB, 1 disk dr. 523I
Add \$2 for shipping & handling. Send check or Money Order.

SATISFACTION GUARANTEED

Lori & Nick's Enterprises
57 Bridge Street • Box 392-A
Hillsboro, New Hampshire 03244

C CODE FOR THE PC

source code, of course

C Source Code

Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	\$400
CQL Query System (SQL retrievals plus windows)	\$325
GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	\$325
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	\$300
Greenleaf Data Windows (windows, menus, data entry, interactive form design)	\$295
Aspen Software PC Courses (System V compatible, extensive documentation)	\$250
Vitamin C (MacWindows)	\$200
Essential resident C (TSRify C programs, DOS shared libraries)	\$165
Essential C Utility Library (400 useful C functions)	\$160
Essential Communications Library (C functions for RS-232-based communication systems)	\$160
Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	\$150
Greenleaf Functions (296 useful C functions, all DOS services)	\$150
OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	\$150
Turbo G Graphics Library (all popular adapters, hidden line removal)	\$135
American Software Resident-C (TSRify C programs)	\$130
CBTree (B+tree ISAM driver, multiple variable-length keys)	\$115
Minix Operating System (U**x-like operating system, includes manual)	\$105
PC/IP (CMU/MIT TCP/IP implementation for PCs)	\$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	\$100
The Profiler (program execution profile tool)	\$100
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	\$100
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	\$100
Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	\$95
Professional C Windows (windows and keyboard functions)	\$80
JATE Async Terminal Emulator (includes file transfer and menu subsystem)	\$80
MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	\$80
ME (programmer's editor with C-like macro language by Magma Software)	\$75
WKS Library (C program interface to Lotus 1-2-3 program & files)	\$65
Quincy (interactive C interpreter)	\$60
EZ_ASM (assembly language macros bridging C and MASM)	\$60
PTree (parse tree management)	\$60
HELP! (pop-up help system builder)	\$50
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticcomm modem card)	\$50
Heap Expander (dynamic memory manager for expanded memory)	\$50
Make (macros, all languages, built-in rules)	\$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	\$50
Coder's Prolog (inference engine for use with C programs)	\$45
C-Help (pop-up help for C programmers ... add your own notes)	\$40
Biggerstaff's System Tools (multi-tasking window manager kit)	\$40
CLIPS (rule-based expert system generator, Version 4.0)	\$35
TELE Kernel or TELE Windows (Ken Berry's multi-tasking kernel & window package)	\$30
Clisp (Lisp interpreter with extensive internals documentation)	\$30
Translate Rules to C (YACC-like function generator for rule-based systems)	\$30
6-Pack of Editors (six public domain editors for use, study & hacking)	\$30
ICON (string and list processing language, Version 6 and update)	\$25
LEX (lexical analyzer generator)	\$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	\$25
AutoTrace (program tracer and memory trasher catcher)	\$25
C Compiler Torture Test (checks a C compiler against K & R)	\$20
Benchmark Package (C compiler, PC hardware, and Unix system)	\$20
TN3270 (remote login to IBM VM/CMS as a 3270 terminal on a 3274 controller)	\$20
A68 (68000 cross-assembler)	\$20
List-Pac (C functions for lists, stacks, and queues)	\$20
XLT Macro Processor (general purpose text translator)	\$20
Data	
WordCruncher (text retrieval & document analysis program)	\$275
DNA Sequences (GenBank 52.0 including fast similarity search program)	\$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	\$60
Webster's Second Dictionary (234,932 words)	\$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	\$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	\$30
KST Fonts (13,200 characters in 139 mixed fonts: specify Tpx or bitmap format)	\$30
USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	\$20
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	\$15
U. S. Map (15,701 points of state boundaries)	\$15

The Austin Code Works

11100 Leafwood Lane

Austin, Texas 78750-3409 USA

acwlnfo@uunet.uu.net

Reader Service Number 4

Voice: (512) 258-0785

BBS: (512) 258-8831

FidoNet: 1:382/12

Free shipping on prepaid orders

For delivery in Texas add 7%

MasterCard/VISA



Building A Fancy Keyboard Translator Around Int 16h

Laine Stump

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

Indomitable Laine is willing to try anything. When he isn't jeeping about in Turkish mountains or rafting virgin rivers, he's braving the vicious winters in... No it can't be.

I've heard of privation, but Lake City? Think of the culture shock. (Is there nothing Laine won't attempt?)

I bet you're expecting to hear some story about fighting off wild hippos from a reed canoe while floating down the Blue Nile. Or maybe rappelling from 5,000 foot cliffs into a neo-Shangri La filled with dark-skinned beauties. Well, I hate to disappoint you, but I've become suddenly, but temporarily, a normal, boring guy.

I've been wishing for a snowy winter for three years (ever since I left Montana) and now I've got my wish. Maybe more than I bargained for. I just moved to Minnesota; talk about culture shock!

I decided that just reading about these great new 80386s, 34010s, and all those other 5-digit numbers (and all those acronyms, too—PC-MOS, OS/2, MTV ...), just wasn't enough. So I called up Dean and Earl at PC Tech in Lake City and made a deal to come work for them during Istanbul's rainy season.

I'll be playing with high tech toys for the next few months while I wait for the return of warm weather on the Med (and while I stock up on paddles, life jackets, and CDs of decadent rockabilly bands).

Topic Of The Month Topic

If you recall from last issue, we were talking about keyboard interrupts (9 and 16h). I covered number 9 (the hardware interrupt, not the Beatles song) last issue and will deal with interrupt 16h (the software service interrupt) this time.

But before I get down to business, I might as well get a few things off my mind, as well as tell you about the great new "stuff" I'm using these days.

Benchmark Blues

They've done it again. I don't know how it happens, but every time somebody tries to publish some benchmarks, they end up with an inaccuracy somewhere.

Take, for example, the running saga in "Tidbits" of the Standard Deviation function which mysteriously runs faster in Prolog than it does in C. Well, I have no idea why, whether, or how, that might be, but when I saw the row in the benchmark spreadsheet showing a 10 Mhz 286 running approximately 7 times faster than an 8 Mhz 186, I simply had to investigate.

Since I'm now working in a shop full of 186 based machines (X16Bs), and since the machine at my desk is a 10 Mhz 0 wait 286-on-a-card (more about that later), I'm set up well to check Gary's results. I did. And I came up with slightly different results.

Compiling with Turbo C, version 1.0 and all optimizations on, including 186/286 code generation, using compact model, I got 17.4 seconds for an 8 Mhz 186 and 8.2 (not, I repeat, **not** 2.5) on the 286. (I think I've met the guy you bought your watch from, Gary; fortunately the tour guide warned me before I gave him any money.)

Even after the inaccuracy is removed, the result is still quite interesting. The differences in the 186 vs. 286 I mean; I don't give a hoot about Prolog—it's too hotsy totsy and high brow for me. For that matter, I don't give a hoot about standard deviations either. (The only reason I passed my Engineering Statistics class when I was in college was because my favorite CS instructor was teaching Stats that quarter due to a staff shortage.)

A difference in timings between two languages on the same machine is explainable. So is a difference in timing between two different processors. But the 186 microcode is the same as the 286, the only immediately obvious advantage that the 286 has is that it does memory accesses in 2 cycles compared to the 186's 4 cycles.

But in an instruction mix with a high per-

centage of arithmetic instructions, especially multiplies, that shouldn't make much difference. Somebody wanna trace through the code this sucker's tossin' out? I might do it someday, but I'm too busy with "real" things right now.

Wait a Minute! I don't even believe in benchmarks. Why am I bothering with all this???

Gary's note: Thanks, Laine. I'm still confused about why we're getting this variation (oops, I used that statistical word) in our benchmark, but I assure you all our benchmark feedback has been from readers' testing on "their" machines. Maybe when you're finished doing "real" things, you can give this one a little more thought and get back to us.

80 What? 86

And what was that I said about a 10 Mhz (12 Mhz, actually) 286 with up to 4 Megs of 0 wait state RAM, all normal AT-type CPU support, a clock, keyboard port, all on a single Compaq 386-like (extended AT) card? Sorry. That's a secret.

Macho Video

I always liked the idea of a graphics based, multi-windowing workstation. The idea, I said. The realization was way beyond my means. Graphics made any machine I could afford too slow to bear, and there just wasn't enough room on the screen to show any more than a single window without feeling claustrophobic.

Well, the price of macho video is still higher than the cost of my old Big-Board, but then I'm not a student anymore, either (although I do still dress, live, and act like one). My main system at the office in bustling downtown Lake City, as well as containing the 286 board (ssh!), has a PC Tech 34010 Mono Graphics Board hooked up to a CPT full page monitor with a resolution of 800 x 1024 at 2 bits/pixel. So much for the claustrophobia. What about speed?

What about 6 MIPs?? (And that 10 Mhz 286 doesn't hurt, either).

I never thought I'd consider using Microsoft Windows as a development environment. Now I'm not sure. When I first tried Windows, it was version 1.03 on a Hercules card. Slow, awkward, tiny little tiled windows.

Now that I've got Windows 2.03 with its new overlapping windows running in native mode on the PC Tech 34010 going into a full page display

I've been wishing
for a snowy winter
for three years
(ever since I left
Montana) and now
I've got my wish.



(three times the dots of a Herc), I think maybe it'd be okay. If someone would just port over a good editor (the Windows NOTEPAD editor sucks) and an assembler and C and Modula compilers (all communicating with the editor, of course) maybe I'd do it.

I know you all think I'm crazy—only a fool would develop his software under an environment so bugridden as Windows. But come on guys, that was version 1, this is version 2!

Anyway, Windows doesn't quite fit the bill right now. It's overkill. All I really need is a multi-window editor that can integrate with my compilers. Oh, and it must be able to recognize the extra 41 lines on my full page display. And it's gotta be cheap.

Looking For Mr. Good-Editor

Let's see. Everyone's first answer to that is probably "Get Brief, you klutz!" Maybe. Brief does automatically recognize my 34010 Mono's full screen mode. And its macro language can integrate it with just about any compiler that spits out error messages. Earl uses Brief every day and thinks it's ludefisk soup (hey, this is Minnesota!). But Brief only has tiled windows (Yuck. Overlapped is where it's at) and besides, it's too expensive for my blood.

What else is there? Vedit? Nah. I still have bad memories of old Vedit that have held over to my impression of new Vedit. It uses tiled windows anyway. And it's expensive, too.

What about Express? Much as I hate to give up such a faithful friend, the times have changed and neither Cecil nor I have had the chance to change Express with them. It may be fast, and it does configure right up for a 64 line screen with no hiccups or growls, but it'll only edit a single file at a time. And lately I'm just too schizophrenic to cope with that.

The Solution?

I haven't found the solution to this dilemma yet, but I do have a good candidate. It's an editor called Point from Logitech. If any of you own Logitech Modula version 3.0 or a recent Logitech Mouse, you already own Point—it's included with the "Plus" software package that comes with most of their mice (as well as with Modula).

As far as I know, they don't sell it separately, but even if you buy the mouse (or Modula) just to get Point, you're still spending less than to buy just Brief, or just Vedit. And you get a great little mouse (or one of the most complete Modula development systems around), too. And Point has OVERLAPPED WINDOWS (Hubba, Hubba!!).

Like I said though, it's just a candidate. It has a few drawbacks. So far, I can only make it go up to 43 lines (the PC Tech 34010 Mono can easily be made to recognize many programs' 43 line "EGA" mode while keeping its normal 9 x 14 character font).

Then there's the problem that Point

uses the mouse heavily. That would be okay if the keyboard commands were also full featured, but unfortunately they make you simulate the mouse by doing two keypresses for each mouse button. If they'd just take a good look at Microsoft WORD. It may be huge (and expensive) but the keyboard vs. mouse thing is handled very well.

And I have found a few serious (crash causing) bugs. (I can recover by quitting and restarting.) I'm sending a list of these off to Logitech so maybe they'll soon be fixed.

Otherwise, Point is great. Extremely fast (oh, did I forget to mention fast as one of my requirements? Silly me...), multi-level undo, very configurable. It goes one step further than a macro language.

You can load in complete programs written in a "real" language (Logitech Modula, of course) which integrate with the editor; that's how they integrate it with their Modula compiler and linker. Now if they'd just explain the details of the interface so we could roll our own...

Point also lets you assign any command to any key, or (and here's the neat part) to any combination of mouse buttons or mouse buttons combined with directional movement of the mouse.

For example, you can assign right button combined with a flick left to mean "DELETE TO SCRAP" while right button with right motion is "COPY FROM SCRAP." Wild idea. Oh yeah, and you can also define your own menus of editor commands that will either pop up wherever you are on the screen or pull down from the top bar.

Still just a candidate, though. I'm very opinionated when it comes to text editors.

But I really like the concepts behind Point. I'm even going so far as to use it to write this column. It's great being able to pop up the code that I'm writing about while I'm writing about it. (And without squeezing the window this text is in, down to nothing.)

I'll be playing around more, hopefully discovering how to make it go to a full 66 lines (or even more with a smaller font). More later.

Down To Business

"How about some substance in this silly column?" you ask. Okay, I'm easy (at least that's what rumor says). Let's talk about programming. In particular, let's talk about INT 16h, the keyboard software interrupt.

I guess, before we rip it apart, a short definition of what it does and how it's normally used is in order. Very short—you can get that kind of information from any book about the IBM PC. For example, *The Pink Shirt Book* by that guy Norton.

By the way, this discussion is going to be in assembly, hex, and binary, (and Turkish) so all you pansies who only know how to program in 123 macros or (shudder) QuackBasic should go back to cutting paper dolls or turn on the Peewee Herman show for awhile.

INT 16h Services

Through INT 16h, the ROM BIOS provides client programs (applications or operating systems) with three functions related to the keyboard—get an input key, check the availability of input, and determine the pressed/unpressed state of all "shift-type" keys (control, alt, etc.).

These functions are accessed by executing an INT 16h instruction with AH set to a function code of 0 - 2. See Figure 1 for a more detailed description as well as examples of using each. By the way—I recommend that you always

use the input services provided by DOS (INT 21h function 6, 7, and etc.) unless you really *must* have the extra information provided by INT 16h (i.e., scan codes, shift status, a prior knowledge of what the next key will be).

What we're going to talk about, though, is not how to use INT 16h, but how to change it. We want to change it so that it will translate certain keystrokes into different keystrokes for us. Kind of like Superkey, only friendlier (to other programs, not necessarily to the user).

Method

We'll change INT 16h by writing a Terminate and Stay Resident (TSR) program that saves the original INT 16h vector and replaces it with our own. Then, every time there's a request for keyboard input, we'll call the original INT 16h to get a key and then translate it into another key according to a lookup table. Simple enough.

I always thought the best way of understanding a program was to look at the code, which you can find on Micro C's RBBS or get on the Issue #41 disk (\$6 to subscribers, \$8 to non-subscribers

Figure 1 - INT 16h function list and examples of use

```
Keyboard      equ    16h
conin equ     0      ;function code to put in AH
constat equ   1
shiftstat equ  2
rshift equ   0000001b ;shift bits returned in AL
lshift equ   00000010b
shifts equ   lshift+rshift
ctrl equ    00000100b
alt equ     00001000b
scroll equ  00010000b
num equ    00100000b
caps equ   01000000b
ins equ    10000000b
```

To get a character:

```
MOV    AH,conin
INT    Keyboard
;character is now in AL (if normal character)
; or AH (if AL=0 means extended code)
```

To check keyboard status:

```
MOV    AH,constat
INT    Keyboard
JNZ    KeyIsReady
JMP    KeyNotReady
```

To find state of shift keys, CTRL for example:

```
MOV    AH,shiftstat
INT    Keyboard
TEST   AL,ctrl
JNZ    CtrlNotDown
JMP    CtrlIsDown
```

and foreign). If you're so inclined, check out the code for the listing of TURKKEYS.ASM (that's Turk Keys, not Turkeys). After you've taken a look at the code, come back here so I can explain a few of the fine points (i.e., weird parts) of the code.

Explanation

You may notice some similarities between this program and the program in issue #39 which intercepted printer output on INT 17h, the ROM BIOS printer output function. That's because they have a lot in common.

The initialization routine is identical except for the difference in procedure names and interrupt numbers. It's the same basic sequence of events to install any TSR. You just print a message, save a vector, install a replacement vector, then Terminate and Stay Resident.

A Few Details

The similarities stop there, though. The service routine we've installed for the keyboard is translating input, not output. So instead of translating and then chaining to the original service routine, we must chain first, then translate.

Because we have to get control back after the ROM BIOS has its say, we can't simply jump to the original, we must call it. But the routine we are calling returns with an IRET instruction, which expects three items on the stack (offset, segment, flags) while a far call instruction only puts two items on the stack (offset, segment). What to do...

There are a couple of possible solutions. The most obvious is to do an INT instruction instead of a CALL. After all, it puts everything we need onto the stack just perfect for an IRET.

That's peachy, dear, but we've just taken the routine we want to call out of the interrupt table and stored it in a pointer variable. "So install it somewhere else in the vector table!" Well, maybe, but then what happens if 367 programs all decide to chain onto INT 16h. What if there aren't any free vectors when our program runs?

I've adopted a much more trouble-free method of chaining interrupts. I just push the flags, then do an indirect far call through the pointer that stores the address of the original routine. That's why I have all those PUSHF instructions without the corresponding POPFs; The IRET instruction in the called routine takes care of POPping the Fs.

Use INT 16h if your purpose is to translate keystrokes into other keystrokes. If you want to activate another program upon a keypress, watch INT 9.

Returning Status In The Flags

We run into another "return problem" in ConStatFunction. It's called by an INT instruction (the call from the application program to NewKbdInt), which saves the flags on the stack before calling. An IRET in the calling procedure would then restore the flags from before the INT, destroying that Z flag we worked so hard to set up.

That's why in ConStatFunction I return to the calling program with a RET 2 instead of IRET. RET 2 does the same thing as a normal RET. But after popping the return address off the stack, it adds an extra 2 to the stack pointer, effectively popping the old flags without ruining the new flags.

Also notice that I declared ConStatFunction as a FAR procedure. This was necessary to force the assembler to generate a FAR return (returning to a different segment) instead of a NEAR return (return to a point in the same segment).

This isn't necessary for the other sub-functions (ConinFunction and ShiftStatFunction) since they do IRETs, and there's only one kind of IRET. I could have declared all the other procedures in the program as NEAR or FAR, but putting in a PROC declaration takes a matching ENDP line, and I'm lazy.

The ALT Key

The only other aspect of TURKKEYS that bears explaining (other than "Why would anyone in Minnesota want to be typing Turkish text?") is the special

handling of the ALT key. The rewrite of ShiftStatFunction will only show ALT as being pressed if CTRL is also pressed.

The modification to ShiftStatFunction was necessary because Microsoft WORD was overriding the character code if it saw that the alt key was pressed. It would see ALT down and ignore the character value in AL, using the extended code in AH instead.

I also check in KeyXlate and only do a translation if my newly modified ShiftStat shows ALT as not down. In other words, holding down CTRL disables the translation and gives you the original keystroke instead. If I want to, I can now redefine all of the ALT+<letter> combinations without losing the original keys. And they always told you that you couldn't have your pizza and eat it too...

Usage

Now don't get all excited and think that you're going to write a popup program that looks to INT 16h for the hotkey. I've seen programs that do that, but they end up with a "warmkey," not a hotkey.

INT 16h is only activated when the current application asks for input, so sometimes nothing happens for quite some time (several milliseconds at least). A hotkey should hook up to INT 9, that way it will be informed the instant a key is struck. In general, use INT 16h if your purpose is to translate keystrokes into other keystrokes. If you want to activate another program upon a particular keypress, watch INT 9.

Extensions

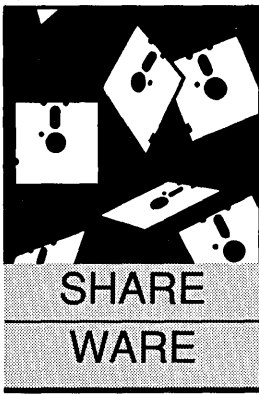
A simple way to extend the functionality of TURKKEYS (the first thing to do is change its name) is to replace the second word in each pair in the translation table with a pointer to a longer, null terminated translation string. Then keep track of where you are in the replacement string during subsequent calls.

Another way of doing the same thing would be to use that pointer to point to a subroutine to execute instead of just pointing to a string.

Then you could hook it up to a "hotkeyed" user interface (through INT 9) which would let the user define new keys, and save and recall files full of macros. Then give it a name... like... uh... QuickKey! Yeah. That's the ticket.

Let me get back to you on that one, huh?

◆ ◆ ◆



ButtonWare, Tour '88

Coming Soon To A Users Group Near You

Anthony Barcellos

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

In January Tony sat through two presentations by Jim Button. Now he's writing about ButtonWare. Am I surprised? Are you? Is ButtonWare really interesting? Sure.

Jim Button was on the road. Two major users groups in three days. San Francisco on Monday and Sacramento on Wednesday. His routine has been honed by many prior appearances, but his message hasn't changed. He just wanted to say thanks.

Over the past year, Button has descended upon practically every notable users group—both at home and abroad. "He has traveled literally all over the world," says DeeDee Walsh, ButtonWare's public relations and advertising specialist.

The California Swing

Jim was spreading the word about shareware's success, and users were eager to hear it. In Sacramento alone over five hundred people were on hand to listen to him. The father of user-supported software was telling his half of the shareware story, and he was talking to people who had helped to make it happen.

"All good programming projects never come to an end," he said, describing how his "little label program" blossomed into the flagship of ButtonWare's fleet of shareware products. PC-File started its life as an AppleSoft BASIC program running on an Apple II+. It stored, sorted, and printed labels. Gradually, however, it began to look more like a database.

When IBM announced its PC in 1981, Jim Button lined up with his fellow Big Blue employees for a new microcomputer. Unable to meet the immediate demand, IBM advised Button that a PC would be his in only nine months. Undaunted, Jim began using his lunch hours at work to convert his label program to the PC.

Jim casually shared his program with his colleagues, all of whom were eager for something to run on their PCs. Then it happened.

"They started feeding me requirements," he said.

Without realizing it, he had gone into the software business. As his program passed hand to hand, Jim found himself getting update requests from people he didn't even know. Soon updates were no longer a matter of dropping a disk onto a colleague's desk—he was laying out cash for mailers and postage. Inspiration struck.

User-Supported Software Is Born

"I began including a message asking for a \$10 contribution from people who were really serious about getting updates," he said. Button put it quite simply: His hobby had to become self-supporting, or he would have to discontinue mailing out disks. Button's wife calmly informed him that he was "crazy" if he thought anyone would send him money.

Almost immediately he discovered that someone else had had the same idea. A friend came back to Button with a disk bearing a program by Andrew Fluegelman called PC-Talk. Said Button's acquaintance, "Fluegelman is doing the same thing you're doing, but it looks like he's doing it a lot more professionally."

Audiences laugh when Button tells this story on himself, but it was a turning point in shareware's history. Fluegelman's "experiment in economics" was called Freeware (a term he trademarked; Button was calling it "user-supported software" and Quicksoft's Bob Wallace later coined "shareware"). He packaged his disk with a nicely written pitch for people to participate in this alternative marketing scheme. Now you could "try before you buy."

ButtonWare's First Price Hike

Button sent Fluegelman a copy of his database program and the two were soon collaborating. "He told me I was making him look like a skinflint by only asking \$10," said Button. "I suggested that he come down in his price. He suggested that I come up to \$25. His arguments were more persuasive than mine. That was my first price hike."

Fluegelman also talked Button into adopting a name similar to that of PC-Talk. Button had been calling his program Easy File; now it became PC-File, the name under which it became famous. The nomenclature became standard for many later shareware products, including ButtonWare's entire list of applications programs: PC-Type, PC-Calc, PC-Graph, PC-Stylist, and PC-Dial.

The software libraries of PC clubs became one of shareware's major outlets. Figures show that 19% of ButtonWare's customers learned of the products at users groups. Another 38% learned of ButtonWare from a friend. Thus user-supported software is truly user-supported, with over half of the activity coming directly from user-to-user interaction.

"How Did I Get Into This Mess?"

Jim recounted his reluctant transition from IBM employee to shareware entrepreneur. The returns from his hobby, however, finally so overshadowed his salary that he jumped.

"In 1984 I left IBM and my wife stopped talking to me."

Communications resumed shortly thereafter, as it became clear that Jim had made the right move.

Button's departure from Big Blue came one year after ButtonWare received a major boost from *PC World* magazine. Doug Clapp's rave review of PC-File III appeared in the September, 1983, issue and the small company's mailbox overflowed.

Aloha, Success

But Jim wasn't there to witness the event. *PC World* appeared on the newsstands while the Button family was flying to Hawaii for a vacation. After a few days, Button called home to check with the high school student who was collecting their mail.

"Where do you want me to put the bags?" asked the desperate boy. He eventually filled the basement with grocery sacks full of mail.

(It was right at the end of 1983 that PC-File entered the software library of the Sacramento PC Users Group, as it probably did the libraries of many other computer clubs. It was the last volume prepared by my predecessor as software librarian. Fluegelman's PC-Talk was already part of the library—though later withdrawn at the author's request. When I included Bob Wallace's PC-Write as my first addition to the library,

the Big Three of shareware were represented in our collection, which then was only five disks!)

By the time the Buttons had cleared away the backlog, Jim had his wife and a son working for ButtonWare full-time.

With each iteration, Button adds new features. Newly released PC-Calc+ and version 2.0 of PC-File+ can generate business charts from spreadsheets or databases.

Over the next four years, ButtonWare grew to a twenty-employee company, with annual revenues exceeding two million dollars. Button's wife eventually agreed that her husband's idea was not so crazy.

Biggest And Best?

Today ButtonWare offers one of the most complete collections of shareware productivity programs, covering all of the major PC applications. None of the software exceeds \$70 for a registered copy. For the list, see the table which follows:

Current Roster Of ButtonWare Software

PC-File+ 2.0 (database)	\$69.95
PC-Type+ 1.0 (word processor)	69.95
PC-Calc+ 1.0 (spreadsheet)	69.95
PC-Dial 1.0 (communications).....	59.95
PC-Stylist (style checker)	29.95
PC-Tickle (tickler/calendar)	29.95
XD (Extended DOS; shell).....	29.95
Baker's Dozen (misc. utilities)	59.95

Meanwhile, Button has been working toward "loose integration" of his

programs. In addition to sharing graphics features, PC-Calc+ and PC-File+ understand each other's file formats sufficiently to share data. In particular, PC-Calc+ can create spreadsheets from PC-File+ databases.

A Selection Of Greatest Hits

Stories have been circulating that ButtonWare will move into the integrated software market with a program called Medley. Medley lives, but not in its originally envisioned form. Instead of an integrated package, Medley has become an "integrator" for ButtonWare's database, word processor, and spreadsheet.

"That's in the vicious rumor category," jokes DeeDee Walsh. "In fact, we're already using Medley in our office. We're thinking of bundling it with PC-File, PC-Type, and PC-Calc as a business package."

However, no release date has yet been set for Medley and its three companions. Testing continues at ButtonWare.

On The Road Again

Jim Button is preparing another swing into Europe, carrying the shareware message to more enthusiastic audiences. The co-inventor of the user-supported software concept is at the peak of his form. Keep on trucking, Jim.

The Local Library

The quickest source of shareware for the interested computer user continues to be the local users group software library or electronic bulletin board system. Look around. ButtonWare is always nearby. You can also obtain ButtonWare products directly via their toll-free order line, 1-800-JBUTTON. The software prices listed above were current as of February, 1988.

For more information on Jim Button and shareware, check out David Thompson's interview in issue #37 of *Micro Cornucopia* (Sept./Oct. 1987).

Editor's note: Micro C has the Button collection. Disk MS45 contains PC-File+ (Version 1.0 with built-in doc. only), MS46 contains PC-Type, MS47 contains the writing analyzer and dictionary, MS48 contains PC-Dial and PC-Graph, MS49 contains PC-Calc and DOS helper, and MS50 contains the Calendar Reminder and a pair of Button Games.

Disks are \$6 each, postpaid, for U.S. subscribers—\$8 each, postpaid, for non-subscribers and foreign. Call 1-800-888-8087 to order.

TURBO C QUICK C LET'S C DESMET C DATALIGHT C ECO-C
LATTICE C MICROSOFT C AZTEC C COMPUTER INNOVATIONS C

NEW --- Limited time offer.

Peacock System's CBTREE

Object library for only \$49!

Our FULL COMMERCIAL VERSION of CBTREE in object library format is being offered for the amazingly low price of \$49.

CBTREE provides you with easy to use functions that maintain key indexes on your data records. These indexes provide you with fast, keyed access, using the industry standard B+tree access method.

Everything you need to fully utilize CBTREE in your applications is included. The CBTREE source code can be purchased later at any time for the difference. Example source programs and utilities are included FREE.

CBTREE source library \$159
Object library only \$ 49

This limited time offer is simply too good to refuse. Peacock's standard ROYALTY FREE, UNCONDITIONAL MONEY BACK GUARANTEE, AND FREE TECHNICAL SUPPORT applies to this offer.

To order or for additional information
call (703) 356-7029 or (703) 847-1743 or write:



PEACOCK SYSTEMS, INC.
2108 GALLOWS ROAD, SUITE C
VIENNA, VA 22180

Trademarks: Turbo C (Borland); Quick C (Microsoft); Let's C (Mark Williams); DeSmet Software); Datalight (Datalight); Lattice C (Lattice); Microsoft C (Microsoft); Aztec C (Manx Software); Computer Innovations C (Computer Innovations); Eco-C (Ecosoft, Inc).

Reader Service Number 20

Noah Boards The ARC

When last we left the archiving scene, the deluge of ARCing and de-ARCing programs was well upon us. (See "All Aboard the ARC" in the Nov./Dec., 1987, *Micro C* issue, #38.) Now shareware author John J. Newlin volunteers to take the helm. What could he have in mind?

Version 1.3 of Newlin's Arcmaster answers the novice archivist's prayers for a respite from the downpour. Arcmaster embraces Phil Katz's and Vernon Buerg's speedy archive utilities. As Newlin points out in his documentation, Arcmaster provides the menu-driven ability to:

- View an .ARC file directory listing at the touch of a key.
- DeARC multiple files.
- Selectively extract (or delete) files from an .ARC file.
- Search any (or all) drives in your system for a file that resides in an .ARC file.
- Browse ARCEd files using Vern Buerg's outstanding LIST program. [It's amusing to see that Newlin is like me in thinking that "outstanding" always precedes

"LIST." -TB]

- Quickly and easily select ARC or DeARC switches and options.
- Specify a target directory for file extraction.
- Specify a target directory as host for created .ARC files.
- Swiftly search entire system for any file.
- View any file through Buerg's LIST program.

In brief, Newlin has taken PKARC and PKXARC by Phil Katz, ARCA and ARCE by Vernon Buerg, and Buerg's LIST utility, and combined them so you can view, combine, extract, and delete archive files. User-friendliness has come to the archiving business.

As shareware programs grow in size and complexity, more software libraries and bulletin boards are using ARCing programs to combine and compress their files. New members of users groups raid the software library and come away with archives that may at first seem impenetrable. Arcmaster goes a long way towards solving that problem. Teach someone you love to use Arcmaster, and archiving worries are forever banished.

IS NOTHING SACRED?

Now the FULL source code for TURBO Pascal is available for the IBM-PC! WHAT, you are still trying to debug without source code? But why? Source Code Generators (SCG's) provide **completely** commented and labeled ASCII source files which can be edited and assembled and UNDERSTOOD!

SCG's are available for the following products:

- TURBO Pascal ver 3 (IBM-PC)* . . \$67.50
 - TURBO Pascal ver 3 (Z-80)* . . . \$45.00
 - CP/M 2.2 \$45.00
 - CP/M 3 \$75.00
- *A fast assembler is included free!

"The darndest thing I ever did see . . ."

Pournelle, BYTE

"I have seen the original source and yours is much better!"

Anonymous, SOG VI

The following are general purpose disassemblers:

- Masterful Disassembler (Z-80) . . \$45.00
- Masterful Disassembler (IBM-PC) . \$47.50
- UNREL (relocatable files) (8080) . \$45.00



"The Code Busters!"

VISA/MC/check Shipping/Handling \$1.50
card # _____ Tax \$ _____
expires ____/____ Total \$ _____

All products are fully guaranteed. Disk format,
8" 5" type_____.

**C.C. Software, 1907 Alvarado Ave., Walnut
Creek, CA 94596, (415) 939-8153**

CP/M and TURBO Pascal are trademarks of Digital Research & Borland Int.

Reader Service Number 31

The registration fee for Arcmaster is a modest \$40, but you can save yourself \$10 if you include the user's questionnaire that Newlin provides at the end of the manual. He's serious about getting feedback, and Newlin is making it worth your while to share your reactions with him.



But Wait—There's More!

In addition, Newlin provides registered users with a copy of Arcmaster Plus. (We really must find an alternative soon to the overworked "plus" suffix; don't you agree?)

The Plus version includes extra features like batch processing of file moves, deletions, or copying; printing archive directories; renaming files; assignment of function keys to filename masks; date/time sorting; aliases for ARCA, ARCE, PKARC, and PKXARC; and DOS wild cards for archive operations. (Arcmaster Plus is restricted to registered users and cannot be otherwise distributed.)

For more information, pick up a copy of Arcmaster from your favorite local source or contact:

ZRP/M™ creates
Z280®
 CP/M®2.2 compatible
IBM PC

ZRP/M is an operating system combined with a Z280 emulator. Either standalone or with DOS present, ZRP/M provides the solid base of a genuine operating system reliably distinct from the facade created by an MSDOS interface. All 2.2 system and CBIOS calls are supported, 56.5k TPA, file date and time stamping, fast virtual disk, lobyte redirection, terminal emulation, color console display, auto relog, COM path, SAVE anywhere, single key phrase recall, built-in access to DOS drives. SETDISK redefines a drive to any of over 80 CP/M formats. System disk with manual \$129. Shipping \$5 (\$10 nonUS)  



118 SW First St. - Box G
 Warrenton, OR 97146
 (503)861-1765

John J. Newlin
 4060-228 Rosenda Court
 San Diego, CA 92122

From Our Foreign Correspondent

I don't often get mail from Hong Kong, but *Micro C* has readers there. One of them is a shareware author named John Scofield.

"Trying to get a program distributed in the USA while living in Hong Kong is nontrivial," he writes.

I would imagine so. And what is it that Scofield has for us? A DOS utility called EDDY, which stands for "EDit DirectorY." (Perhaps coining acronyms isn't John's strong suit.) EDDY can sort a directory, fiddle with file attributes, search and view files, patch bytes, and copy, move, or delete files.

That really doesn't sound too exciting or original, does it? But EDDY is a slick tool just the same.

Full-Screen Files

True to its name, EDDY lets you wander about a full-screen display of your current directory and *edit*. You can edit the filename, date, time, or attribute

byte. Just cruise about with your cursor to whatever you want to change. Press Enter to record your changes or Esc if you'd rather not. There's also an Undo command.

Full-screen editing is a new idea in the directory utility field, particularly when it extends to all of the vulnerable file characteristics—like name (including extension), date, time, and certain attribute bits. The cursor leaps lightly over the file size because that's not one of the things subject to EDDY's editing.

EDDY's screens are a bit busy, but the information is useful and he includes help screens.

You can sort your directory several ways and you can invert the order with an Alt-I. You can customize EDDY, the program has a "patch" area. Since EDDY can edit any bytes in a disk file, you can use EDDY on itself, to change the defaults. (Be careful, this qualifies as user-driven self-modifying code. The most interesting stuff.)

The Better Mousetrap

Clearly there's nothing in EDDY that's really new. Scofield feels EDDY is

easier and cheaper than similar programs.

The 39-page user's manual is written in a breezy style. My particular favorite is the section titled "What Annoys Me Most About EDDY Is:" Scofield then gives a laundry list of potential problems and recommended resolutions.

The Path To Scofield's Door

EDDY is up to version 2.31 and is beginning to make its way into software libraries and onto bulletin boards. Scofield asks a modest registration fee of US\$15. Write to:

John Scofield
 Box 47136
 Morrison Hill P.O.
 Hong Kong

He notes that EDDY is also available on the *Computer Language* magazine forum on CompuServe.




ShareWare™
BROWN BAG Software™
PC Outline!

- Organize Reports and/or Papers
- Brainstorming
- Managing a Project

SOME FEATURES:

- Structured Indentation
- Powerful Editing
- Outline Rearrangement Functions
- Choice of Multiple Numbering Schemes
- Margin Control
- Centering, Left & Right Justification
- Optionally Memory Resident
- Requires Less than 90K
- User Definable Windows
- Multiple Line Text
- Importable from dBase™, DataBase™, Wordstar™, Max™, and ThinkTank™

SOME USES:

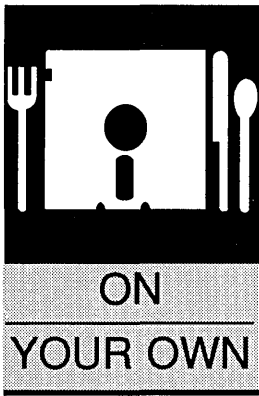
- Things-To-Do List
- Name & Address Lists

\$89.95

1-800-523-0764
 IN CALIFORNIA 1-800-823-5885 or (408) 559-4545

SHAREWARE DISK AVAILABLE FOR \$10.00

Reader Service Number 87



Is There A Market For Your Product?

Gary L. Scott
Decision Technology
P.O. Box 5040
Aloha, OR 97007

You're a technical person, right? So how are you researching the market for your new product idea? Not very technically I'd guess. In the following article Gary shows you how to apply a matrix to marketing.

How do you determine the market potential for your brainstorm? Is the easiest and best method to simply spend the next six months developing your new product? Then call up all of the dealers on the West Coast that carry your type of product and offer to sell it to them? Then sit back and wait for the money to roll in?

This article will address commercial vertical market software. Although we will concentrate on marketing commercial software, most of what we discuss here should also work for the programmer who has written a fancy new utility.

Before we get in too far, it's important to define the difference between horizontal and vertical markets:

Horizontal market: Programs like word processors that almost anyone in any job can use.

Vertical market: Specialty programs directed toward a narrow market segment or specialized need. Programs fitting into this category include routines which calculate the optimum way to saw a log into lumber.

Direction

To sell a product you must understand the needs and wants of your target audience. Your goal is to meet those needs more effectively than your competitors. This article will cover marketing. We'll also discuss tools for determining what to offer your target audience.

Let's say you have an idea for a new program and you want to assess its potential market. The easiest way to do that is to find out how many other companies are already there. A quick trip to your local software supermarket is a good start. Also check a couple of years

worth of *BYTE*, *PC*, and *Micro C* magazines.

Let's say your search turns up nothing similar to your idea.

Perhaps you've stumbled on the one remaining unexploited vertical market. But have you? Probably 75% of all software never makes it into your local software supermarket or any of the popular computer press. Why? Because the people who purchase that kind of software don't read computer magazines.

What do these folks read? Sometimes nothing. But most of these people subscribe to one or more trade magazines, and you'll usually find ads for software that appeals to the same vertical markets that the magazines do.

Be prepared to tell the inevitable "white lie" since no vendor wants to send literature to a competitor.

Great, so now that we have a source for investigating our potential market, how do we find these magazines? Normally the best source for this type of magazine is someone in the industry. They can not only tell you the names of the magazines that cater to their industry, but more important, they can often tell you which are highly regarded and which ones are ignored.

If you don't know anyone in the industry, your next best bet is a university library, preferably one that offers a degree in your chosen field.

University libraries usually subscribe to a

large number of trade magazines since many trade magazines send copies to libraries for free. If the nearest college is a two-day drive from your house, you can normally get a list of titles from the Readers Guide at your local library and then get copies via inter-library loan.

Once you have the magazines in hand, check them for software ads. Xerox any ads offering products similar to yours.

Also, it's not uncommon for trade magazines to have annual comparisons of software. For example, every December, *Sales & Marketing Management* magazine runs a comparison of several hundred sales and marketing applications packages.

Often the comparison done in these magazines is superficial, but you can get some very important information from these articles. First is an indication of the size of the market both in number of programs sold by individual vendors and in the number of vendors that are currently in the market. You can also get a feel for the price the market will pay for your product. Most important, you'll probably get the names and addresses of your potential competitors.

Armed with the names of your competitors, the work is ready to begin. All serious software vendors have literature describing their product and many have demo disks (for a nominal fee). Call them and request a copy of their literature. You may talk to one of their sales personnel who will probably ask you some questions to try and weed out the serious customer from the person collecting fuel for his wood stove.

Be prepared to tell the inevitable "white lie" since no vendor wants to send literature to a competitor. Often

you can get information from the companies by filling out bingo cards in the magazines, but you should still expect a follow-up call from the companies you selected.

Meanwhile

While you're waiting for the literature, start analyzing your competition. Go to your local office products store and purchase the largest pad of paper you can find (desk pads work well, especially if they have a pre-printed grid).

Divide the pad into a matrix of boxes

The columns in the matrix need to address all of the four Ps of the marketing mix: Product, Price, Promotion, Place. (See Figure 1.)

The list isn't complete, it's just to get you started. Note, however, that product, price, and promotion are the most important.

Phoning

After you have completed a preliminary cut of the matrix analysis, it's time to get on the phone. Start with the magazines. Talk to the authors of articles covering your type of software.

Figure 1 - The Four Ps

Product quality options packaging warranty	Price list price discounts /	Promotion advertisement how sold?	Place how sold? coverage
---	--	--	---------------------------------------

two to three inches square. Along the Y-axis write the names of all of the competing programs. (Be sure to leave room for your package.) Along the X-axis jot down all of the important features and benefits.

For instance, if you're designing a data base you should have a column for the maximum number of records—a column for variable or fixed length records—a column for the maximum length of each record—etc. Be sure and leave a number of columns unlabeled. (You'll think of more after the literature comes in.) Also include a column to track how long it takes the companies to send you literature and how often they call.

Check with columnists and editors who have mentioned your competitors' products. Keep in mind that these people are busy, don't expect them to spend hours discussing the pros and cons of every package.

Write down the questions ahead of time and then pare them down to five minutes worth. Always ask the people you call if they have the time to answer your questions, or if it would be more convenient to call at another time. They will appreciate your thoughtfulness and may very well give you more information than you would have gained otherwise. Above all, be Professional.

Record the conversations. If you find the practice distasteful, take notes



during and immediately after the conversation. Don't assume that you will remember the details two days later.

It's also a good idea to follow up with a letter of thanks. You may well be contacting these people again as you start marketing your product. You want them to remember you as a professional.

Context Questions

When you talk to these folk, you want to ask context, not content, questions. You will find context questions to be harder to answer but more informative. An example of this line of questioning: "How important is local support for the product?" The content equivalent: "How many companies offer local support?"

What you are trying to do with the interviews is fill in the fuzzy areas. For instance, the annual comparison of sales and marketing software mentioned earlier can give you all kinds of factual data, but it probably wouldn't reveal a controversy among users over the value of certain features (or the value of using a computer at all).

I can't overemphasize the impor-

tance of the matrix analysis and interviews. This work will provide a foundation for your project, so do a thorough job and keep it up-to-date.

First Decision Point

Once you have completed this matrix, you've reached your first decision point. You have enough information to make your first educated "stop now" or "continue" decision.

Look at the matrix and ask, "How am I going to differentiate myself from everyone else?"

Remember we are discussing a market-driven business. You are not IBM, and even IBM has discovered that customers will not always line up to buy a product just because it says IBM on it.

When trying to determine how you are going to differentiate your product from competition, keep in mind the four P's of the marketing mix. Product, price, promotion, place all give you potential opportunities for differentiation.

It's easy at this point to fall into the trap of saying to yourself, "I'll add features and lower the price so my product will sell like canned beer."

Editor's note: Lots of beer companies have tried the same procedure. Unsuccessfully.

Remember that everything is a tradeoff. Can you really lower the price and still offer good customer service?

Keep in mind that price differentiation is the worst reason to make a "continue" decision. Price competition can be very destructive to yourself, your competitors, and eventually, your customer.

Feasibility Study

What you have just done is a simple market feasibility study. This is not to say that you are finished with your market research, but you should now have enough information to make an informed decision on market potential.

The matrix is a tool. Like any other tool you can use it to create a work of art or a pile of kindling. It depends on you.

In a future article I'll discuss more tools—tools that take you further into the realm of market research.

If you have questions, feel free to drop me a letter.



Software Developers

We need your program!

Do you have a program that's good enough to sell, but don't want the problems or financial risk of producing, typesetting, printing, packaging, warehousing, marketing, distributing and supporting a product?

Why start your own software house?
We've done it for you!

Merlin Publishing Group
is now accepting submissions of micro-computer software for publication.

You get:

- initial cash payments
- generous royalties
- to spend your time programming

But we can't help you if you don't submit. Call or write today for our submission guideline kit.



1240 Johnson Ferry Place, Suite A10
Marietta, GA 30068
(404) 977-6034

P. S. See us in Atlanta at COMDEX-Spring '88

Reader Service Number 35

ICs

PROMPT DELIVERY!!!

SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN for FEB. 15, 1988

OUTSIDE OKLAHOMA: NO SALES TAX

640K MOTHERBD UPGRADE: Zenith 150, IBM PC/XT, Compaq Portable & Plus, hp Vectra

DYNAMIC RAM			
1Mbit	1048Kx1	100 ns	\$36.00
51258	* 256Kx1	100 ns	9.25
4464	64Kx4	120 ns	6.95
41256	256Kx1	80 ns	9.85
41256	256Kx1	100 ns	9.75
41256	256Kx1	120 ns	8.95
41256	256Kx1	150 ns	6.65
41264	+ 64Kx4	120 ns	9.25
EPROM			
27C1024	64Kx16	150 ns	\$37.95
27C1000	128Kx8	200 ns	37.50
27C512	64Kx8	200 ns	14.95
27256	32Kx8	250 ns	6.75
27128	16Kx8	250 ns	6.50
STATIC RAM			
43256L-12	32Kx8	120 ns	\$10.95
5864LP-12	8Kx8	120 ns	4.85

*STATIC COLUMN FOR DRAM COMPAG +2-PORTRAM VIDEO 8087-2 80287-8 80387-16 80387-20 \$160.00 \$245.00

OPEN 6 1/2 DAYS, 7:30 AM-10 PM: SHIP VIA FED-EX ON SAT.

SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY, VIA U.S. EXPRESS MAIL

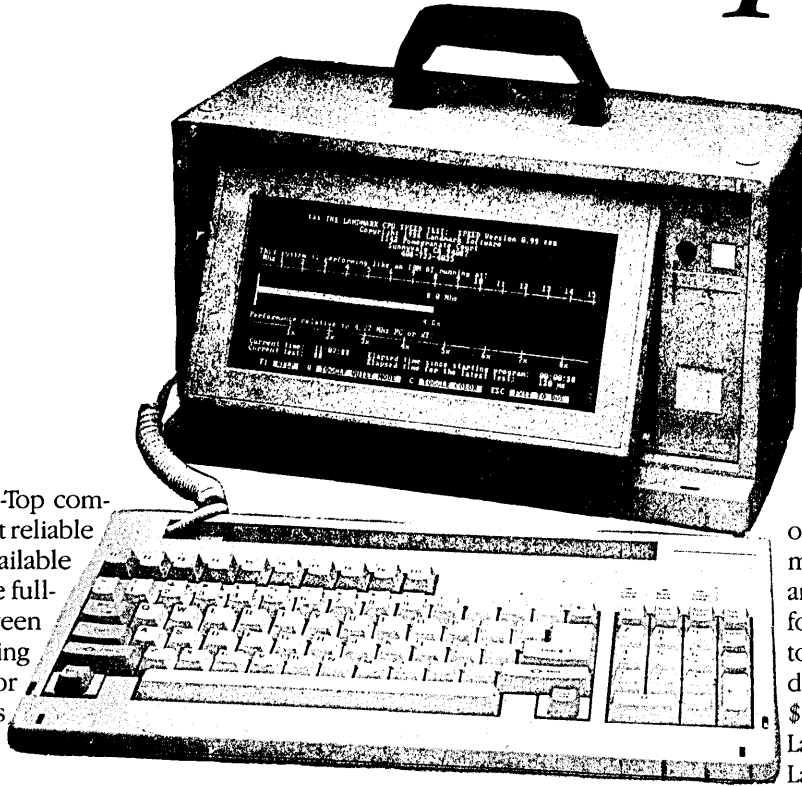
SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY: Th: Std Air \$4/1 lb Fr: P-1 \$10.50/2 lb

MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts μ P ∞ MICROPROCESSORS UNLIMITED, INC.
24,000 S. Peoria Ave., (918) 267-4961
BEGGS, OK. 74421

No minimum order. Please note that prices are subject to change. Shipping & insurance extra, & up to \$1 for packing materials. Orders received by 9 PM CST can usually be delivered the next morning, via Federal Express Standard Air @ \$4.00, or guaranteed next day Priority One @ \$10.50!

Reader Service Number 37

The Ultimate Lap-Top



The McTek286B Lap-Top combines the fastest, most reliable AT motherboard available with the most visible full-size LCD lap-top screen on the market. Running at a switchable 8 or 10 MHz, it includes a 20MB hard disk, 3½" floppy drive, parallel & serial ports, Award 3.01 bios, 640k, turbo indicator LCD & mouse interface. The screen is a fantastically readable,

electroluminescently backlit, 80-column by 25-line LCD with adjustable intensity and screen-angle. It's as readable as a CRT. You can also plug in a digital or analog

color monitor or a digital or composite monochrome monitor. Included also is an external 5¼" floppy port for reading and converting to 3½" disks (5¼" external drive w/case, power supply: \$179 when purchased with Lap-Top). The McTek 286B Lap-Top comes fully assembled with our one-year parts & labor guarantee, and sells for an amazing, complete price of only **\$1799!**

3 MB On-Board AT!

Our McTek 286A is the most integrated AT-compatible to date. It utilizes the highly regarded Chips & Technology chip set, and includes memory upgradable *on board* to 3 megabytes. No more worries about speed compatibility with expanded memory cards! The 8/10MHz, 0-wait state McTek 286A runs at 11.5 Norton SI, and an effective 13.2MHz on the Landmark test. Serial, parallel & game ports are all standard *on board*. With Award 3.01 bios, 640k, 200W power supply, Samsung amber monitor with Hercules-compatible controller, locking case, AT-style keyboard, 1.2MB drive, 20MB Seagate. Assembled & fully tested, with a full one-year warranty. Get in on the most advanced AT-compatible on the market, at the lowest price ever offered! **\$1399!!**

XT Turbos & Supers

640k 4.77/8MHz and 4.77/10 switchable XT turboboards; two 360k floppy-disk drives with controller; one parallel, one serial and one game port; AT-style keyboard; clock, FCC-approved slide-case; eight slots; Hercules-compatible graphics card; amber monitor w/base; fully assembled and tested; one-year parts *and* labor warranty.

\$599 XT Turbo 4.77/8MHz Complete **\$659** Superturbo 4.77/10MHz Complete

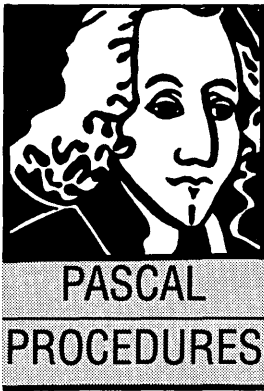
McTek Systems, Inc. • 1411 San Pablo Avenue • Berkeley, CA 94702 • 414-843-0714



DISK DRIVES	PRINTERS	MONITORS	PC/XT	PC/AT	MISC.
Fujitsu 360k\$75	Citizen CD 120 \$159	Samsung amber\$79	640k TurboMothrbrd\$85	McTek286 6/8/10/12MHV \$289	Kingtech Portable Computer Kits: XT/AT (power supply, case, keyboard, monitor) \$380/\$410
Fujitsu 1.2MB\$99	Citizen CD 180 \$189	TVM EGA color \$399	10MHz TurboMothrbrd...\$89	Baby McTek 286B-AT 10/13 O-wait \$409	Eprom burner 4-socket \$139
Teac\$79	HPLASAR Serial2..... \$1799	TVM RGB color \$289	Multi I/O w/disk contrir ..\$59	McTek 286A O-wait 3MB 4 ports on board \$449	Lap-Top Kits \$799
Teac 1.2MB \$105	Epson LX-800 \$219	NEC Multisync \$559	640k RAM card\$39	McTek 286A O-wait 3MB 3 MB Multifunction card \$125	AC power center \$25
Toshiba 3½" 720k \$119	Epson LX-800 \$219	NEC Multisync \$559	2MB Expansion card... \$115	2MB Expansion card... \$125	AC power strips \$15
Toshiba 3½" 720k \$119	Toshiba 321 XL..... \$559	Sony Multiscan \$650	RS232 2-port card\$35	Multi I/O card\$59	Diskette file box \$9
Floppy controller\$22	Call for prices of other brands	HGC-compat.mono card . \$55	4-serial port card\$95	Locking slide case\$65	Printer or serial cable \$8
20MB Hard Disk Kit..... \$289		Color graphic card\$49	Game I/O card\$15	200W power supply\$79	386 16MHz 2MB .. \$1599
30MB Hard Disk Kit..... \$319		EGA Paradise \$159	384k Multifunction card ..\$69	Enhanced keyboard \$59	
ST-225\$219			FCC-app. slide XT case ..\$29	WD HD/floppy controller \$149	
ST-238\$249			150W power supply\$55		
ST-4038\$529			XT keyboard\$49		
ST-251 40MB..... \$429					
	MODEMS	MOUSE			
	Easydata Int. 300/1200 ...\$79	Logimouse C7.....\$75			
	Taiheho external 3/12... \$105				
	Everex 2400 external... \$195				

Reader Service Number 42

*Prices subject to change without notice.



A Pascal Debugger

(For Programmers Who Are Less Than Perfect)

John P. Jones
6245 Columbia Ave.
St. Louis, MO 63139
(314) 645-1596

There's a debugger for C? So what? There are lots of debuggers for C. How about a debugger for Pascal? It's a commercial debugger, but at \$45 it sounds like a steal.

It's April 14 and you've just finished your 15,000 line Turbo Pascal program which insures that Uncle Sam doesn't get more than his fair share. After putting in all the financial data (that you of course have fully organized and ready to input), the program chews on the data for a few seconds, then proudly informs you that you still owe the IRS \$7,324,857.82.

Since you're not Chairman of General Motors, you suspect something's amiss. You've tested each of your subroutines and UNITS during development, so where do you go from here?

Debugging, like the original coding, should be done in a logical and structured manner. First, isolate the problem area to minimize the number of routines that need to be examined. The next step, which is also the least effective, is to reexamine the source. Look for things like parameters mistakenly passed by value, uninitialized variables, missing calculations, and the like.

Next add output statements to indicate the progress of the program. At critical or suspect points, put in a sequence of statements of this form:

```
WriteLn ('In Procedure ''CriticalProc'',  
        Taxes = ', Taxes);  
RestartCh := ReadKey;
```

This technique of course has its disadvantages; it involves repeatedly editing, compiling and running the program to check each section. The program will be larger. Also, the informational writes will most likely interfere with other screen output. If your program uses graphics, it may be very inconvenient to use this method.

Perhaps the most effective tool to use in

debugging is a symbolic source code debugger. Borland has announced that they are working on debuggers for their products. One for Turbo C is due in the first quarter of 1988 with a version for Turbo Pascal to follow.

Until that time, there are alternatives—one of which is from Turbo Power Software of Scotts Valley, California. Tdebug+ version 4.0 (available March 1) provides all the facilities needed to effectively locate any insects (such as giant-winged tax bills) lurking in your code.

What's Needed

What should a source level debugger do? At the very least, it should provide a means to step through the program and examine the changing values of variables at each step. A "window" into the source with an indication of what statement is to be executed is helpful, but at the very least it should display the current source line number.

It should understand a variable's symbolic name rather than its memory address.

Breakpoints and passpoints are next in order of importance—they let you execute at full tilt until execution reaches a certain point (or event).

How is all this possible? The files generated by assemblers and compilers include information about both symbols and the source. The linker, which then combines these files into an executable program, can also produce a "map" file that provides the information to a debugger.

In Turbo Pascal, the compiler puts the information into the .TPU file (linkable object) and the linker can generate a .TPM or special map file. Borland provides a utility to convert .TPM files into standard .MAP files. Only information about global variables and procedures is normally included in these files. Local variables and nested procedures are only visible within their own block and therefore need only be known to the compiler, not the linker.

For Pascal and Modula-2, local variables only exist during the execution of the procedure that contains them. When the procedure is invoked, storage for the variables is allocated

on the stack and then released on return from the procedure. Obviously, a debugger has a problem providing symbolic access to local variables.

Tdebug+

In order to allow symbolic access to local variables and nested procedures, Turbo Power includes a program that patches both of the Turbo Pascal compilers so that local symbol information goes into their output. The resulting files are fully compatible with those produced by the normal compilers but can be double the normal size.

After installing the debugger and compiling your program with the proper switches to provide debug information (\$T+, \$D+), you are ready for the chase. Invoke the debugger with the command line:

```
TDEBUG [OPTIONS] PROGNAME [PARAMETERS]
```

Tdebug will load, then load your program, read the source and .TPM files, and display the debugging screen. This initial screen has two windows: one for the source, which has the next statement to be executed highlighted, and the command window.

Within the text window, you can scroll back and forth through the source, view other source files, and display a symbolic disassembly of the machine code. When in assembly mode, another window opens which shows the current values of all the CPU registers (this window can also be activated in the source mode).

The watch window is opened whenever you "watch" a variable; it is expanded and contracted as watch variables are added or deleted. The watch window is updated periodically to insure the values displayed are correct.

The final window is the memory window. It allows you to monitor an area of memory (normally a data area) and is updated whenever you return to the debug screen.

Permanent breakpoints and passpoints are set with the P command. These are monitored for all the tracing commands. Temporary breakpoints are set either with the tracing commands or with the B command. The B command is used to monitor a variable for change or reaching target value, or to monitor a range of memory for change.

Program execution is initiated/followed with the tracing commands.

- T<n> - executes <n> statements in either source or assembly mode

The assembly mode is fun—it'll teach you a lot about your compiler.

- N - executes to the next statement (procedure calls in one step)
- J - like trace, but always forward in the source
- G<bkpt> - executes until breakpoint set here reached
- GM<bkpt> - like G, but when the set breakpoint is reached, it checks the permanent and temporary breakpoints and stops if any reached
- GT - trace (T) with breakpoint monitor at every statement (slow!)

Tdebug knows about graphics—for all of the above tracing commands the debug screen is swapped with the program's output screen unless the "noswap" option in the command is used. The possible problems you may encounter with some graphics adapters are detailed in the manual (I had no problems in my testing). Each time you return to the debug screen, the program's screen is saved and the debugging windows updated.

Variables can be examined/modified by name or address with the E command. The W command sets up watch variables. The D command opens the memory window and allows viewing of variables in any of 14 different formats. Again, these commands are fully symbolic. The E command automatically determines the type of the variable for its display, but type casting can be used to force a different representation.

The TB command is used to show "how did we get here?" information. It examines the stack and provides a traceback all the way to the main program.

Tdebug+ has some additional features. It can use either extended or expanded memory for its source and symbol buffers; this reduces disk accesses and lets you debug larger programs. It supports keyboard macros; you can assign frequent command sequences to a function or control key. You can also

search the text buffer forwards or backwards to rapidly find a procedure or variable name.

The assembly mode is fun—you can learn a lot about how the compiler operates by using it. Since the original source (if available) is interspersed with the symbolic disassembly, you can see exactly what the compiler did for you. You can also disassemble to the printer or a disk file. This raises the possibility of hand optimization of time critical routines.

I haven't found any bugs in the beta test copy I've been evaluating, but I haven't really been able to push it to its limits either. At \$45 (\$90 with source), I think it's an exceptional buy.

Add Ons

The only computer language with all features for all situations is Ada. It's large, cumbersome, and difficult to learn. Most languages try to provide a balance between features and size and let the programmer create the routines he/she needs for specific problems. Turbo Pascal is one of the languages that provides a good balance, fast and efficient but with sufficient power built-in to make the coding easy.

The lack of some special features has created a market for useful, unique and/or difficult-to-program subroutine libraries. If you are programming to learn, or just for recreation, you should write all your own procedures and units. The best way to learn a language is to use it.

If you are pressed for time, or are programming for pay, it makes more sense to take advantage of the commercial add-ons (why reinvent the wheel?). Also, since source is usually included, the routines can be modified or used as models for your own routines. I'll introduce you to two of the packages I've been looking at recently, and in later columns will look at others.

Borland's Graphix Toolbox

The whole series of toolboxes for Turbo Pascal has been upgraded for use with version 4. This time I'll look at the graphics package.

I must admit I was a bit disappointed with this upgrade; I expected it to be a total revision of the version for Turbo 3. Instead, the package is basically the same, with the exception that it uses linkable .OBJ modules for the display adapters rather than \$include files.

It seems to be a step backwards from the device independent graphics sup-

port of the version 4 compiler package to the device dependent nature of this toolbox. Of course, to be fully display independent, all of the drivers must be present when your program initializes the graphics system, but that's better than needing several versions of the program.

Grumbles aside, if you need support for multiple screen windows in a graphics environment, high-level plotting routines, curve fitting, pie charts or histograms, this package may be for you.

One especially good feature of the window system is the world coordinate concept. Each window can have its own coordinate system, independent of its physical screen coordinates. For instance, a window can be created whose "world" includes only X and Y values between -1 and +1, even though it fills the screen. This can make plotting of real life values easier.

Turbo Professional V 4.0

This package, from the Tdebug+ people, rates an unqualified WOW! In it you'll find routines for:

- Fast windows and screen I/O

- Virtual screens
- PopUp windows
- Pulldown menus
- Line editor (very small but very useful)
- Long (up to 65520 character) strings
- String manipulation
- String formatting
- DOS command line parser
- DOS and BIOS calls
- Interrupt and TSR management routines
- In memory sort routine (fast)
- Keyboard macro processor and editor
- Routines to use both extended and expanded memory
- Large (> 64K) arrays, in normal or EMS memory or in virtual memory
- BCD arithmetic (which Turbo lost in the upgrade to version 4)
- Runtime error recovery (if not fatal, your program can continue)

They include a set of programs that demonstrate the kinds of things you can do with the package—several PopUps, a dandy little source generator for the menu unit, utilities and some impres-

sive window demos.

I've had the product for two weeks and haven't begun to explore all the possibilities. The routines use both Pascal and assembler; complete source is provided for everything. At \$99 list, if you need even a few of the tools this package provides, you can't go wrong.

Next Time

Next time I plan to talk about mixed language programming. Some "foreign" languages are more difficult to integrate than others.

Sources

Turbo Graphix Toolbox
Borland International
 4585 Scotts Valley Dr.
 Scotts Valley, CA 95066
 (408) 438-8400

Tdebug+, Turbo Professional
Turbo Power Software
 3109 Scotts Valley Dr., Suite 122
 Scotts Valley, CA 95066
 (408) 438-8608

◆ ◆ ◆

XEROX 820-1 AND 820-2 ITEMS Reconditioned, Assembled and Tested	
820-1 8" COMPUTER SYSTEM.....	\$330.00
5 1/4" COMPUTER SYSTEM.....	\$350.00
820-2 8" COMPUTER SYSTEM.....	\$395.00
5 1/4" COMPUTER SYSTEM.....	\$415.00
820-1 COMPUTER MONITOR (COMPLETE).....	\$125.00
820-2 COMPUTER MONITOR (COMPLETE W/CONTROLLER).....	\$195.00
820 COMPUTER MONITOR (NO MAIN BOARD).....	\$85.00
HIGH PROFILE KEYBOARD (COMPLETE).....	\$ 45.00
820-1 MAIN COMPUTER BOARD.....	\$ 50.00
FULLY POPULATED BOARDS, AS IS (NEED REPAIR).....	\$ 20.00
820-2 MAIN COMPUTER BOARD.....	\$ 70.00
FULLY POPULATED BOARDS, AS IS (NEED REPAIR).....	\$ 30.00
820-2 FLOPPY CONTROLLER BOARD.....	\$ 95.00
DUAL 8" SSDD DISK DRIVES/ENCLOSURE (COMPLETE).....	\$175.00
DUAL 8" DISK DRIVE CABINET (NO DRIVES).....	\$ 75.00
5 1/4" DUAL DISK DRIVE CABLE.....	\$ 20.00
8" DUAL DISK DRIVE CABLE.....	\$ 35.00
RS-232 CABLES.....	\$ 10.00
LINE CORDS.....ea.	\$3.00
Z80-B 6MHz.....ea.	\$3.00
Z80-H 8MHz.....ea.	\$9.50
5 1/4" DSDD DISKETTES.....ea.	\$.60
8" SSDD DISKETTES.....ea.	\$1.25
DC300A DATA CART..USED.....2/\$5.00	
E2I COMPUTER PRODUCTS 2273 AMERICAN AVE. #8 HAYWARD, CA 94545 (415) 786-9203	
<small>TERMS: Pre-payment, COD, Visa/Mastercard. California residents add sales tax. Orders are FOB Hayward, CA.. Shipments by UPS Ground unless otherwise requested. Prices and availability are subject to change without notice. All products are assembled and tested and have a 30 day warranty unless otherwise stated. Call or write for current product and price listing. Xerox is a trademark of Xerox Corporation. CP/M is a trademark of Digital Research.</small>	


Reader Service Number 33

**DISK FORMAT
CONVERSION**

XenoCopy-PC

\$79.95 + \$5.00 S/H
 Sales Tax if CA.

PC-DOS program
lets your PC



UPS
COD

**READ / WRITE /
FORMAT / DUPLICATE**

Disks from over 300 other micros

**Upgrades available from previous versions
for only \$25.00 Call for Authorization**

To Order Contact:

XENOSOFT™

2210 SIXTH ST. BERKELEY, CA. 94710
 (415) 644-9386

Reader Service Number 39

ERAC CO.

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

AT

Motherboard 6 & 10 Meg
Zero Wait State
8 Expansion Slots
1 Meg RAM On-Board
Math Co-processor Option
Phoenix Bios
200 Watt Power Supply
Hercules Compat. Video Board
Parallel Port
2 Serial Ports Active
Game Port
Clock/Calendar
Hard Disk & Floppy Controller
20M Hard Drive
1.2M 5 1/4" Floppy Drive
360K 5 1/4" Floppy Drive
5061 Keyboard
Case with Turbo & Reset,
Hard Drive Light and
Keyboard Disable Switch
Amber Graphics Monitor

\$1581

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

BABY AT

Motherboard 6 & 10 Meg
Zero Wait State
8 Expansion Slots
80286 Processor
Math Co-Processor Option
1 Meg RAM On-Board
Phoenix Bios
200 Watt Power Supply
Hercules Compat. Video Board
Parallel Board
2 Serial Ports Active
Game Port
Clock/Calendar
Hard Disk & Floppy Controller
20M Hard Drive
1.2M 5 1/4" Floppy Drive
360K 5 1/4" Floppy Drive
5061 Keyboard
Mini AT Case with Turbo &
Reset, Hard Drive Light and
Keyboard Disable Switch
Amber Graphics Monitor

\$1531

EGA ADD \$449
40M HD ADD \$150

XT/TURBO

Motherboard
5 & 8 MHz Switchable
8088 — V20 Optional
Optional Co-processor
8 Expansion Slots
ERSO or Bison Bios
640K RAM
150 Watt Power Supply
Hercules Compat. Video Board
Parallel Board
2 Serial Ports Active
Game Port
Clock/Calendar
Hard Disk and
Floppy Controller
20M 5 1/4" Hard Drive
2 ea. 360K 5 1/4" Floppy Drive
AT Style Keyboard
Standard Slide Case
Amber Graphics Monitor

\$999

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

UNINTERRUPTIBLE POWER SUPPLY \$129

ELGAR MODEL SPR401 — THESE SUPPLIES MAY HAVE SOME MINOR COSMETIC DAMAGE, BUT ARE ELECTRICALLY SOUND. 350VA WAVEFORM RECTANGULAR. RUN ON INTERNAL OR EXTERNAL 24VDC BATTERY WHEN LINE GOES DOWN. TYPICAL TRANSFER TIME = 12MS.

NEW 24V INTERNAL BATTERY \$75

KAYPRO EQUIPMENT

9" Green Monitor\$35.00
Keyboard75.00
Hard Disk Cable Set (4)15.00
PRO-8 Mod. to your board149.00
Host Interface Board15.00

KAYPRO IC'S

81-189 Video Pal\$15.00
81-194 RAM Pal15.00
81-Series Character Gen. ROMs ..10.00
81-Series Monitor ROMs10.00

POWER SUPPLIES

0-8VDC 100A Metered\$249.00
Volt & Current Regulated
5V/1A, -5V/1.2A, 12V/1A,
-12V/1.2A, -24V/0.05A9.90

HOURS: Mon. - Fri. 9 - 6 — Sat. 10 - 4

MINIMUM ORDER — \$15.00

TERMS: VISA, MasterCard, Certified Checks, Money Order, NO COD. Visa and MasterCard add 3%. Personal checks must clear BEFORE we ship. Include shipping charges. California residents add 6% Sales Tax. For more information please write (or call).

CPU & SUPPORT CHIPS

MC68000-8 CPU\$8.00
Z80 CPU75
Z80A CPU1.50
Z80 CTC1.50
Z80A PIO2.00
Z80A SIO5.00
80886.50
8089-36.50
D8284A2.50
4164-151.90
4164-122.10
17936.00
17977.00
ICL7107 LCD Driver7.00
68455.00
VC3524 Switching Regulators5.00
1458 Dual Op-AMP70
LM2877P 4W Stereo Amp Dual2.50
MB81464-152.75
27163.00
27323.25
27643.50
27C128-19.00
74HC0038
74LS12530
74LS37350
74LS17430

SWITCHERS

5V/9.5A, 12V/3.8A, -12V/1.8A\$39.00
5V/3A, 12V/2A, -12V/1.4A19.50
5V/6A, 12V/2A, -12V/1A29.00
5V/6A, 24V/1 1/4A, 12V/1.6A,
-12V/1.6A29.00
5V/10A19.00
5V/20A24.00
5V/30A39.00
5V/75A, 12V/8A, 24V/5A55.00

MISCELLANEOUS

Z80 Controller w/8-bit A/D Conv. ..\$15.95
Nidc Pack 12V/1.5AH6.50
Joystick 4 Switches 1" Knob5.50

TEST EQUIPMENT OSCILLOSCOPES

Phillips 3260E 120 MHz Dual\$975
TEK 7403N/7A18N/7B50A 60 MHz ..695
TEK 455 50 MHz Dual Trace595.00

ANALYZERS

TEK 491 10MHz-40 GHz\$4500.00
Nicolet 500A 1 Hz 0-100 KHz1800
Biomation 805 Waveform Anlyzr ..259.00

DBASE BOOK OF BUSINESS APPLICATIONS by Michael J. Clifford

Reg. \$19.95 **NOW ONLY \$3.95**

So when you linked routines from C and assembler, you had the eight-character limit—due to the assembler, not the C compiler.

The author states that "this limitation [name length] of one C compiler extends to all C compilers (if you want portability)." I've tried to port 8086 assembly code from MASM to Aztec and other 8086 assemblers. They don't all work the same, so we have a similar problem with assembly language.

The code generated from his example obviously comes from an older compiler without any optimization. The `jmp` to a label on the next line always gets removed by even simple peephole optimizers.

The `chkstk` code in MSC may be disabled during compile. This makes the code to call a function smaller. But I've found `chkstk` to be a wonderful aid during debugging. Stack problems can be notoriously difficult to find, especially in code dealing with many interrupts.

I use assembly for speed and C for portability and readability. Code is often maintained by another person than the author and my experience has been that C is far easier to maintain. So use whatever language is appropriate for the task at hand. Assembly language certainly has its place; so does C. And perhaps there's even a good use for COBOL!

Martin Nohr
4152 Rosenbaum Ave.
San Jose, CA 95136

Editor's note: If nothing else, Isaacson's article generated a flood of great letters. I haven't seen so much good-natured disagreement since a state department official suggested statehood for South Africa.

Fractal Error

Thanks for the free sample issue #39, which did indeed have a bunch of worthwhile stuff—I might even subscribe.

Larry Fogg's article on the Mandelbrot and Julia sets was useful, not because of still another bubbly description of the unquestioned beauty of the sets, but because of the simple core programs which tempt one to play.

The programs probably would have been slightly more useful if they had not had a trivial mathematical error. If you want to find the intercolumn spac-

ing for columns ranging from zero to `maxcol`, you divide the range by `maxcol`, not (`maxcol - 1`). Oh, what a familiar error. He did it for rows and columns in both programs and is not the first.

Hal Lewis
Dept. of Physics
University of California
Santa Barbara, CA 93106

Editor's note: Thanks for the correction, Hal. A number of readers have sent in their fractal code, also. They used 8087 assembly, fixed point math, and more efficient algorithms. Look for an article in the near future explaining these techniques.

The RAM Price Debacle

If you've been following RAM prices over the last few months, you know what's been happening. RAMs have been skyrocketing in price at a rate totally unanticipated. Here's a brief chronology, with some whys and wherefores. Keep in mind that I'm quoting volume pricing.

Early 1987-April: The price of 256K DRAMs in large quantities is about \$1.60; 1 Meg DRAMs go for about \$18. A reliable supplier of mine tells me that 1 Meg DRAMs will be less than \$10 by the end of the year.

American DRAM manufacturers, chiefly TI and Micron Technology, have been complaining to their congressmen about alleged DRAM "dumping" by Japanese manufacturers. (Dumping is either described as selling below the cost of production of a product, or selling at a price below the price in the home market.)

Congressmen yelled at the Japanese government, which yelled at Japanese DRAM manufacturers to "stop it." Japanese manufacturers "voluntarily" limited their production, which gradually raised the price of 256K DRAMs to over \$2. Korean manufacturers, primarily Samsung, although not bound by any agreement, raised their price to follow the market.

Mid 1987: The rise in the value of the yen against the dollar continued to put pressure on the semiconductor market. Still, 256K DRAM prices were holding fairly steady at about \$2.25.

4th Quarter 1987: Samsung, the main Korean manufacturer, reaches an out-of-court settlement with TI, which accused Samsung of violating various semicon-

ductor patents with their 256K DRAM. Samsung agrees (not happily) to pay TI fifty cents per part as a royalty; 256K DRAM prices are now about \$2.60, and rising.

End of December, 1987: As usual, Japanese manufacturing and shipping shut down for two weeks at the end of the year. This exacerbates an already bad shortage of parts. By the first work week of January, prices of typical 256K DRAMs have shot up \$1.

January, 1988: DRAMs simply aren't available in large quantities. A broker suggests to me that Samsung doesn't want to pay the patent royalty on DRAMs to TI, so is simply not importing them into the U.S. By mid-January, prices have shot past \$4.50 per chip—1 Megs aren't so bad, "only" about \$28.

In a weird violation of the ironclad rule of RAM sales (slower is cheaper), a broker offers to sell 120 nsec DRAMs for \$4.50, 150 nsec for \$4.65. Why? He's out of stock on 150 nsec, but has plenty of 120 nsec.

End of January: A broker calls me up and offers 256K DRAMs for \$6.50. I don't faint. By this time, I'm numb to the news.

Now (February 3, 1988): No relief in sight, yet I can't imagine this price run-up lasting much longer. There are too many companies which depend on reasonably-priced RAMs.

Open up a current PC-type magazine. Notice the ad prices on 2 meg populated RAM boards (prices which are now probably below their current RAM cost!). Call these guys up, and you'll probably hear either that they're "out of stock," have raised their prices, or both! How can you blame them?

I conclude that only about 20% of the RAM price increase we've seen has been caused by the Yen-dollar exchange ratio, 50% is due to politically-inspired production cuts in Japan, and 30% is caused by opportunism by non-Japanese suppliers.

James Bell
SemiDisk Systems, Inc.
P.O. Box GG
Beaverton, OR 97075

◆ ◆ ◆



CULTURE CORNER

Illustration by Greg Cross



"You know, a simple binary model would work beautifully for this species. There's no doubt that applying bear cognition would be overkill."

THE Z88 UNDER 2 LBS.
A Computer Without Compromise



• Where laptops compromise on display and RAM capacity to achieve portability, and desktops seem to equate price with power, the Z88 is a personal computer which makes no compromises • A CMOS-technology computer with the power to address 4 Mbytes of memory • A computer with a work-free display of 8 lines of 80 characters, an LCD screen which outdates all others, and a unique dynamic page map on screen • A computer with solid-state permanent storage • A computer with advanced word-processing, spreadsheet and ingenious time- and data-management software built-in • A computer which is completely self-contained, which gives you up to 20 hours active computing from just 4 AA batteries, yet which talks and listens to your IBM • A computer with a full-size keyboard, in a package less than the size of an 8 1/2 x 11, with a total weight of less than 2 lbs. • The Z88. A computer without compromise.

Z-88 Computer	\$479.00	SERIAL to PARALLEL I/F	\$ 45.95
PC to Z-88 Linkup	45.00	1/2 MEG. RAM	359.95
32K RAM	39.95	DATABASE S/W	119.95
32K EPROM	39.95	CARRYING CASE	19.95
128K RAM	89.95	Z-88 MAGAZINE	5.00
128K EPROM	89.95	X-MODEM S/W	89.95
RS 232 LEAD	19.95		

\$5 SHIPPING ON Z-88. VISA/MC Accepted with 3% surcharge. C.O.D. charge is \$2.25. If there are any questions, feel free to call or write.

SHARP'S, INC.

Route 10, Box 459
Mechanicville, VA 23111
(804) 746-1664



Add \$5 for shipping and handling.
3% surcharge on credit cards.

ShareWare™

BROWN BAG Software™

HARD DISK MADE EASY

PowerMenu™

- Pop-up User Menus
- File Security
- Colorful and "Goof-Proof"
- Runs Your Program with One Keystroke
- Great for Networks
- All DOS Functions without "Computereze!"
- DOS Window



Copyright 1988, Brown Bag Software
 PowerMenu System v3.00a, Ser # 123456789

PowerMenu™ is Brown Bag Software's DOS operating environment. It creates colorful pop-up stacking menus that run your programs at the press of a single key. Programs can be password protected if you wish and access to "Raw DOS" can be restricted. **PowerMenu™** also features a powerful file manager that displays files and directories. Multiple files can be "Flagged" for "One Pass" copy erase or more. Both novices and Power Users will appreciate **Power Menu's™** speed and ease of use. Takes just 2.5K of RAM! A **ShareWare™** version is available on Compuserve™, Genie™, Delphi™, Bix™, and The Source™ or directly from the Brown Bag BBS.

\$89.95

(408) 571-7854

1-800-523-0764

IN CALIFORNIA 1-800-323-5335 or (408) 559-4545

VISA, MasterCard accepted or mail your check to:
File #116, Box 60000, San Francisco, CA 94160-1719

SHAREWARE DISK AVAILABLE FOR \$10.00

Reader Service Number 87

CP/M Notes

\$19.95 Real Time Clock

My Kaypro 2-84 is equipped with Micro C's Pro 884 Max ROM. Recently I had the lid off, performing the drive step rate mod and noticed several empty sockets and missing components. Checking the schematic, it appeared that all I needed to have a clock were these parts:

- Z80A-PIO (put in position U35)
- MM58167A clock chip (U36)
- 32.768 KHz crystal (Y4)
- 1N4148 diodes (two at CR6 and CR7)
- 0.1 uF disk capacitor (C54)
- 22 pF disk capacitors (two at C64 and C65)
- 3 volt battery and holder (BT1)

However, there were a couple of problems with the schematic. Diode CR6 is shown as a "1001". The only thing the local radio shop had with that number was a transistor. The board is drilled and printed identically for CR6 and CR7, with CR7 shown as a 1N4148 diode. Since the diodes appeared to have similar functions, I assumed a 1N4148 for both.

The schematic also calls for 20 pF capacitors. That ain't no sich of a thang! I used 22s.

All of the above parts are available from JDR Microdevices for \$19.95 including shipping. A little overkill here—the 1N4148s come 25 for a dollar.

Also, the 3 volt lithium battery holder doesn't match the board holes exactly. I managed to jimmy it in place using one of the snipped capacitor leads as an extension to the plus terminal. Anyone finding a better fit can leave \$3.45 off the order to JDR.

I had no difficulty soldering in (gingerly!) all these parts from the top of the board. But use the solder sparingly. You can unplug the holes by touch-

ing them with a soldering iron and immediately plunging in a wooden toothpick (if there's an easier way, I don't know it).

MCONFIG (from the Pro 884 Max utilities disk) allows setting and display of the clock. Other Micro C disks (K46, K49) have such routines for those without the ROM.

Finally, if anyone tries this mod and fries their board, I will disavow all knowledge of their actions.

JDR Microdevices
110 Knowles Dr.
Los Gatos, CA 95030

Billy Guthrie
PO Box 8184
Gadsden, AL 35902

Editor's note: The universal board in the 2-84 Kaypros not only has room for the clock (thanks Billy), but for the internal modem as well. We made an attempt to chase down the parts a while back with somewhat discouraging results. If anyone has done the modem mod and has a good source of parts, let us know.

A 1N4148 works fine in place of the 1001. They're both just low power diodes. A solder sucker will also clean out the circuit board holes but a toothpick is cheaper.

A Plea From Indonesia

I am an Australian who this year commenced a two year stint on an Australian Foreign Aid program in Indonesia. Sources of useful information and public domain software are very difficult to find here. Mind you, *software* is available for about 30 cents plus the cost of the disk. Sickening, although here there's nothing illegal about it.

Having been president of a Kaypro user group in Australia, I am always very careful about piracy issues. Here, anyone concerned about piracy is a

laughing stock.

I would be very keen to hear from anyone who has overcome problems with the clock in the 84 series Kaypro 10. I bought the machine partly because it had the clock. Since new, it has jumped hours between power down and power up. Kaypro in Australia looked at it under warranty. But, as it was one of the first they had seen, they couldn't help (although they were most obliging).

I bought the Advent "clock fix" board, but it didn't make any difference. Advent very promptly refunded my money. I have the Advent Turbo ROM but the benefit of on-screen time display is lost. Has anyone any comments?

Has anyone ever seen a public domain multiformat program (like MFDISK) for Kaypros which has source included? I'm surprised that there doesn't seem to be anything available. I would be most interested in hearing from anyone, especially as I am rather isolated here in Indonesia.

Dr D. Yates, IPB-Aust. Project
Jakarta Bag
Locked Bag 40
Queen Victoria Tce
Parkes, ACT, 2600
Australia

Editor's guess: You've moved the Kaypro through so many time zones that it's jumpy. Or your board might be having an AI problem.

◆ ◆ ◆



Small C Utilities, Continued

Stephen S. Mitchell

320 King St., Suite 506
Alexandria, VA 22314
(703) 360-4659

Small C has been a part of the computer community longer than most computers. In this final look at Small C utilities, Steve brings us details on special things to watch for in printf() and the ASM assembler.

In *Micro C* Issue #21 (Dec-Jan 1985), Eric Sosman presented a BASIC program to strip the comments from an assembly language file. The program works fine. However, if you are serious about learning to program in C, why not write your comment stripper in C. Hence, NOCMNT.C (see Figure 1), which does precisely that.

The algorithm is essentially a straightforward translation of Sosman's BASIC into C, coupled with part of Kernighan and Ritchie's code for removing trailing white space.

In order to reproduce Sosman's logic, both C's "continue" and "break" statements were used. However, rather than opening and closing files, I relied on input/output redirection (which the compiler on K7 supports, but Fred Scacchitti's version does not).

Actually, I agree with Fred's viewpoint in that CP/M is not UNIX, and I would prefer a command line syntax such as:

```
A>nocmnt fatfile.asm slimfile.asm
Instead of:
A>nocmnt <fatfile.asm >slimfile.asm
```

However, redirection is very convenient when you are developing a program. Test input can come from the keyboard and you can immediately see the result on the screen. This makes debugging much easier.

Also, sometimes the code to fetch and validate file name arguments can be as complex as the rest of the program. (If there is no such file, do you simply abort or do you give the user a second chance? Should you show the user a directory? Of which disk...)

For this reason, if the program is only used occasionally, I don't mind typing in the < and >

redirection operators. If it is an everyday utility, however, I would recode with VIEW.C to check for file name arguments as part of the command line. And failing that, to prompt for them.

Since the output of the program is going to "stdout," which is more likely to be redirected to a file than sent to the screen, we must send operator messages to "stderr" (which is always the console). This avoids their burial within the output file.

Because fgets() in IOLIB.ASM was still buggy, I used getline() from Kernighan and Ritchie to fetch the input lines. Besides, getline() returns the length of the text line it fetches, eliminating the need for strlen(). As in VIEW.C, none of the functions in LIBASM.C are needed, nor is LIBASM.C appended.

Tabbing

ENTAB.C (see Figure 2), which replaces spaces with equivalent tabs, takes the same approach as NOCMNT.C with respect to input and output.

Note that this version of the compiler does not support redirection to or from a device, only to or from a file. For example, if you attempt to send the output to a printer with >LST:, you will get a disk file named "LST:," but you will not be able to TYPE, RENAME, ERASE, or PIP. Most aggravating.

You can, however, use a disk editor like DU to go into the directory and change the name, byte by byte, to something CP/M recognizes as a file name.

The algorithm comes from Software Tools, tweaked a bit by James Hendrix to allow for the possibility of tabs and backspaces in the input. A fixed tab size of 8 (the standard CP/M tab interval) is used to simplify the program. If you really need to "entab" a file with other tab sizes, just retrieve the value from the command line or prompt for it.

In either event, the obtained value could be converted to an integer value with the function atoi() in LIBASM.C and assigned to the variable "tabsize." Currently, however, LIBASM.C

Figure 1—NOCMNT vers. 6/30/87

```

/*
 * Removes comments, trailing white space, and blank
 * lines from assembly language source files.
 *
 * Written for Small-C compiler vers. 2.03 (ASM)
 *
 * Note: does not properly handle literal semicolons
 * such as in DB ';' or CPI ';' or MVI A,';'.
 */

#include <stdio.h>
#include "iolib.asm"
#include "call.asm"

#define MAXLINE 127
#define NOCCARGC
#define NOTFOUND -1

char line[MAXLINE + 1];
main() {

    int len, i, j;

    fputs("NOCMNT -- strips comments from .ASM files.\n",stderr);
    fputs("Usage: nocmnt <fatfile.asm >slimfile.asm\n",stderr);
    while ((len = getline(line, MAXLINE, stdin)) != NULL) {
        i = index(line, ';');
        if (i == NOTFOUND) i = len;
        while (--i >= 0) /* find last non-blank char */
            if ((line[i] != ' ')
                && (line[i] != '\t')
                && (line[i] != '\n'))
                break;
        if (i == -1) continue; /* line is blank, so ignore it */
        line[i+1] = '\0'; /* otherwise mark new end */
        fputs(line,stdout); /* and send string to output */
        fputc('\n',stdout); /* followed by newline char */
    }
}

/*
 * Return position (i.e., array index) of first occurrence
 * of c in str, else -1.
 */

index(str, c) char *str, c; {
    int i;
    i = 0;
    while(*str) {
        if(*str == c) return (i);
        ++str;
        ++i;
    }
    return (-1);
}

/*
 * Fetch a line of input from fd and store it
 * (including any terminating newline) in s.
 * Return the length of the fetched line. From
 * Kernighan and Ritchie, The C Programming
 * Language, copyright 1978.
 */

getline(s, lim, fd) char s[]; int lim, fd; {
    int c, i;
    i=0;
    while (--lim > 0)
        && ((c=fgetc(fd)) != EOF)
        && (c != '\n')
            s[i++] = c;
    if (c == '\n')
        s[i++] = c;
    s[i] = '\0';
    return(i);
}
END OF LISTING

```

is not needed and is not appended.

Avoiding The Singles Scene

The best things in life often come in pairs—braces and brackets in C programs, for instance; quotation marks and parentheses in ordinary English text; and ^S and ^B underline and boldface controls in WordStar files.

PAIR.C is designed to keep your files honest. It makes sure that items which *should* be paired, are. To do this, it reads the file character by character, counting braces, brackets, and so on. When it's finished, it determines whether the number of left-hand characters equal those on the right. (Or, in the case of quotation marks or print controls, which don't have a right and left, whether the total is evenly divisible by two.)

Editor's note: PAIR and COUNT (which follows) are available in the Issue #41 file on the Micro C RBBS, 503-382-7643, and on the Micro C Issue #41 disk. The disk is \$6.00 for U.S. subscribers, \$8.00 for everyone else, ppd. To order, use the prepaid order form in this magazine or call 1-800-888-8087.

It either reports any discrepancies or pronounces the file healthy. As with VIEW, you can specify the file to be checked on the command line. If you forget or fail to do so, the program asks.

Note the use of #define and #ifdef to control conditional compilation. If WordStar isn't used, there's obviously no point in looking for ^S's and ^B's. Simply omit (or comment out) the line #define WORDSTAR. This will ensure that the code between each #ifdef WORDSTAR and its terminating #endif won't be compiled.

You can easily compile two different versions of PAIR—one with WORDSTAR defined for checking your word processing files, and one with WORDSTAR not defined for checking your C program source files.

Either way, each character fetched from the input file is tested using a "switch" statement. A series of "if ... else if ... else if ... else if" tests could have been used instead, but the switch statement produces more efficient code.

Note that the counting variables rparen, lparen, etc., are declared globally (outside the main() function), while ch is declared locally. The Small-C compiler generally produces more efficient code fetching and retrieving global variables than it does with locals. (Except for the last two local variables declared. They can be pushed and popped on and

off the stack.)

Since `ch` is so heavily used, I declared it locally.

Names

I got bitten by two of my variable names. In my first attempt at the PAIR program, I named the counters for `^S` and `^B` as `ctrl_s` and `ctrl_b`, respectively. The compiler had absolutely no problem with these names, which are after all perfectly legal C names. But global variables create assembly language labels of the same name. And ASM doesn't like the underline character.

Had `ctrl_s` and `ctrl_b` been local, however, this would not have been a problem. Local variables are stored on the stack and there is no assembly language label created. In this case, I left them global for the sake of efficiency and simply changed the name to a form ASM could live with. Incidentally, Hendrix's book has an extensive chapter on Small-C efficiency considerations. I recommend it highly.

Editor's note: A limited supply of "The Small-C Handbook" by James E. Hendrix is available from Micro C for \$17.95, postage paid in the U.S.

Screen Output

The messages to the screen could have been output more simply by using `puts()` rather than `fputs()`. However, the `puts()` function supplied in `IOLIB.ASM` has a bug which prevents it from responding properly to the newline escape sequence `\n`. Additionally, it does not automatically append a newline to the string being output.

You can fix the bug (see Figure 3), but remember, `puts()` still remains non-standard (no automatic newline). However, since you can force a newline with the `\n` escape sequence, that should not present a problem.

PAIR compiles to a 5K `.COM` file. Given an 18K text file to examine, it ran through it in 19 seconds on an unmodified Kaypro II with 2.5 MH clock. By comparison, I have a low-priced commercial C compiler I play around with (particularly when floats or structures are needed, neither of which Small-C offers). Using it, the same program compiled to a 16K (!) `.COM` file and took 50 seconds.

Even though this particular package is not the most highly-touted C compiler ever to come down the pike, the figures do serve to show that just because Small-C is public domain doesn't

Figure 2—ENTAB -- replace blanks with equivalent tabs

```
/* Based on Small-Tools program of the same name,
 * copyright 1982 by J. E. Hendrix, in turn based
 * on algorithm in Software Tools by Kernighan and
 * Plauger, copyright 1976.
 *
 * Written for Small-C compiler vers. 2.03 (ASM)
 *
 * Date: 6/27/87
 */

#include <stdio.h>
#include "iolib.asm"
#include "call.asm"

#define NOCCARGC
#define MAXLINE 127

char tabs[MAXLINE + 1];
int tabsize = 8;

main() {
    char c;
    int col, newcol;

    fputs("ENTAB - replaces blanks with tabs.\n", stderr);
    fputs("Usage: entab <oldfile.txt >newfile.txt\n", stderr);
    settabs();
    col=1;
    while (YES) {
        newcol = col;
        while ((c = getchar()) == ' ') {
            ++newcol;
            if (tabpos(newcol, tabs) == YES) {
                fputc('\t', stdout);
                col = newcol;
            }
        }
        if (c == '\t') {
            while (tabpos(newcol, tabs) == NO)
                ++newcol;
            fputc('\t', stdout);
            col = newcol;
            continue;
        }
        while (col < newcol) {
            fputc(' ', stdout);
            ++col;
        }
        if (c == EOF) break;
        fputc(c, stdout);
        if (c == '\n')
            col = 1;
        else if (c == '\b')
            --col;
        else ++col;
    }
}

/*
 * set up array tabs[] with tabs at interval tabsize
 */

settabs() {
    int i;
    for (i = 0; i < MAXLINE; ++i)
        tabs[i] = NO;
    for (i = 0; i < MAXLINE; ++i)
        if ((i % tabsize) == 1) tabs[i] = YES;
}

/*
 * return YES if col is a tab stop else NO
 */

tabpos(col, tabs) int col; char tabs[]; {
    if (col > MAXLINE) return YES;
    else return tabs[col];
}

END OF LISTING
```

mean it generates second-rate code.

I've deliberately omitted a comparison of compilation times where, not

surprisingly, the commercial package (which includes a linking loader) is significantly faster. For an application like

PAIR, which I might call on several times a day, however, I'm more than willing to trade an extra couple minutes of compilation for the additional speed and the smaller code size.

The Virtues Of Compactness

The question of size brings me to that tantalizing suggestion by the author of SAMPLE.DOC, who says, "As you grow with C you will learn how to reduce the size of your final programs." I know I'm not the only one who regrets that no clues were offered as to what the technique was. (Presumably it was left as an exercise for the reader.)

Of course, you can reduce the size of the compiler's .ASM output by slimming down the runtime library files. After that, probably the best way to keep your object files small is to avoid appending LIBASM.C and use printf() as little as possible.

Without a linker, it's a real pain to have to #include needed routines from the C library. Especially since you have to be careful about functions that in turn call other functions (as both printf() and dtoi() do). But simply appending LIBASM.C to your source file will add 3K to your final program.

The biggest offender is printf(). Avoiding it altogether (or at least reserving it for those programs where the alternative would be truly cumbersome or inconvenient) will pay handsome dividends in terms of reduced program size.

When you think about it, how often

do you really need all the flexibility of printf()? Most of the time you can output a line by repeated calls to fputs(), fputc(), and the Small-C conversion functions itod(), itox(), and itou().

Obviously, it's a lot handier to write:

```
printf("Words = %d\n", words);
```

than it is to muddle your way through:

```
fputs("Words = ", stdout);
fputs(itod(words, numstr, 6), stdout);
fputc('\n', stdout);
```

But the latter eliminates the considerable overhead associated with printf(), and will go a long way toward keeping your programs compact.

printf() Replacement

You can have formatted output without printf(), however. COUNT.C (also on the Micro C RBBS and Issue #41 disk) is an adaptation of the familiar word counting program from *The C Programming Language*.

Since Small-C has short integers, and since the number of characters in even a moderately large text file could well exceed the maximum (signed) integer value of 32,767, I declared the counting variables to be character pointers (even though they are used like integers). Small-C, while it does not have an explicit unsigned integer type, treats pointers as unsigned integers, which can have a maximum value of 65,535.

To print out these unsigned values, I

used the function itou() from the Small-C library along with a separate function putnum(). The interface buries some of the complexity and makes it easier to understand.

Epilogue

Well, that's it—enough examples for a month of Sundays. They won't necessarily make you an expert Small-C programmer, but a least you should be pointed in the right direction. Now you can start working your way through the exercises in the many excellent books on C programming.

Proficiency in programming, as in any other worthwhile human endeavor, comes with practice. So don't just read about C programming—fire up your editor and start writing.

No matter how many mistakes you make along the way—and I've made just about every one, often more than once—there's still no thrill like seeing the source code you've labored and puzzled over come alive, ready to do your bidding at the press of a few keystrokes.

Of course, the thrill is there in any language, but Small-C is special. It belongs to the hacker community and not to the commercial interests. By using it, you're helping to keep a remarkable tradition alive.

◆ ◆ ◆

Figure 3—Small-C Vers. 2.03 (ASM) Bug Fixes

(1) The puts() function supplied as part of IOLIB.ASM does not respond properly to the newline escape sequence \n, sending only a bare carriage return rather than the carriage return/line feed pair that CP/M expects. (The functions fputs(), putchar(), and fputc(), by contrast, all respond properly to \n). To correct this problem, edit the file IOLIB.ASM to add the instructions shown here:

```
; added instructions are marked with *
;
PUTS1:
    POP    H
    MOV    A,M
    ORA    A
    JZ     PUTSRET
    MOV    E,M
    INX    H
    PUSH   H
    PUSH   D        ; * save char on stack
    MVI    C,PUTCH
    CALL   BDOS
    POP    D        ; * retrieve char
    MOV    A,E      ; *
    ANI    7FH      ; * mask parity bit
    CPI    EOL      ; * is it a CR?
    JNZ    PUTS1    ; * no, loop for next char
    MVI    E,LF     ; * else send line feed
    MVI    C,PUTCH  ; *
    CALL   BDOS     ; *
```

```
JMP     PUTS1
PUTSRET:
    RET
```

(2) When reading a disk file, fgets() fails to terminate as it is supposed to when a carriage return is encountered, and continues reading characters until the maximum number of characters is read. (This plays havoc with programs doing line-oriented input.) To cure the problem, change the one instruction in the file IOLIB.ASM as shown:

```
; changed instruction marked with *
;
FGETS3:
    MOV    A,L
    POP    H
    MOV    M,A
    INX    H
    ANI    7FH
    CPI    EOL
        ;*was CPI LF
    JNZ    FGETS2
```

(The problem arises because FGETS fetches characters with calls to GETC, which in turn 'swallows' the character—assumed to be a line feed—following a carriage return. Thus CPI LF is never true.)

END OF LISTING

Later

Just the other day, I was showing a fellow through the office. He was thinning a bit at the top and his quiet demeanor reminded me of the old man at the coast.

I proudly pointed out our new clones, the ones with the high-power processors, copious RAM, winchesters, everything that technology could offer. Then he noticed my Big Board.

"That's an early CP/M machine," I told him.

He brightened a bit and ran his fingers over the dusty circuit board.

"A friend had one of these. Once."

SOG Fly-In

Speaking of flying, Bill Davidson (PC Tech) will be bringing his Bonanza to SOG. Cecil Stump will no doubt be showing up in his 172. So, they're organizing a fly-in.

On July 14-16 Bend International Airstrip will host the SOG VII fly-in. There'll be a couple of pilot rental cars at the airport (large, old, station wagons), space to park your plane, room to sleep under the wing, and a chemical john (a nice one) available when the FBO is closed.

If the weather's nasty (extremely unlikely) we'll throw my Stinson out of the hangar so you'll have a dry space for sleeping.

Bend has a published VOR approach; Redmond (10 miles north) has ILS and FAA flight service (makes it a good alternate). Bend has 100 low lead and Jet A (for the Stinson, of course). Redmond has 100LL, Jet A and unleaded auto fuel (at Butler Aviation).

Bend's runway is paved, north-south (16-34), 5,000 ft. long, 3200 ft. elevation. Bend unicom is 123.0 and is monitored by the FBO during daylight hours. Amateur radio operators who are flying in might also monitor 146.52, simplex. Ground bound hams should monitor the Bend repeater, 146.94-34.

If you'd like to join the fly-in, call Bill Davidson at PC-Tech (612) 345-4555. He'll be coordinating transportation to and from the college and he's organizing some social to-do's.

Other Ways To Bend

You can drive to Bend if you live on the West Coast, or you can fly into a commercial airport and drive. We're 11 hours by car from San Francisco, 6 hours from Seattle, 5 hours from Boise, Idaho, 3 hours from Portland, Oregon, 2 1/2 hours from Eugene, Oregon, or 15 minutes from Redmond, Oregon.

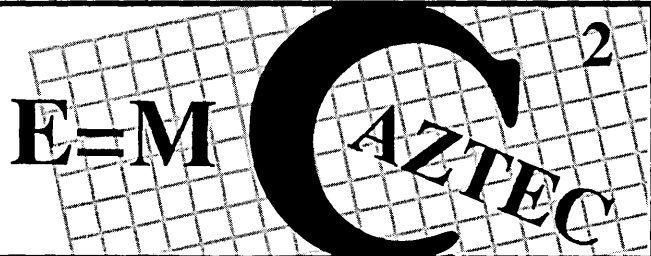
Redmond has the only commercial airport in the area and it's served by three regional airlines. However, you'll often pay more to get from Portland to Redmond than from New York to Portland.

A lot of people fly to Portland, rent a car, and drive to Bend. There are two major routes from Portland to Bend and both are beautiful drives across the Cascade Range. The prettiest follows Interstate 5 from Portland to Salem and then Highway 20 from Salem to Bend. Reserve a couple of extra hours for exploring and picnicking along the North Santiam River.

Graphics Snafus

Desktop publishing is going along quite well. At least the text part is. The graphics end still has problems.

I purchased a Microtek MSF-300C scanner after our borrowed HP scanner was repossessed. I chose the Microtek because it was Editor's choice in PC Magazine.



Genius Begins With A Great Idea...

**FREE
2 DAY
DELIVERY**

Aztec C86 4.1
New PC/MS-DOS
CP/M-86 • ROM

Aztec ROM Systems
6502/65C02 • 8080/Z80
8086/80x86 • 680x0

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under \$1,000 outperform systems priced at over \$10,000.

Aztec C86-p.....\$199

• optimized C with near, far, huge, small, and large memory - Inline assembler - Inline 8087/80287 - ANSI support - Fast Float (32 bit) - optimization options • Manx Aztec 8086/80x86 macro assembler • Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Initial Host Plus Target...\$ 750
Additional Targets.....\$ 500
ROM Support Package....\$ 500

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

Cross Development

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

Aztec C86-d.....\$299

• includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c.....\$499

• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

CP/M • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Third Party Software

A large array of support software is available for Aztec C86. Essential Graphics • C Essentials • C Utility Library • Greenleaf Com. • Greenleaf General • Halo • Panel • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data • C terp • db_Vista • Phact • Plink86Plus • C-tree.

Aztec C II-c CP/M & ROM....\$349

Aztec C II-d CP/M.....\$199

C' Prime

PC/MS-DOS • Macintosh
Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as \$500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more. Special discounts are available for use as course material.

C' Prime\$75

How To Become A User

To become an Aztec C user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, Master Card, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

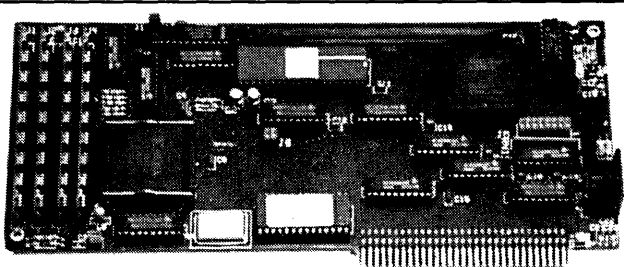
Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee. Most systems are upgradable by paying the difference in price plus \$10. Site licenses, OEM, educational, and multiple copy discounts are available.



To order or for information call today.
1-800-221-0440
In NJ or international call (201) 542-2121
TELEX 4995812

Manx Software Systems
One Industrial Way
Eatontown, NJ 07724

A Reliable PC/XT Compatible For The Corner Stone of Your Products



Announcing The SLY40-XT

The SLY40-XT is a small (4-1/4" by 9-1/4"), four layer card featuring all of the PC/XT mother board functions. The board simply plugs into a passive back plane or SLICER'S 10 slot bus board.

- High Integration — Composed of just 17 Low Power CMOS ICS
- NEC's 8 MHZ V40
- One Megabyte of Zero Wait State RAM
- 8087 Co-Processor Socket
- Standard Keyboard Connector
- Slicer's Own Bios, Source Code Included
- Ideal For Tough Industrial, OEM and Portable Applications
- American Made and Fully Supported by Slicer

- Complete SLY40-XT System with 20 MEG Hard Disk — Just 1299.95, Retail
- Without Hard Disk — Just 995.95, Retail

Ask About Our Complete Line of Computer Products and Accessories!

MasterCard, Visa, Check, Money Order, or C.O.D.
Allow four weeks for delivery.
Prices subject to change without notice.

NOTE NEW ADDRESS & PHONE NO.

Slicer Computers Inc.
3450 Snelling Ave. So.
Minneapolis, MN 55406
612/724-2710
Telex 501357
SLICER UD

PC and XT Are Trademarks of International Business Machines

Reader Service Number 19

The scanner works fine but the software which comes with it stinks. I've called Microtek over a dozen times. Each time they've told me they'd send something that would help; only once did I receive anything and it was for the Mac.

Bugs: The Genius display driver doesn't work. The software can't generate a usable paintbrush (PCX) file. Saving the whole scanned image in Eystar format (its native mode) occasionally destroys the hard drive's FAT. Changing the scan resolution or size often causes an "illegal size" error message.

It doesn't matter what you do after that, the program won't work. And, the manual is so opaque that it took me almost two weeks to generate my first image. The Microtek may be the PC editors' choice, but it isn't the *Micro C* editor's choice.

The HP's software was so easy to use that I was generating images within ten minutes. And, it never ate a FAT or garbled an image.

I've tried to find a paint program or other package which supported the MSF 300-C. No luck. (They all support the HP.)

Today I received some welcome news. Microtek will be releasing a bug fix in March. Unfortunately, I'll have to pay \$49 for the fixed version or sign up for one year of support for \$150. (Meanwhile, HP is sending out new software with new features, free to registered owners. And the HP scanner would have been cheaper.)

Microtek also sent a package called Gemmate. (No documentation, no read-me file, just a disk.) Very little help from tech support. It turns out that Gemmate lets you run the scanner while inside Ventura. Great. However, Gemmate won't give you even a rough look at the image so you're scanning blind. Thanks, but no thanks.

Reflection

Out of the blue (where else?) came a care package from IMSI. In that package was a small translation program called Reflection and Reflection cured my tif with Microtek.

Microtek's Eystar package generates reasonable .TIF files. Reflection can turn .TIF files (despite a few erroneous characters left by Eystar) into .PCX files. Once I have .PCX files, I can edit the images with Publisher's Paintbrush and dump them into Ventura. Hooray.

If all this sounds more complicated than it should be, you're right. But it's workable because Reflection runs very easily.

By the way, Reflection translates any of the following to any of the following: Amiga ILBM, Compuserve GIF, Halo CUT, H.P. PCL, Inset PIX, Lotus PIC, MACPAINT, PC Paintbrush PCX, text, TIF.

Reflection (\$89.00)

IMSI
1299 Fourth St.
San Rafael, CA 94901

Theoretical Issue

If this issue doesn't give your brain a charley horse then you're either just out of school (which means your brain already has a charley horse) or you're a lot like that perverse associate editor Gary Entsminger.

It's not that this isn't an interesting issue. It's quite interesting. But then *Micro C* has become famous for mucking around in the fun, cheap, practical crannies of computing. We're project oriented, not theoretical.

But once in awhile, a little theoretical is good for you. (Like

salad.) It stretches your brain, exercises your imagination, helps you grow hair in all the right places.

This was supposed to be an AI issue. It is, sort of. But it's much broader than that and I'm glad. I think that classic AI is in the doldrums right now. I think people are afraid of it.

Meanwhile our C issue (Issue #40) generated more excitement, more feedback, more activity on the RBBS, more mail, more newsstand sales, more new subscribers from word of mouth, more everything than anything we've ever done. And the hottest thing in the issue was Gary Mellor's Turbo C debugger. Thanks Gary.

More Of Last Issue

"What's the page number for the culture corner?"

"Culture Corner?"

Last issue was a killer: missing columns (two), a shortage of graphics, confusing page layouts, incorrect page numbers...

"Can we skip to issue 41?"

"Nope. Somebody'd notice."

If you've been following the continuing saga of *Micro C* and Desktop Publishing, you might expect me to take a poke at Ventura. I can't. It worked better than ever.

Our output was almost untouched from laser typesetter to printer. Our turnaround was faster than ever. In two weeks we'd Ventura'd the articles, laid out the magazine, typeset the pages, pasted everything together and shipped it off to Michigan for printing. (Michigan? Hey Marge, *Micro C's* being printed in Michigan.)

The problem was human. Very human. Two weeks aren't enough when you're as imprecise, scatterbrained, and pushed as I am at deadline time. I'm an idea person. You want an idea? Here. You want something edited? Great.

You want the table of contents to point to the right page? (We're talking infinite precision.) Find someone else.

Fortunately I'm surrounded by people who cope pretty well with deadlines. They flagged most of the errors before those errors saw the dark of ink—but Culture Corner and CP/M Notes were misplaced. Oh, we've found them, but they didn't make issue #40.

Next Issue

National Advancement Corp. (they teach seminars on servicing XT's, AT's, etc.) says they've discovered a fix for flaky Seagates. You and I should see a major repair article from them (including the Seagate fix) in issue #42.

FastCopy

Every once in awhile a treasure shows up in the mail. Sometimes it's a large, four-color box with manual, disks, and a cover letter from a corporate officer. Other times it's a disk or two in a plain brown mailer.

FastCopy came in plain brown.

It came from Jim Nech, its author, and it's super.

It's easy to use, very fast, powerful, everything. It's everything that MS-DOS's diskcopy should have been.

For instance, you can load a disk into memory, set the number of copies you want and start feeding blanks. It formats the destination disk if necessary (automatically), will verify the data, can alternate between two floppy drives (so you can put a disk in one while it's copying data to the other), and it'll do it all very quickly.

If I were a user group librarian or distributor of user disks (oh yeah, I forgot) and couldn't afford one of those fancy

Complete 8MHz Monochrome System \$995

- Fully compatible with IBM AT™
- Intel 80286 CPU; 80287 socket
- 512K Memory on 1MB motherboard
- HD/FD controller; Battery Backed Clock/Calendar
- 1.2 MB Teac Floppy Drive; MaxiSwitch AT Keyboard
- FCC Class B approved
- 48 hour factory burn-in and testing
- 12" Amber Monitor (720x350) with Tilt/Swivel
- Hercules Compatible Monographics video card
- 200-page Documentation and User Manual
- Designed and Made in U.S.A. with 1 year warranty

Call for our AT-386 machine (4075 Dhrystones)

Options:

10MHz Upgrade	\$150	14" EGA Monitor/Card	Add \$399
20MB XT Kit, ST225	\$279	Taxan 770 Multisync	\$499
30MB XT Kit, ST238	\$309	2MB EMS Card	\$129
Seagate ST225, 20MB	\$249	EVGA 1280x600	\$295
Seagate ST251, 40MB	\$449	Everex 1200B modem	\$89
Miniscribe 6053, 43MB	\$595	Everex 2400B modem	\$179
Micropolis 70MB 28ms	\$895	Software	Call

All hardware/boards include manuals and software for easy installation. Call for further information and other products. Ask for our catalog.

AmTech Computers

3701 Guadalupe St., Suite 103, Austin, Texas 78705
(512) 451-0921

Terms: Cashier's Check, Money Order, VISA/MC (3%), personal checks (allow 10 days to clear). Shipping will be added. Prices and availability subject to change without notice. Texas residents add 7% Tax.

Reader Service Number 44

New from MSC

\$99.99

NanoLISP

A Common LISP Interpreter for MS-DOS

NanoLISP contains a large useful subset of the Common LISP standard, including most Common LISP operations, and adheres *precisely* to the specifications of the standard.

- **Helpful Features:** extremely thorough error-checking, explicit error messages, excellent debugging facilities
- **Advanced Features:** lexical and dynamic scoping, closures, lambda-list keywords, structures, bit-arrays, generic sequence functions, transcendental functions, output formatting
- **UnCommon Features:** graphics, low-level DOS access, customization
- Sample AI application programs
- Fully-indexed 150-page reference manual
- Unlimited free technical support

Free shipping on prepaid orders.

MSC Microcomputer
Systems
Consultants

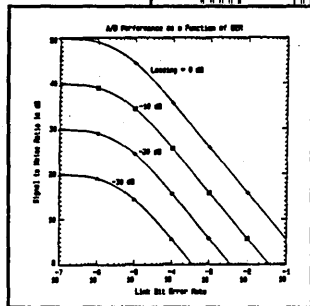
P.O. Box 747, Santa Barbara, CA 93102
(805) 963-3412

Reader Service Number 36

SCIENTIFIC GRAPHICS

Presentation Quality Graphics
For Printers and Plotters

Screen Graphs for
Fast Previews



Curve-Smoothing
Interpolations

Legends Placed Anywhere

Built-In Editor Auto/Manual Scaling Log/Lin/SemiLog

An Indispensable Tool For Technical Professionals

Special
Introductory **\$79**
Price

System Requirements: IBM-PC, XT, AT or Compatible running DOS 2.0 or higher. Screen graphs require graphics card. Printer Graphs require Epson EX, FX, JX, RX, HS; Star Gemini, Rudix, SD, SG, SR; IBM Graphics; or compatibility with one of the above. Plotter graphs require HP-GL compatibility.



P.O. Box 956, Dept. M, Valley Forge, PA 19482
For Technical Information: (215) 269-0198

Reader Service Number 60

automatic copy units, this would definitely do the trick.

How Fast?

On my system, for instance, diskcopy takes about 60 seconds to copy to an unformatted disk (that's without verification).

FastCopy, on the other hand, did the following: With verify off, it copied the data onto a formatted disk in 12 seconds, and onto an unformatted disk in 38 seconds. With verify on, it took 24 seconds to a formatted disk and 50 seconds to an unformatted disk. And, remember, your system doesn't have to reread the master each time it makes a copy.

And I still haven't mentioned the online help, the fancy menus, all the information the program tells you about the source and destination disks and so forth.

Fantastic package. Jim, you did a great job with FastCopy and I wish you all the best.

FastCopy 3.0 \$69.95
Systems, Software, Support
P.O. Box 751022
Houston, TX 77275-1022
(713) 941-3100

Larry's 8088 Piece

This last PC LSI piece from Larry has been one of the most difficult. The 8088 is a complex chip, no doubt about it, but we both expected it to be well documented. It is, and it isn't.

The way it's used in the PC/XT threw us both a curve. Especially when it came to the LOCK pin. The LOCK pin supposedly goes low (true) when you run a LOCK instruction. A LOCK instruction tells the processor that on the next instruction it gets RAM all to itself (no one else gets access to RAM during that instruction). We couldn't figure how the 8088 could access RAM after it had locked out everyone else.

We tried figuring signals logically, then illogically. Neither worked.

Maybe the processor doesn't access RAM during a LOCK. Maybe it makes up its own data during a LOCK. Maybe LOCK doesn't work and no one realizes it. Maybe the XT doesn't work. Maybe I should go back to writing captions for the culture corner.

P.S. Dean Klein straightened us out. I guess that's what deans are for.

Fixing Our Credibility

I received a note from Darrell Bethune of West Vancouver, British Columbia. It was the first time someone had managed book-length prose on a single renewal form. One of Darrell's points:

"Keep editorial integrity. Zortech and Trilogy plugs in the same issue as their first ads?"

You're right Darrell, that's suspicious.

However, both companies earned their plugs from the editors. And, they earned them long before they decided to advertise. See last issue's "On Your Own" column for my thoughts on advertising when you're getting editorial mention. I think they were smart. And, I reserve the right to say what I think about a product whether or not the company advertises.

For instance, I've received three products which print out information on Ventura style sheets (or chapters). Two are standard commercial products, one is shareware. All have bugs. Many of the bugs are significant.

The best, and only really workable package, is called

HANDS-ON PC REPAIR TRAINING

NAC offers the highest quality maintenance training available for PC technicians. The course and its documentation focus on the IBM personal computer lines, from PC to PS/2; peripheral devices, and compatibles such as Compaq and AT&T.

- Diagnose more quickly and accurately, complete repairs faster, and reduce materials costs.
- Discover better diagnostics, parts sources, and new trouble-shooting tools and test units.
- Toll-free technical help and parts-locator line, technical and vendor update service.
- Used by GTE, ITT, Rockwell, SoCal Edison, Southern Bell, the U.S. Dept. of Commerce, and many others.

Our five-day PC Maintenance Course is offered in Southern California, New York, Washington, Chicago, Dallas, Seattle, and other major cities. Call without obligation to get a course outline and dates for the class locations most convenient to you.

**National
Advancement
Corporation**

Computer Hardware Training Specialists

17985-F Sky Park Circle
Irvine, CA 92714
(714) 250-0834

Reader Service Number 59

Withstyle. It's distributed by Pecan Software Systems and it lets you print out tag information as well as transfer tags from one style sheet to another (hooray). You can even use Ventura to print out (beautifully) the style sheet info.

I had problems using this package to modify a style sheet—the numeric displays for leading, spacing, etc., were weird, but if I entered a number several times it finally took. (I just talked to the author and the problem's fixed in version 1R0.1.)

This is by far the most complete style sheet printer/editor. Definitely worth considering now that they've polished up the rough spots. A limited version of Withstyle should be available soon on the *Micro C* RBBS. It only handles five tags. (Meanwhile, the shareware style sheet display is rough, but then it hasn't been released. I'll keep you posted.)

Withstyle \$79.95 (version 1R0.1)
 Pecan Software Systems, Inc.
 1410 39th St.
 Brooklyn, NY 11218

Two Good Books

If you're planning to get into desktop publishing on a PC, you should pick up *Desktop Publishing With Style* by Daniel Will-Harris. This is a huge look at the whole PC publishing arena from Ventura to hard disk utilities.

Daniel compares some of the graphics packages, Desktop packages, typesetters, laser printers, dot matrix printers. Cheap options with reasonable results, etc.

It's a book that'll get you up to speed on the whole desktop publishing arena and you'll enjoy his style. The writing is friendly and easy to read.

If you already have Ventura Publisher and are looking for hints and kinks, I'd recommend *Ventura Tips and Tricks*. It's the book Carol keeps right next to her full-page monitor and it's the one the rest of us borrow carefully (after hours).

Desktop Publishing With Style
 \$19.95, 448 pages
 ISBN 0-89708-162-5
 And Books
 702 South Michigan
 South Bend, IN 46618

Ventura Tips And Tricks
 \$15.95, 286 pages
 ISBN 0-938151-01-0
 Peachpit Press
 2127 Woolsey St.
 Berkeley, CA 94705
 (415) 843-6614

Help!

I'm looking for information on database programs. If you've had experience with one program or a bunch, I want you. Write, log onto the RBBS, call me, whatever. Send an outline, short synopsis, a disk, a desk, your mother (return postage with mothers, please), anything that tells us what you're into.

I'd like to do a complete look at databases. We're talking database design. Database utilities. dBASE III+ compilers, interpreters, look alike... Menu-driven database creators and report generators. SQL. Everything.

Finally

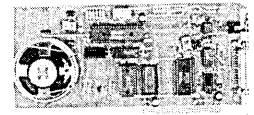
See you at SOG, get those database cards and letters in the mail today, keep your chin up, and your nose to the grindstone—because that's it from greater Bend.

David Thompson



ALL SALES SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. FREE COPY UPON REQUEST.

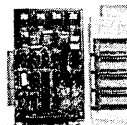
TEXT TO SPEECH BOARD!
PC/XT COMPATIBLE. MAKE YOUR COMPUTER TALK!
 A VERY POWERFUL AND AMAZING SPEECH CARD. USES THE NEW GENERAL INSTRUMENTS SPO256-AL2 SPEECH CHIP AND THE CTS256A-AL2 TEXT TO SPEECH CONVERTER.
 THIS BOARD USES ONE SLOT ON THE MOTHERBOARD AND REQUIRES A COM SERIAL PORT. BOARD MAY ALSO BE USED IN A STAND ALONE ENVIRONMENT WITH ALMOST ANY COMPUTER THAT HAS A RS232 SERIAL PORT. FEATURES ON BOARD AUDIO AMP OR MAY BE USED WITH EXTERNAL AMPS.
 DEMONSTRATION SOFTWARE AND A LIBRARY BUILDING PROGRAM ARE INCLUDED ON A 5 1/4 INCH PC/XT DISKETTE. FULL DOCUMENTATION AND SCHEMATICS ARE ALSO INCLUDED.



NEW! **PRICE CUT!** **\$69.95 ASSEMBLED & TESTED**

NEW! IC TESTER! \$149.00
 SIMILAR TO BELOW EPROM PROGRAMMER. PLUGS IN TO YOUR PC OR XT. TESTS ALMOST ALL 14, 16, AND 20 PIN 74XX SERIES. INCLUDES STANDARD POWER, "S" AND "LS" DEVICES. ALSO TESTS CD4000 SERIES CMOS. SOFTWARE INCLUDED CAN EVEN DETERMINE PART NUMBERS OF MOST UNMARKED AND HOUSE NUMBERED DEVICES WITH SIMPLE MOD. THIS UNIT CAN ALSO TEST 6.4K AND 256K DRAMS! WITH MANUAL AND SOFTWARE: \$149. PERFECT FOR SCHOOLS.

PC/XT EPROM PROGRAMMER \$169



ASK ABOUT OUR NEW PAL PROGRAMMER!

* LATEST DESIGN * PROGRAMS UP TO 4 DEVICES AT ONE TIME * FEATURES EASY TO USE MENU DRIVEN SOFTWARE THAT RUNS UNDER PC OR MS-DOS. * USES AN INTELLIGENT PROGRAMMING ALGORITHM FOR SUPER FAST (8X) EPROM BURNING. * THIS PLUG-IN BOARD ATTACHES TO AN EXTERNAL MINI CHASSIS CONTAINING 4 TEXTOL Z.I.F. SOCKETS. * NO PERSONALITY MODULES REQUIRED * AUTOMATIC VPP SELECTION: 12.5V, 21V, OR 25V. * EPROM DATA CAN ALSO BE LOADED FROM OR SAVED TO A DISKETTE. * PROGRAMMING SOFTWARE SUPPORTS: 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 27256A, 27512, AND 27512A. * ASSEMBLED AND TESTED, BURNED. IN WITH MANUAL. \$169 WITH SOFTWARE.

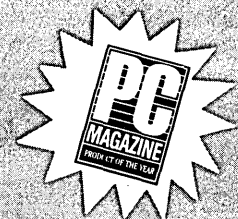
JUST RECEIVED. SAME AS ABOVE PROGRAMMER, BUT PROGRAMS 8 UNITS AT ONE TIME - \$299.

Digital Research Computers
 P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

TERMS: Add \$3.00 postage. We pay balance. Orders under \$15 add 75¢ handling. No C.O.D. We accept Visa and MasterCard. Texas Res. add 6-1/4% Tax. Foreign orders (except Canada) add 20% P & H. Orders over \$50 add 85¢ for insurance.

Reader Service Number 32

ShareWare™

HOMEBASE™

"An optionally memory-resident desktop organizer."

INCLUDES:

- Calculator
- DataBase
- Text Editor
- Communications Program
- Things-To-Do List
- Cut & Paste 4 Items
- Autodialer
- Calendar
- Alarm Clock
- DOS Services
- Expense Tracker

It's no wonder that *HomeBase™* was chosen **Software Product of the Year by PC Magazine!** *HomeBase™* has just about everything for everybody, and it is always available at the press of a key.

If you have a hard-disk system, *Homebase™* takes just 80K of RAM! If you use a computer, you need *HomeBase™!*

\$89.95

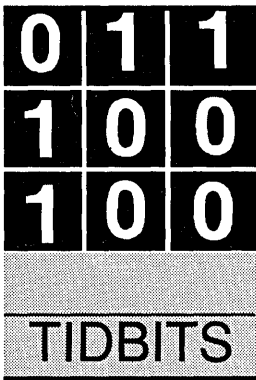
Just give us a call at:
1-800-523-0764
 IN CALIFORNIA 1-800-323-5335 or (408) 559-4545

VISA, MasterCard, accepted or mail your check to:
 File 41719, Box 60000, San Francisco, CA 94160-9719

SHAREWARE DISK AVAILABLE FOR \$10.00

Reader Service Number 87

© Copyright 1988, TeleMarketing Resources, Inc.



Chaos, Butterflies, And The Borland Graphics Interface (BGI)

By Gary Entsminger

1912 Haussler Dr.
Davis, CA 95616

The whole AI field probably appears chaotic. Well, chaos describes lots of things, more things than you might believe.

Most of us believe, perhaps intuitively, that if we know the initial value of a variable (say, X) and the function that describes how it changes (in time), we can predict the future value of the variable at any time.

Consider the equation—

$$X_{Next} = R * X$$

where X is the variable and R describes its rate of change.

If we assume that each NextX is the X of the next equation (in other words, if we feed each XNext back into the equation as its new seed or X), then—

If $R > 1$, XNext will always increase, until at some infinite time, XNext will grow to be infinitely large.

And if $R < 1$, XNext will decrease until at some infinite time, it will approach 0. Either way, the function is dependable.

A new science, called Chaos, which is loosely the study of feedback or iterative equations, has recently challenged this simple notion of predictability, demonstrating convincingly that initial values and a function (or rate equation) aren't always sufficient for prediction.

I'll try to explain, and then I'll show you some code which will let you generate Chaos on your PCs.

Non-Linearity & The Butterfly Effect

Let's go back to the equation, "XNext = R * X." A little thought gives us one reason why it's so predictable—it's linear.

A moment's reflection should convince you that a linear description of behavior (or change) is the simplest of many possible descriptions (just as a straight line is the simplest connection between two points). But unfortunately, it

doesn't describe most interesting phenomena in nature—waves, clouds, turbulence, biological systems, population dynamics, thermodynamics, the weather, oscillation, and so on. Typically, nature requires more complex (often non-linear) equations.

So, let's consider one of, if not the simplest, non-linear equation—

$$X_{Next} = (R * X) * (1 - X)$$

which describes a familiar shape, the parabola. It reaches a peak (determined by R) and then folds down. We can say that R expresses the linearity in the system.

If we start with some initial value of X (or Seed value) and iterate the function, each time putting XNext in for X, we'll discover that XNext reaches some equilibrium, some expected value. That is—we'll find that it reaches an expected value some of the time, but not always.

In fact, not only will it not reach an expected value some of the time, but it will explode into bizarre patterns and shapes, eventually leading to and through Chaos, or totally unexpected values.

The key to how the function behaves is R. At low R values (up to 3), the function is predictably predictable. But at R values between 3 and 4, anything can happen; and perhaps most significant, anything can happen after very subtle or minute changes in R.

For example,

at 3.8284, a basic cycle of 3 first appears;

at 3.8415, it becomes unstable;

at 3.99026 a basic cycle of 5 appears;

at 3.99030, it becomes unstable;

and so on. The subtlety in variation is virtually infinite.

This effect—of small changes in behavior (often generations before) leading to significant

changes—is known as the butterfly effect, and has subtle implications in nature.

If you're interested in pursuing this fascinating subject, I enthusiastically recommend James Gleick's new book, *Chaos*, the first full-length feature study of this new science which promises to affect many different areas of science.

Meanwhile, I'll briefly show you how to simulate Chaos on your PC using the Borland Graphics Interface. I'm using it with Turbo Prolog, but you C and Pascal programmers should have little trouble extrapolating to your favorite language. If you just want to see Chaos without programming it, grab CHAOS.EXE from the Micro C Bulletin Board.

BGI

I've only had a few days to work with BGI, but I've already found it to be easy to use and powerful.

In order to generate Chaos (and to get BGI up and running in general), you need to—

Initialize the graphics system;

Set foreground and background colors;

Set a ViewPort (or window) for graphic display;

And draw.

You can set a device driver (to enable EGA or Hercules compatibility, for example) or use a BGI predicate (or function) to determine which graphics display is online. For a demonstration of the latter, check out the Micro C RBBS or get the Issue #41 disk (\$6 for subscribers, \$8 for non-subscribers and foreign). Borland supplies drivers for EGA, CGA, MCGA, VGA, Hercules, AT&T 400-line, 3270 PC, and IBM 8514 graphics.

In all, the BGI includes 70 or more predicates for drawing dots, lines, rectangles, arcs, circles, 3-d bars, text, and so on in various line widths and fonts. It's by far the most complete graphics toolbox I've used and is a big improvement over the Turbo Prolog and early Turbo Pascal graphics routines. I haven't had a chance yet to compare the Turbo Prolog BGI with Turbo Pascal 4.0

or Turbo C 1.5 BGIs, but I expect they're very similar.

The code on the bulletin board uses a linemenu (from the Prolog Toolbox), but you can use the vertical menu supplied with Turbo Prolog instead. Just substitute menu for linemenu and omit the "tpreds," "tdoms," and "linemenu" include files.

For more information about BGI—

Borland International
4585 Scotts Valley Dr.
Scotts Valley, CA 95066

For more info about Chaos—Gleick, James. *Chaos, Making A New Science*. 1987. Viking.

Hofstadter, Douglas. *Metamagical Themas: "Questing For The Essence Of Mind And Pattern"*. 1985. Basic Books.

May, Robert M., *Simple mathematical models with very complicated dynamics*. Nature. Vol 261. June 10, 1976.

And that's Tidbits.

◆ ◆ ◆

Figure 1 - Turbo Prolog Chaos

```

include "tdoms.pro"
include "tpreds.pro"
include "linemenu.pro"

Database - graphics

Determ driver(Integer,Integer,String)
Determ maxcolors(Integer)
Determ maxX(Integer)
Determ maxY(Integer)
Determ aspectCorr(Real)
Determ graphCoord(Integer,Integer)

params(Real,Real,Real,Real,Real,Real,Real,Real,Real)

PREDICATES

GetDriverName(Integer,String)
Initialize
KeepColor(Integer,Integer,Integer)
ToGraphic
ToText

CLAUSES

Initialize:-
  ToGraphic,      GetMaxColor(MaxColors),
  assert(maxcolors(MaxColors)),
  GetMaxX(MaxX),  assert(maxX(MaxX)),
  GetMaxY(MaxY),  assert(maxY(MaxY)),
  GetAspectRatio(Xasp,Yasp),
  AspectRatio = Xasp/Yasp,
  assert(aspectCorr(AspectRatio)).

ToGraphic:-      /* Detect graphic equipment */
  DetectGraph(G_Driver, G_Model),
  KeepColor(G_Driver,G_Model,G_Mode),
  GetDriverName(G_Driver,G_Name),
  assert(driver(G_Driver,G_Mode,G_Name)),
  InitGraph(G_Driver,G_Mode,_,_,bgi_Path),!.

/* If you want to set a driver w/o detecting it,
use the following 2 lines instead of
the previous 5:
assert(driver(7,7,"HercMono")),
InitGraph(7,0,_,_,bgi_Path),!.      */

ToText:- closegraph().

KeepColor(1,_,0). KeepColor(_,Mode,Mode).

GetDriverName(0,"Detect"). GetDriverName(1,"CGA").
GetDriverName(3,"EGA").
GetDriverName(7,"HercMono").

/* Main */
PREDICATES

Function(Real,Real,Real,Real,Real,Real,Real,Real,Real,Real)
Iterate(Real,Real,Real,Real,Real,Real,Real,Real,Real,Real)
Iterate_Trace(Real,Real,Real,Real,Real,Real,Real,Real,Real,Real)
Main
Main menu
NextProcess(Integer)
Process(Integer)
Testwindow(Integer)

CLAUSES

Iterate(X,L,Count,I,XScale,YScale,Height,Width,
YStart,XStart):-
  Y = X * (1-X),          /* Y = XNext */
  Y1 = Y * L,             /* Row & Col are scaled */
  Row = Height-(((Y1 - YStart)/YScale) * Height),
  Col = (((Count - XStart)/ XScale)) * Width,
  putpixel(Col,Row,7),
  Newcount = Count + I,   XEnd = XScale + XStart,
  Newcount <= XEnd,      /* Have we reached
  user-defined end? No, then iterate. */
  !, Iterate(Y1,L,Newcount,I,XScale,YScale,Height,
Width,YStart,XStart).
Iterate(_____,_____,_____,_____,_____,_____,_____,_____,_____,_____).

```

```

Iterate_Trace(X,L,Count,I,XScale,YScale,Height,
Width,YStart,XStart):-
Y = X * (1-X), Y1 = Y * L,
Row = Height-((Y1 - YStart)/ YScale) * Height),
Col = (((Count - XStart)/ XScale)) * Width,
shiftwindow(3), write("YStart: ",YStart),
write("X : ",Count," "), write("Y : ",Y1," "),
write("Col : ",Col," "),
write("Row : ",Row," "),nl,
putpixel(Col,Row,7),
Newcount = Count + I, XEnd = XScale + XStart,
Newcount <= XEnd, !,
Iterate_Trace(Y1,L,Newcount,I,XScale,YScale,
Height,Width,YStart,XStart).
Iterate_Trace(_____.

/* Draw parabola to scale. */
Function(I,X,Height,Width,XScale,YScale,L,
XStart,YStart):-
Y = L * X * (1 - X),
Row = Height-((Y - YStart)/ YScale) * Height),
Col = ((X - XStart)/ XScale)) * Width,
NextXRel = X + I, XEnd = XScale + XStart,
NextXRel <= XEnd,
putpixel(Col,Row,7),
!,function(I,NextXRel,Height,Width,XScale,
YScale,L,XStart,YStart).
function(_____.

Main:-
makewindow(1,7,0,____,0,0,25,80),
makewindow(2,7,0,____,2,0,12,80),
makewindow(3,7,0,____,22,0,3,80),
Main_menu.
Main_menu:-
linemenu(0,18,0,["Parameters","Function",
"Iterate","Trace","Quit"],C),
Process(C), ToText, Testwindow(81), Main_menu.

Testwindow(N):-
existwindow(N), shiftwindow(N), removewindow.
Testwindow(N).

Process(1):-
linemenu(1,103,0,["Show","Lambda","Step",
"XStart","XEnd","YStart","YEnd"],C),
NextProcess(C), testwindow(81).
Process(1).
Process(2):-
Initialize, ClearDevice,
SetColor(1), % Set current color to white
SetBkColor(0), % Set background to black
maxX(MaxX), maxY(MaxY), R = MaxY - 50,
SetViewport(0,0,MaxX,R,1),
% Open port to upper screen
GetViewSettings(Left,Top,Right,Bottom,____),
Rectangle(Left,Top,Right,Bottom), % Draw Border
Height = Bottom-Top, Width = Right - Left,
params(XStart,____,XScale,YStart,____,YScale,Seed,
Lambda,Inc),
function(Inc,Seed,Height,Width,XScale,YScale,
Lambda,XStart,YStart), readchar(____).
Process(2).
Process(3):-
Initialize,
SetColor(1), % Set current color to white
SetBkColor(0), % Set background to black
maxX(MaxX), maxY(MaxY), R = MaxY - 50,
SetViewport(0,0,MaxX,R,1),
% Open port to upper screen
GetViewSettings(Left,Top,Right,Bottom,____),
Rectangle(Left,Top,Right,Bottom), % Draw Border
Height = Bottom-Top, Width = Right - Left,
params(XStart,XEnd,XScale,YStart,____,YScale,
Seed,Lambda,I),
iterate(Seed,Lambda,Seed,I,XScale,YScale,Height,
Width,YStart,XStart), readchar(____).
Process(3).
Process(4):-
Initialize,
SetColor(1), % Set current color to white
SetBkColor(0), % Set background to black
maxX(MaxX), maxY(MaxY), R = MaxY - 50,
SetViewport(0,0,MaxX,R,1),
% Open port to upper screen

```

```

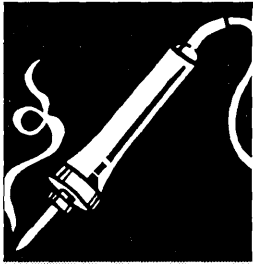
GetViewSettings(Left,Top,Right,Bottom,____),
Rectangle(Left,Top,Right,Bottom), % Draw Border
Height = Bottom-Top, Width = Right - Left,
params(XStart,XEnd,XScale,YStart,____,YScale,
Seed,Lambda,I),
Iterate_Trace(Seed,Lambda,Seed,I,XScale,YScale,
Height,Width,YStart,XStart), readchar(____).
Process(4).
Process(5):- testwindow(81), exit(1).

NextProcess(1):-
shiftwindow(2), params(A,B,C,D,E,F,G,H,I),nl,
write("XStart : ",A),nl,
write("XEnd : ",B),nl,
write("XScale : ",C),nl,
write("YStart : ",D),nl,
write("YEnd : ",E),nl,
write("YScale : ",F),nl,
write("Seed : ",G),nl,
write("Lambda : ",H),nl,
write("Step : ",I),nl.
NextProcess(1).
NextProcess(2):-
shiftwindow(2), write("Lambda : "),
readreal(Lam), Lam <> 0,
params(A,B,C,D,E,F,G,H,I),
retract(params(A,B,C,D,E,F,G,H,I)),
asserta(params(A,B,C,D,E,F,G,Lam,I)).
NextProcess(2):-
shiftwindow(2), write("Invalid Lambda = 0").
NextProcess(3):-
shiftwindow(2), write("Step : "),
readreal(Inc), Inc <> 0,
params(A,B,C,D,E,F,G,H,I),
retract(params(A,B,C,D,E,F,G,H,I)),
asserta(params(A,B,C,D,E,F,G,H,Inc)).
NextProcess(3):-
shiftwindow(2), write("Invalid Step = 0").
NextProcess(4):-
shiftwindow(2), write("XStart : "),
readreal(XS), params(A,B,C,D,E,F,G,H,I),
XScale = B - XS, XScale <> 0,
retract(params(A,B,C,D,E,F,G,H,I)),
asserta(params(XS,B,XScale,D,E,F,G,H,I)).
NextProcess(4):-
shiftwindow(2), write("Invalid X Scale = 0").
NextProcess(5):-
shiftwindow(2),
write("XEnd : "), readreal(XE),
params(A,B,C,D,E,F,G,H,I),
XScale = XE - A, XScale <> 0,
retract(params(A,B,C,D,E,F,G,H,I)),
asserta(params(A,XE,XScale,D,E,F,G,H,I)).
NextProcess(5):-
shiftwindow(2), write("Invalid X Scale = 0").
NextProcess(6):-
shiftwindow(2), write("YStart: "), readreal(YS),
params(A,B,C,D,E,F,G,H,I),
YScale = E - YS, YScale <> 0,
retract(params(A,B,C,D,E,F,G,H,I)),
asserta(params(A,B,C,YS,E,YScale,G,H,I)).
NextProcess(6):-
shiftwindow(2), write("Invalid Y Scale = 0").
NextProcess(7):-
shiftwindow(2),
write("YEnd : "), readreal(YE),
params(A,B,C,D,E,F,G,H,I),
YScale = YE - D, YScale <> 0,
retract(params(A,B,C,D,E,F,G,H,I)),
asserta(params(A,B,C,D,YE,YScale,G,H,I)).
NextProcess(7):-
shiftwindow(2), write("Invalid Y Scale = 0").

GOAL assert(params(0,1,1,0,1,1,0.04,3.0,0.003)),
Main.

END OF LISTING

```



Technical Tips

Logitech/Zenith Compatibility

Laine Stump complained recently (*Micro C* issue #40, p.58) about incompatibility between Logitech's C7 Serial Mouse and the Zenith Z181 laptop. I also own both units.

Calls to Zenith and Logitech identified the problem as the source of power to the Logitech mouse. The mouse receives power from the RS232 port. Zenith has confirmed that their serial driver chip does not provide as much voltage as the chips used by other manufacturers. This conserves battery power.

I suggest that anyone who wishes to use a Logitech mouse with the Zenith laptop add a serial to serial interface which would supply the necessary power to the mouse. I've thought about connecting the required 1488 and 1489 drivers back to back but haven't had the time.

Rod Morimoto
1107 Oregon Ave.
Palo Alto, CA 94303

Fractal Parameters

I received my first issue of *Micro Cornucopia* and fed the "Mandelbrot Set Generator" routine into my Turbo C compiler. After a few trials and errors, it generated a fair sample of the set—produced as a screen dump of CGA graphics using DOS's GRAPHICS.COM.

A colleague of mine, Rick Daley, requested a copy and produced the enclosed sketch of a "Mandelbrot Buddha" (See Figure 1). I found it amusing enough to pass it along.

Since I'm aware of how difficult it is to do technical writing, and to decide what to put in and what to leave out, I'm reluctant to be critical of such a nice and useful article as "Drawing The Mandelbrot And Julia Sets." It would, however, have been extremely helpful if the plot had been accompanied by its range.

And had some interesting "zoom" ranges been included, naive experimenters could have saved themselves a lot of time. A friend suggested several ranges that I haven't tried yet. (See first three lines of Figure 2.) I've been told that a cutoff of 256 iterations works nicely for

the first two and 512 iterations works well for the third.

Gregory K. Landheim
349 Almond St.
Salt Lake City, UT 84103

Figure 1—Formal Mandelbrot Buddha



Editor's note: Gregory's right. Infinitely many choices for Mandelbrot parameters means we also have infinitely many bad choices. Apologies to those of you still (slowly) cranking out blank screens.

Figure 2 shows the parameters Gregory sent, plus a few from The Beauty of Fractals. Remember, any region near the boundary of the Mandelbrot set will produce interesting results. Also, a full 1000 iterations doesn't add much resolution, so stick to 500 or less.

Seagate Step Rates

The Seagate ST4000 series full-height, linear voice coil drives have an unusually narrow step

Eco-C88 C Compiler with Cmore Debugger

Professionals prefer the Eco-C88 C compiler for ease of use and its powerful debugging features. Our "picky flag" gives you nine levels of lint-like error checking and makes debugging easy:

"I'm very impressed with the compiler, editor, and debugger. I've tried quite a few different compilers for the PC and have given up on all of the others in favor of yours... I've gotten to the point where I download C code from a DEC VAX/VMS system just to be able to compile it with the picky flag set at 9. It finds lots of things VMS totally ignores..."

JS, Oak Ridge, TN

The Eco-C88 compiler includes:

- A full-featured C compiler with 4 memory models (up to 1 meg of code and data) plus most ANSI enhancements.
- Without a doubt, the best error checking you can get. We catch bugs the others miss, making you much more productive.
- Cmore is a full-featured source code debugger, not some stripped-down version.
- Robust standard library with over 230 useful (no "fluff") functions, many of which are System V and ANSI compatible. Full source is available for only \$25.00 at time of order.
- CED, a fast, full screen, multiple-window program editor with on-line function help. You can compile, edit, and link from within CED.
- cc and mini-make utilities included that simplifies the most complex compiles.
- Users manual with over 150 program examples (not fragments) to illustrate how to use the library functions.
- Fast compiles producing fast code.

Our Guarantee: Try the Eco-C88 compiler for \$99.95. Use it for 30 days and if you are not completely satisfied, simply return it for a full refund. We are confident that once you've tried Eco-C88, you'll never use anything else. Call or write today!

Orders: **1-800-952-0472**
Info: **1-317-255-6476**



Ecosoft Inc.
6413 N. College Avenue
Indianapolis, IN 46220

ECOSOFT

Reader Service Number 9

rate range. This may require setting the step rate on the controller to a non-default value. Otherwise, the drive may operate unreliably or not at all.

Our experience has been limited to problems with the (recently discontinued) ST4026, together with the ubiquitous Western Digital WD1002A WX1 controller. The ST4026 has the narrowest step rate range of all of this family of drives: 25 - 50 usec. The WX1 defaults to a step rate of 70.5 usec, which is too slow for this drive, and possibly too slow for its siblings as well (See Figure 3).

Seagate says that voice coil drives typically have a narrow step rate window. Seagate's stepper motor drives do in fact have much larger windows, typically 5 - 200 usec (ST225).

The ST4026 OEM Manual says that step rates between 25 and 50 usec and above 3 msec are supported, but that

errors immediately, on every sector.

We discovered that the controller's default step rate (70.5 usec) was outside of the 25 - 50 usec range specified for the ST4026. We changed it to a smaller value (28.5 usec) and reformatted the drive. It has since worked perfectly, for many hours.

Different controllers have different default step rates, though 70.5 seems to be an oft-used value. The OMTI 5510 defaults to 15 usec, according to its manual. The Centan 5027 controller has a default step rate of 18 usec, thinks Mark Miyakawa of NCL America (Centan). There should be a way to change the default step rate for any controller, either through software or via jumpers on the board.

The step rate for the Western Digital WD 1002A WX1 controller is changeable only through the formatting routine in its BIOS ROM. Furthermore,

Figure 2 - Fractal Parameters

Mandelbrot Sets

Pmin	Pmax	Qmin	Qmax
-0.74877	-0.74162	0.1091	0.11625
-0.7454716	-0.7453868	0.1129605	0.1130453
-0.74554433	-0.7454151	0.1129958	0.1130241
-2.25	0.75	-1.5	1.5 (shows whole set)
-0.1992	-0.12954	1.0148	1.06707
-0.95	-0.88333	0.23333	0.3
-0.75104	-0.7408	0.10511	0.11536
-0.74591	-0.74448	0.11196	0.11339

Julia Sets

P	Q	Xmin	Xmax	Ymin	Ymax
0.32	0.043	-2.0	2.0	-1.5	1.5
-0.39054	-0.58679	-1.5	1.5	-1.5	1.5
0.27334	0.00742	-1.3	1.3	-1.3	1.3

Figure 3 - Step Rate Ranges for Seagate ST4000 Series Drives

	ST4026	ST4038	ST4051	ST4053	ST4096
Formatted Capacity (MB)	21.4	31.9	42.5	44.5	80.2
Step Rate Range (usec)	25-50	10-70	10-70	3-70	3-70

"Step pulses issued between 50 usec and 3 msec may be lost." (Who needs 'em?!)

Using the WX1 controller, we low-level and DOS-formatted three ST4026s. In each case the drive appeared to operate normally until we attempted to copy over files. With each drive, 20 or fewer files went over fine, followed by "Seek Error Writing Drive C:".

We were able to read endlessly from the drives without errors but read/write operations produced read

one must choose to "dynamically" format the drive (specify parameters at the keyboard). This is not an option with the early BIOS ROMs for this controller (or with the BIOS on the WX1s shipped with Kaypro PCs). Either the Auto-Config BIOS (62-000043-xxx) or the Super BIOS (62-000094-xxx) must be present in socket U12 for this to be possible.

For dynamic formatting, leave all jumpers at jumper block S1 OFF. Run the formatter from DEBUG by entering: G=C800:5

at DEBUG's "-" prompt.

The formatter will ask for the drive ID (C:/D:), desired interleave (7 recom-

Figure 4 - WX1 Step Rate Codes

0	3.1 msec
1	46.5 usec
2	22.5 usec
3	10.5 usec
4	202.5 usec
5	70.5 usec
6	28.5 usec
7	10.5 usec

mended for XT-class machines), and whether you wish to "dynamically configure" the drive (Yes). You will then be prompted for number of heads and cylinders, etc.

The last number to specify is the "CCB Option Byte," otherwise known as the step rate. The default value is 5, which sets the rate as 70.5 usec. Use 6 instead for the ST4026, which corresponds to 28.5 usec. This is probably a

good value for the other members of the ST4000 series, too, although one source (Konan controller manual) indicates that these drives should be dealt a step rate of a mere 10.5 usec (option 7). This makes sense, given the range for those drives.

The numbers are entered all on one line, separated by spaces. For the ST4026, you should key in:

615 4 616 300 11 6

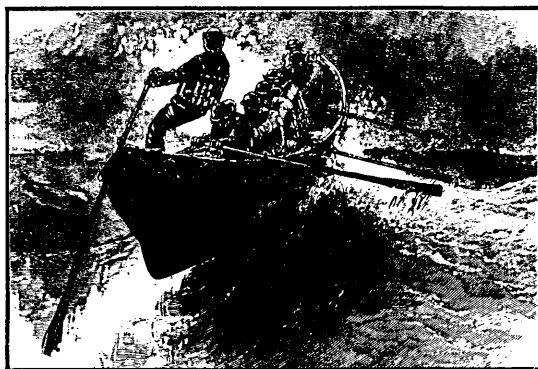
Which is to say: 615 cylinders, 4 heads, no reduced write current (think about it), write precompensation starting at cylinder 300, the DOS-standard error burst length of 11 (I don't know what it means either), and a step rate code of 6 (See Figure 4).

For further formidable details, see Western Digital's WX1 manual, and Seagate's ST-4026 OEM Manual.

Warren Allen
439 Main St.
Bennington, VT 05201

◆ ◆ ◆

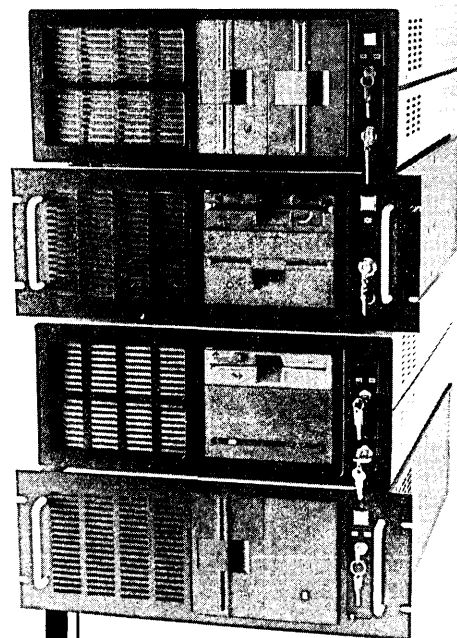
*Tired of the same old white water?
Want a truly exciting, memorable way to spend
Thursday July 14?*



While ordinary folks face death in tiny boats you could be having the time of your life. You could be enjoying a day-long Introduction to Desktop Publishing featuring those desktop ne'er-do-wells from Micro C, Carol Steffy, Sandy Thompson, and David Thompson.

You'll be introduced to scanning, Ventura'ing, PageMak'ing, graphics generation, laser printers, the works. Not only will you understand how desktop works but you'll also have a better idea what you will (and won't) need to get started. Sign up yourself, your spouse, your boss, everyone and join us July 14 in Bend, OR. for this kickoff of SOG VII. Only \$65.00 including lunch and barbeque cookout.

To register, fill out the SOG order form (in this issue) or call
1-800-888-8087.



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 PC/AT Chassis

Rack & Desk Models

Accepts PC, XT, AT Motherboards and Passive Backplanes

Doesn't Look Like IBM

Rugged, Modular Construction

Excellent Air Flow & Cooling

Optional Card Cage Fan

Designed to meet FCC

204 Watt Supply, UL Recognized

145W & 85W also available

Reasonably Priced

Reader Service Number 22

INTEGRAND
RESEARCH CORP.

Call or write for descriptive brochure and prices:
8620 Roosevelt Ave. • Visalia, CA 93291
209/651-1203

TELEX 5106012830 (INTEGRAND UD)
EZLINK 62926572

We accept BankAmericard/VISA and MasterCard

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

SOG VII

July 14 - July 16

THURSDAY RAFTING

- We start SOG VII off by offering two trips down the Deschutes River. You can choose between...
- A 2 1/2 hour raft trip with 3 miles of class I-III rapids. Your price of \$32 per person includes the Old Fashioned Barbeque. Check in at the Pinckney Center registration table before 11 a.m. Thursday for raft scheduling.

OR...

- An all day expedition down the lower Deschutes River Canyon approximately 1 1/2 hours from Bend. This trip includes class I-IV whitewater with 7 major rapids. A deli lunch and the Old Fashioned Barbeque are included in the \$70 per person price. Busses leave Thursday morning at 7:30 a.m. sharp. Check in at the Pinckney Center registration table by 7:00 a.m. for a head count and bus loading.

INTRO TO DESKTOP SEMINAR...

- White water rafting not your thing? Then try the day-long introduction to desktop publishing featuring those desktop ne'er-do-wells from Micro C, Carol Steffy, Sandy Thompson, and David Thompson.
- You'll be introduced to scanning, Ventura'ing, PageMaking, computer graphics generation...the works. You'll break into groups for design competitions and problem solving. Plus you'll get fed! Included in the \$65 price are lunch and the barbeque cookout. This seminar must be prepaid and will run from 9 a.m. to 5 p.m., Thursday, July 14. Check in at the Pinckney Center registration desk Thursday morning. For more information, contact Carol Steffy at 503-382-8048.

THE OLD FASHIONED BARBEQUE

- Will be held at Shevlin Park Thursday evening. Since Saga Foods did such a fine job of making sure we had plenty to eat last year, they will again be cooking

It's SOG time again!

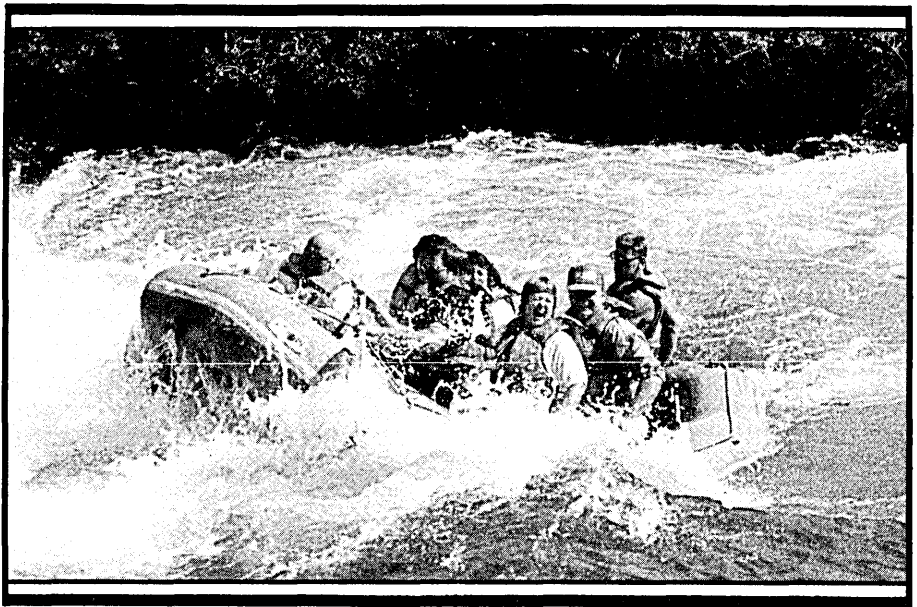
The seventh annual SEMI-OFFICIAL GET TOGETHER (SOG) is coming to Bend, Oregon, July 14 through July 16, 1988. The conference will be held on Friday and Saturday at Central Oregon Community College, but we'll get things started Thursday with river rafting and a barbeque. Let us know your plans so we can throw another burger on the barbie.

REGISTRATION Fill out and return the attached registration form indicating which activities you wish to attend. Enclose your full payment with the form. All technical sessions on Friday and Saturday are free, but to help defer the cost of the conference, we are asking for donations in any amount

Check in at the Pinckney Center Thursday afternoon or Friday morning to receive your SOG packet. (The SOG packet will include your tickets to the barbeque, Saturday night banquet tickets, and a speaker schedule.) If you're going rafting, check in at the Pinckney Center at the times mentioned under rafting.

People with display tables can set up their equipment in the Pinckney Center Thursday.

The dorms at Central Oregon Community College will be closed for remodeling this year. We'll send out lodging pamphlets with the SOG information packets. Travel arrangements can be made through Cascades Travel, 503-382-3772.



up the food. Prices for the barbeque are \$10 for adults and \$6 for children under 12. Thursday Night Barbeque is included in the price of all raft trips.

FRIDAY & SATURDAY

- These two days will be filled with lectures, discussions, workshops, and forums. We'll be discussing new C Compilers, C++, a new AI language, and more. For you hardware people PC Tech will be there in force to talk about graphics and extended memory boards. We'll continue with parallel processing and the transputer and delve into desktop publishing, fractals and packet radio.

SECOND ANNUAL BUILD YOUR OWN SYSTEM

- Friday night, MicroSphere will be hosting another "Build Your Own Computer" night at SOG. This event will be held at MicroSphere starting at 7 p.m.
- Several participants in last year's class are now teaching XT construction classes at

community colleges in their hometowns.

- They will be offering 8088 based XT, 80286 based AT, and 80386 kits at special SOG prices. As with last year, shipping the system home can be arranged. Contact Don Thompson or Cindy Johnson of MicroSphere at 503-388-1194 for more information and a registration form.

The SATURDAY NIGHT BANQUET is an all-you-can-eat affair at the college. Price is \$12 for adults or \$7 for children under 12

CHILD CARE There will be a supervised playground for the kids Friday and Saturday from 9-4.

DEADLINES All activities must be paid for by July 1, 1988.

SOG REGISTRATION, 1988

NUMBER	EVENTS	PRICE	TOTAL
	Thursday all day Raft trip & Old Fashioned Barbeque	\$70.00	
	Thursday 2 1/2 hour Raft trip & Old Fashioned Barbeque	\$32.00	
_____ Adult _____ Child	Thursday night Old Fashioned Barbeque only (Included in Raft trips)	Adult \$10.00 Child \$6.00	
	Thursday Desktop Seminar (Includes lunch and Barbeque)	\$65.00	
_____ Adult _____ Child	Saturday night SOG Banquet	Adult \$12.00 Child \$7.00	
	Semi-Official SOG T-shirt Adult sizes Small Medium Large Ex-large	\$7.00	
	Semi-Official SOG T shirt Youth sizes Small Medium Large	\$5.50	
	Donations		
<input type="checkbox"/> Check enclosed Register by July 1st Number attending in your group _____		TOTAL ENCLOSED	

Exp. _____ Signature _____

Card No. _____

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

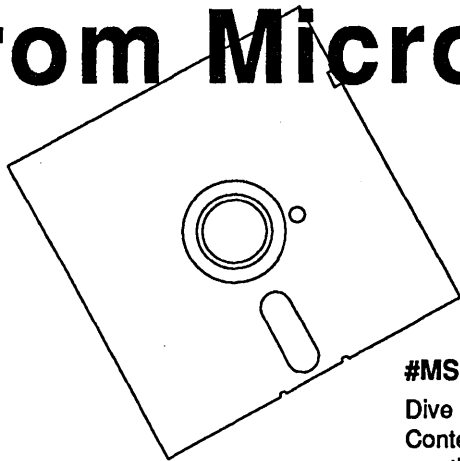
MICRO CORNUCOPIA • P.O. BOX 223 • BEND, OR 97709 • 503-382-5060

Come join us and learn more about...

- C compilers and C tools
- Trilogy: A new AI language
- Advanced graphic processors
- Super fast ray tracing on the 34010 graphics processor
- Fractals
- Diagnostic software available for the XT and AT and how to use it
- Disk diagnostics
- Pseudo concurrency using probabilistic technique
- Numerical applications
- Desktop publishing
- All about SCSI
- Generic microprocessors
- Packet radio
- and much, much more.....



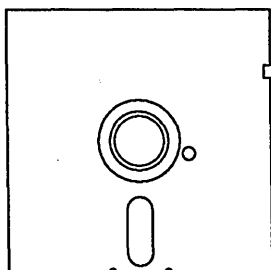
New Users Disks From Micro Cornucopia



These new Micro C Users Disks contain the famous Buttonware collection. This is shareware and we encourage you to register if you find the software useful. We also encourage you to share copies of this software with other PC owners.

\$6.00

(Each For U.S. Subscribers)



#MS45 PC-File+

Dive right into this database manager. Context sensitive help screens will guide you through the entire process.

PC-File+ includes macro capability, password access to foreign databases, and over 64,000 records per database.

Note: PC-File+ has on-screen help. You can get more complete documentation by registering with Buttonware.

#MS46 PC-Type+

Another great Buttonware product. Macros, mail merge, and multiple file editing capabilities make this text editor compete favorably with commercial editors.

Documentation exists in the form of extensive help files along with more brief, context sensitive help screens.

#MS47 PC-Style and Dictionary

Buttonware's PC-Style analyzes your writing for readability, personal tone, and action. Find out if your sentences and words are too short or too long. PC-Style also reports the "grade-level" of your writing.

PC-Type+ uses the dictionary on this disk for its spell-check function.

#MS48 PC-Dial and PC-Graph

Next in the line-up of Buttonware products comes PC-Dial. This communications program supports scripts and macros, has a built-in mini-editor, and talks up to 9600 baud.

Any database or report file created with PC-File can be graphed using PC-Graph. This program requires the Color Graphics Adapter.

#MS49 PC-CALC and Extended DOS

This 64 column X 256 row spreadsheet can import its data from any number of PC-File databases. Or you can treat it as a normal spreadsheet and enter data by hand.

The program is complete but this "evaluation copy's" documentation contains only portions of the manual.

Extended DOS (XD) extends the standard DOS commands. At any time you can view a menu of XD commands which include—killing entire subdirectories, string search through multiple files, file attribute and time/date editing, and more.

#MS50 PC-Tickle and Button Games

Organize your life! Schedule events, keep track of tasks, balance your checkbook, even follow your caloric input and output. Includes its own mini-editor.

And, lest things should get too serious, we finish the Buttonware Collection with two adventure games. Become the main character in a South American Trek or a Castaway adventure.

To Order:

**Call
(503) 382-5060 or
1-800-888-8087**

**Mailing Address:
Micro Cornucopia
P.O. Box 223
Bend, Oregon 97709**

**Prices: (each, postpaid)
\$6.00 for U.S. Subscribers to Micro C
\$8.00 for non-subscribers & foreign**

ORDERFORM

POSTAGE-PAID SELF-MAILER

Tear out, fold, and staple both ends if check is enclosed.

YES, I WANT TO SUBSCRIBE!

NEW

RENEWAL

	U.S.	CAN./MEX.	FOREIGN
1 yr. 6 issues	<input type="checkbox"/> \$18	<input type="checkbox"/> \$26	<input type="checkbox"/> \$36
2 yrs. 12 issues	<input type="checkbox"/> \$34	<input type="checkbox"/> \$50	<input type="checkbox"/> \$68
3 yrs. 18 issues	<input type="checkbox"/> \$48	<input type="checkbox"/> \$72	<input type="checkbox"/> \$99

DISKS MS DOS 5 1/4" MS DOS 3 1/2"
 Other
 Specify Disk # and size _____

DISK TOTAL

OTHER PRODUCTS
 Back Issues, T-shirts... specify size _____

OTHER TOTALS

CHECK ENCLOSED

U.S. funds drawn on a U.S. bank, please

VISA

MASTERCARD

 /

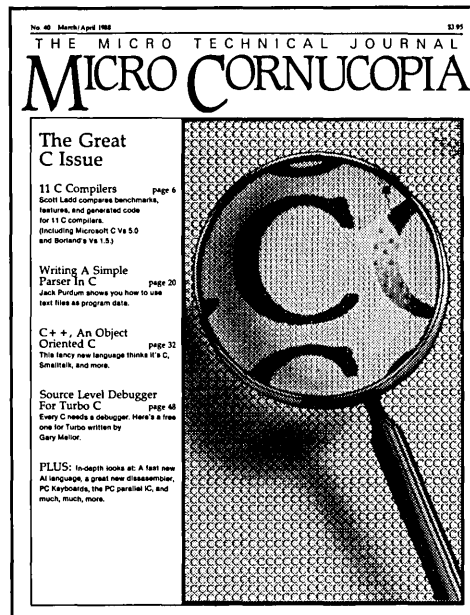
Are you a current Micro C subscriber? Yes No

NAME

COMPANY

ADDRESS

CITY STATE ZIP



Save
24% Off
 the
 newstand
 price

GRAND TOTAL

To Place Your Order Immediately

CALL: 1-800-888-8087

9-5, M-F, Pacific Time

MICRO CORNUCOPIA

READER SERVICE CARD MAY/JUNE 1988 ISSUE NO. 41

Write in the reader service numbers of any advertisers from whom you would like to receive free information.

NAME _____

COMPANY _____

ADDRESS _____

CITY _____

STATE _____ ZIP _____

FOLD HERE



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

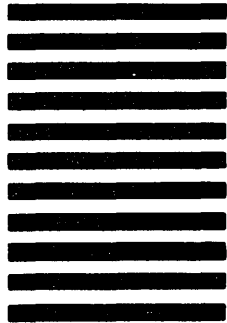
BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 19 BEND, OR

POSTAGE WILL BE PAID BY ADDRESSEE

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

P.O. Box 223
Bend, OR 97709-0223



FOLD HERE

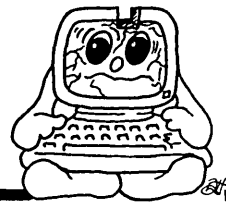
**NEXT
ISSUE**

Diagnostic Issue

- Symptoms and Solutions, Curing Your XT or AT
- Theory of Operation for the XT
- Drive Faults and Fixes
- Plus
- 3-D Graphics Part II
- More on Marketing Your Own Software
- Laine Takes on OS-2

STAPLE TO CLOSE

Fill in your back issues of Micro C today



ISSUE #14 (10/83)
BBII Installation
The Perfect Terminal
Interface to Electronic Typewriter
BBI Video Size
Video Jitter Fix
Slicer Column Begins
Kaypro Color Graphics Review
48 pages

ISSUE #15 (12/83)
Screen Dump Listing
Fixing Serial Ports
Playing Adventure
SBASIC Column Begins
Upgrading Kaypro II to 4
Upgrading Kaypro 4 to 8
48 pages

ISSUE #16 (2/84)
Xerox 820 Column Restarts
BBI Double Density
BBI 5 1/8" Interface Fix
Kaypro ZCPR Patch
Adding Joystick To Color Graphic
Recovering Text From Memory
52 pages

ISSUE #17 (4/84)
Voice Synthesizer
820 RAM Disk
Kaypro Morse Code Interface
68000-Based System Review
Inside CP/M 86
56 pages

ISSUE #18 (6/84)
Kaypro EPROM Programmer
I/O Byte: A Primer
Kaypro Joystick
Serial To Parallel Interface
Business COBOL
60 pages

ISSUE #19 (8/84)
Adding Winchester To BBII
6 MHz On the BBI
Bulletin Boards
Track Buffering On Slicer
4 MHz For The 820-1
64 pages

ISSUE #20 (10/84)
HSC 68000 Co-processor
DynaDisk For The BBII
Serial Printer On BBI Sans S10
Cheap & Dirty Talker For Kaypro
Extended 8" Single Density
72 pages

ISSUE #21 (12/84)
Analog To Digital Interface
Installing Turbo Pascal
Low Intensity BBI Video
Turbo Pascal, The Early Days
80 pages

ISSUE #22 (2/85)
Xerox 820-II To A Kaypro-8
Sound Generator For The STD Bus
Reviews Of 256K RAM Expansion
In The Public Domain Begins
88 pages

ISSUE #23 (4/85)
Automatic Disk Relogging
Interrupt Drive Serial Printer
Low Cost EPROM Eraser
Smart Video Controller
Review: MicroSphere RAM Disk
Future Tense Begins
86 pages

ISSUE #24 (6/85)
C'ing Into Turbo Pascal
8" Drives On the Kaypro
48 Lines On a BBI
68000 Versus 80x86
Soldering: The First Steps
88 pages

ISSUE #25 (8/85)
Why I Wrote A Debugger
The 32-Bit Super Chips
Programming The 32032
Modula II
RS-232C: The Interface
104 pages

ISSUE #26 (10/85)
Inside ZCPR3
Two Megabytes On DSI-32
SOG IV
The Future Of Computing
MS-DOS In The Public Domain
Graphics In Turbo Pascal
104 pages

ISSUE #27 (12/85)
SOLD OUT

ISSUE #28 (2/86)
Pascal Runoff Winners
Rescuing Lost Text From Memory
Introduction To Modula-2
First Look At Amiga
Inside The PC
104 pages

ISSUE #29 (4/86)
Speeding Up Your XT
Importing Systems From Taiwan
Prototyping In C
C Interpreters Reviewed
Benchmarking The PCs
104 pages

ISSUE #30 (6/86)
PROLOG On The PC
Expert Systems
Logic Programming
Building Your Own Logic Analyzer
256 K RAM For Your 83 Kaypro
PC-DOS For Non-Clones
104 pages

ISSUE #31 (8/86)
RAM Resident PC Speedup
Practical Programming In Modula-2
Unblinking The PC's Blinkin' Cursor
Game Theory In PROLOG and C
104 pages

ISSUE #32 (10/86)
Public Domain 32000:
Hardware and Software
Writing A Printer Driver For MS-DOS
Recover A Directory By
Reading & Writing Disk Sectors
96 pages

ISSUE #33 (12/86)
Controlling Stepper Motors
From Your PC
Introduction To Fractals
The Secrets Of MS-DOS, From
Boots To Devide Drivers
Poking About In The System
With Turbo Pascal
96 pages

ISSUE #34 (2/87)
Designing With The 80386
Build A Simple Oscilloscope
A Cheap 68000 Operating System
A concurrent Operating System
Recovering Directories And FATs
96 pages
SOLD OUT

ISSUE #35 (4/87)
SOLD OUT

ISSUE #36 (6/87)
Build A Midi Interface For Your PC
Designing A Database, Part 2
Interrupts On The PC
Hacker's View Of MS-DOS Vs 3.X
Digital To Analog Conversion, A
Designer's View
96 pages

ISSUE #37 (9/87)
Desktop Publishing On A PC
Build Your Own Hi-Res Graphics
Scanner For \$6.00, Part 1
Designing A Database, Part 3
Controlling AC Power From Your PC
Expanded Memory On The PC/XT/AT
Uninterruptible Power Supply For
RAMdisks
96 pages

ISSUE #38 (11/87)
Parallel Processing
Laser Printers, Typesetters
And Page Definition Languages
Magic in the Real World
Build a Graphics Scanner
for \$6.00, Part 2
Writing a resident program
extractor in C.
96 pages

ISSUE #39 (1/88)
PC Graphics
Drawing the Mandelgrot and
Julia Sets
Desktop Graphics
Designing a PC Workstation Board
Around The TMS 3410
96 pages

ISSUE #40 (3/88)
The Great C Issue
11 C Compilers
Writing a Simple Parser in C
C++, An Object Oriented C
Source Level Debugger for Turbo C
96 pages

BACK ISSUES OF MICRO C

U.S., Canada & Mexico

Issues #1-34\$3.00 each ppd.

Issues #35 through current issue\$3.95 each ppd.

Foreign (air mail)

Issues #1 through current issue\$7.00 each ppd.

Advertisers Index

Issue 41

Reader Service	Page Number				
		07 CompuView	.47	12 Logitech Inc	29
		90 Dair Computer Systems	.20	** Lori and Nick's Enterprises	52
72 Acquired Intelligence	.12	08 Datadesk Int'l	.01	17 Manx Software Systems	.77
44 Amtech Computers	.79	32 Digital Research Computers	.81	42 McTek Systems	.65
38 Analogic Company	.50	33 E21 Computer	.68	35 Merlin Publishers	.64
48 ASCII	.35	09 Ecosoft Inc	.86	** Micro Cornucopia	87,88,89
04 Austin Codeworks	.53	10 Emerald Microware	.27	** Micro Methods	.61
05 Blaise Computing	.05	93 Erac Company	.69	36 Microcomputer Systems	.79
01 Borland Int'l	Back Cover	** Gimpel Software	.15	37 Microprocessors Unlmted	.64
87 Brown Bag Software	.61,71,81	11 Halted Specialties	.21	24 microSOLUTIONS	.45
53 C Store	.52	26 Harger I.N.T	.44	02 Microsphere	.02
15 Cascade Electronics	.34	22 Integrand	.87	59 National Advancement	.80
31 CC Software	.60			03 PC Tech	Inside Front Cover
71 Complete Logic Systems	.25			20 Peacock Systems	.25,60
				68 Programmer's Paradise	.07
				** Project Data Systems	.28
				60 Scientific Software	.80
				27 Serengeti Software	.46
				** Sharp's Inc	.71
				19 Slicer Control/Computer	.78
				40 Star-K Software Systems	.14
				62 V Communications	.18
				14 WindowDOS Associates	.39
				39 Xenosoft	.68
				70 Zortech, Inc.	Inside Back Cover
				** Contact Advertiser Directly.	

MICRO ADS

A Micro Ad is an 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.

WINDOWS • SPRITES • MULTIPLE SCREENS

ARE EASY WITH
OMNIVID - PC™

for the **IBM - PC/XT and compatibles.**
BLAZING FAST

Compatible with any language.
Offers support for multitasking.
CGA • Monochrome • Hercules Compatible.
Available March 1988

 **69.95** 

(606) 325-3736

 **FLEXISOFT** 3987 Valley View Drive Ashland, KY 41101

Reader Service Number 25

QUALITY SPEECH I/O PRODUCTS

Digitized speech, synthetic text to speech, voice recognition, music composing with voice, and more are available in hardware/software systems from COVOX. IBM PC/XT/AT and compatibles, Apple, Commodore, and Atari machines supported. Easy programming for fun and education. Or use to design your own products for this rapidly expanding technology. Prices starting at only \$39.95. Call or write the manufacturer direct for FREE product information package.

 **COVOX INC.**
675-D Conger Street
Eugene, OR 97402
(503) 342-1271

Reader Service Number 67

Periscope
*Debugging Systems for IBM
Personal Computers*

FREE INFORMATION
800/722-7006

The **PERISCOPE**  1197 PEACHTREE ST.
PLAZA LEVEL
ATLANTA, GA 30361
Company, Inc. 404/875-8080

Reader Service Number 94

SourceTools

FULL SOURCE CODE MANAGEMENT

- Document revision history
- Audit trails
- Branching and key word insertion
- Utilities

POWERFUL MAKE FACILITY

Available for **VAX, PDP-11 & MS-DOS.**
To order or for more information, call 1-800-874-8501.

OREGON SOFTWARE

6915 SW Macadam Ave., Portland, OR 97219.

Reader Service Number 85

FUTURE86, THE LANGUAGE


Variable level, extensible, rommable language gives complete control. Including 80186/88. Simple syntax. Easy programming. Compact code. Easily optimized for speed or size. Interactive debugging facilities. Library source available. Investigative FUTURE86... world classes! Used in thousands of applications. Prices start at \$349.

Development Associates
1520 S. Lyon, Santa Ana, CA 92705
(714) 835-9512

Cross Assemblers
Universal Linker Librarian

hosted on: **PC/MS DOS, micro VAX,
VAX VMS, VAX UNIX/ULTRIX**

Targeting over 30 microprocessors

 **Enertec inc.**
(215) 362-0966 19 Jenkins Ave
Lansdale, PA. 19446

Reader Service Number 82

PAUL MACE SOFTWARE, INC.

MACE UTILITIES
Data recovery and disk optimization tools. Eliminates hard disk risks and optimizes hard disk performance!
\$99.00

HTEST/HFORMAT
Advanced hard disk diagnostics. Delivers what the manufacturer promised!
\$89.95

400 Williamson Way, Ashland, OR 97520.
(800) 523-0258

Reader Service Number 65

TurboGeometry Library
(Source & Manual)

Turbo Pascal 4.0/C/Mac, & Microsoft C. Over 150 2 & 3 dimensional routines including: Intersections, Transformations, Equations, Hidden Lines, Perspective, Curves, Areas, Volumes, Clipping, Planes, Matrices, Vectors, Distance, Poly Decomp. IBM PC & Comp., Mac.
\$99.95 + S&H. Visa/MC/AE
DISK SOFTWARE, INC.
2116 E. Arapaho, No. 487
Richardson, TX 75081 214/423-7288

Reader Service Number 80

PC C SHELL

A faithful implementation of the Berkeley Unix C Shell for MS-DOS.

- Full History Syntax
- Looping Statements (while, foreach)
- Shell Variables and Arrays
- Cursor Key Command Line Editor
- Runs as default command processor
- WORKS ON ANY MS-DOS COMPUTER!!**

\$50.00
Omega MicroConsulting
722 Rundell Street • Iowa City, IA 52240
(319) 338-6053

Reader Service Number 77

Instruction Manuals
for your product

Written in an informal style by techies for real users.
Free quote.

Rare Earth Services, Inc.
3115 Willow Road NW
Roanoke, VA 24017
703-343-4565

Reader Service Number 73

DEVELOPERS, NOW AVAILABLE

Sof-Tel Inc., in their goal to provide innovative Software Solutions, now has ready for you two new IBM PC software packages. Do you need Modem dialing and interrupt Comm port support? Get AUTO-DIAL! User-definable script files allow use with many modems. Send \$49.95 plus \$3 P&H. Also available. STAGE2-PC, the PC version of the mainframe macro processor STAGE2. Disk plus printed manual only \$49.95 plus \$3 P&H. (Florida residents add 6% sales tax.)

Soft-Tel Inc.
P.O. Box 5116, Lighthouse Pt. FL 33074
(305) 942-6671

Reader Service Number 92

ATRON HARDWARE DEBUGGER

- Hardware breakpoints, source level debugging.
- Includes performance analysis software, 4K hardware trace buffer, external break switch.
- "AT Source Probe" has 1mb onboard memory invisible to DOS. Currently retails for \$2795, will sell for **\$1,450.00 OBO**
Call Michael Crandell 805/969-6851

CASINO CARD GAME PAC \$29
POPUP VT100 EMULATOR \$49
POPUP TIME & BILLING \$59
DES SECURE ENCRYPTION \$59
BBS & REMOT ACCESS \$69
QUICK DATABASE & MAIL \$99
STOCK TRADING SYSTEM! Demo \$49
 IBM PC Compatibles **GUARANTEED SATISFACTION** or
 your Money Back! Don't Delay
 Get Our Free Catalogue Today

Hawkeye Grafix Inc. Box 1400, OLDSMAR
FLORIDA 33557
Dial 813-855-5846

CROSS ASSEMBLERS
PseudoCode releases version 2 of its cross
 assemblers. Assemblers for the 8048, 8051,
 8096, 8085, Z80, 6502, 1802, 6800, 6805, 6809,
 68000, and 32000 microprocessor families are
 available. Macros, Conditional Assembly,
 Include Files plus extensive expression hand-
 ling. Virtually no limit to program size. For IBM
 PC's and true compatibles with MS-DOS 2.0 or
 greater, and 256K memory. Complete with
 printed manual for \$35.00. Each additional is
 \$20.00. (Michigan residents add 4% tax).
 Visa/MC. Order from distributor:
Micro Kit
 6910 Patterson, Caledonia, MI 49316
 616/791-9333

Reader Service Number 75

68000 SOFTWARE
 • K-OS ONE operating system
 uses MS-DOS disks
 with source code \$50
 • K-OS ONE manual \$10
 • HT68K SBC w/K-OS ONE . \$395
 • Screen Editor Toolkit \$50
 • HT-FORTH \$100
 • BASIC \$149
 Free newsletter & spec sheets

HAWTHORNE TECHNOLOGY
 1411 S.E. 31st Ave., Portland, OR 97214
 (503) 232-7332

Reader Service Number 34

Before you code:
The Idea Generator
 "Quickens and improves
 problem solving." - InfoWorld
 Usually \$195. For you, \$145.
 Experience in Software, Inc.
 2039 Shattuck Ave.
 Berkeley, CA 94704 415-644-0694

Reader Service Number 63

8051 SIMULATOR
 Superb full function debugger
 simulator supports all 8051 modes
 of interrupt, just like the real thing!
 Full disassembler and many unique
 features, ONLY \$150.00
CROSS ASSEMBLERS
 For Z80, 8080/8085, 8048, 8051 and
 8096 still \$75.00 each!
LEAR COM COMPANY
 2440 KIPLING ST., SUITE 201
 LAKEWOOD, CO 80215 303/232-2226

Reader Service Number 84

SCREEN MANAGER
 MENU, WINDOW and DATA ENTRY Support
 for the Professional Programmer! Interfaces
 to most languages, BASIC, C, FORTRAN,
 COBOL, PASCAL, ASSEMBLER. 100 Page
 Manual. 30 day money back guarantee.
 No Royalties.
 from The West Chester Group
\$79 P.O. Box 1304
 VISA/MC West Chester, PA 19380
 (215)644-4206

CALL FOR FREE DEMO
 CIRCLE 279 ON READER SERVICE CARD

Reader Service Number 78


**Want to Throw Out your U.P.S.
 Log Book? Now You Can!**
 Here's what EASY-SHIP can do for you:

- ✓ Automatic U.P.S. Shipping to all of U.S & Canada.
- ✓ Fast, Easy Multiple-Shipments with All Options.
- ✓ U.P.S. Approved Shipping Labels & C.O.D. tags.
- ✓ Access to your ASCII Customer Data File.
- ✓ U.P.S. Approved Reports and Manifest Summary.
- ✓ Approved Nationally by United Parcel Service.
- ✓ NO MORE MANUAL LOGGING! And more!

For all IBM PC, AT, OS/2 Systems. Only \$365 + \$3 S/H.
 Stat Supply Company
 20214 Brondesbury, Katy, TX 77450
 (800) 666-4567 or (713) 492-1931

GRAPHICS TOOLKIT
 COMPATIBLE WITH PC/XT/AT AND NEW PS/2
 LIMITED INTRODUCTORY
\$59.95 OFFER (ADD \$3 S/H)

- SUPPORTS NEW VGA GRAPHICS MODES
- 50+ FUNCTIONS
- DATA COMPRESSION ALGORITHM
- ALL SOURCE CODE INCLUDED (NO ROYALTIES)
- ROUTINES WRITTEN IN MICROSOFT AND ASSEMBLER
- BUFFER & DISK SAVE/RESTORE
- PROGRAMMER SUPPORT PROVIDED

 **DEVTRONICS, INC.**
 1571 MAIN STREET
 ATLANTIC BEACH, FLA. 32235
 ORDERS ONLY: 1-800-332-4230 AMEX/COD
 TECHNICAL INQUIRIES: (904) 241-3281

Reader Service Number 86

**35 mm SLIDES FROM
 YOUR IBM/PC**

Computer Slide Express®
 Finally, a one day service
 to convert your graphic
 files into 4000 line color
 slides from your disk for
\$9 or less!

24 HR SERVICE
 Since 1971

Visual Horizons®
 180 Metro Park
 Rochester, N.Y. 14623
 (716) 424-5300

Reader Service Number 79

CBASIC, CB80, CB86 USERS
 Convert your Cbasic and Cbasic compiler programs into C
 with MB86! MB86 supports Access Manager, Display
 Manager, and allows your new C programs to use existing
 CB86 data files. MB86 breaks CB86 file size and record
 number limits and enables network support. MB86 en-
 hances the Cbasic language with new features as well as fixing
 many old CB86 problems. Versions of MB86 are available
 for Microsoft C, Turbo C, Aztec C, and Microport Unix. Call
 or write for more information. We also carry a complete line
 of enhancements for CB86 and Pascal MT+.
 Minnow Bear Computers
 P.O. Box 2233 Station A
 Champaign, IL 61820-8233
 (217) 344-1113

Reader Service Number 83

RAM DISK
S-100
2 Meg, Port I/O
New, Warranted
\$725


S. Lugert
 439 Peck Slip or call:
 NY, NY 10272 718-622-0654

Reader Service Number 52

**16 Megabytes EMS and/or
 Extended Memory**

- Works on 8 or 16 bit bus
- 16 bit transfer on AT bus
- Single board design
- Includes RAM disk and
 extensive diagnostics
- Quantity/OEM discounts

XT and AT
 Compatible

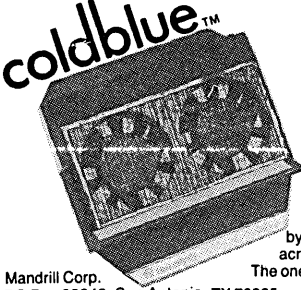
Designed,
 Manufactured,
 Sold and Serviced by 

904 North 6th St. Lake City, MN 55041 (612) 345-4555

Reader Service Number 54

OPT-TECH SORT/MERGE
 Extremely fast Sort/Merge/Select utility.
 Run as an MS-DOS command or CALL
 as a subroutine. Supports most lan-
 guages and filetypes including Btrieve
 and dBase. Unlimited filesizes, multiple
 keys and much more! MS-DOS \$149.
 XENIX \$249.
(702) 588-3737
Opt-Tech Data Processing
 P.O. Box 678 - Zephyr Cove, NV 89448

Reader Service Number 64


coldblue™

A Ventilation
 System
 designed
 to prolong the
 life of IBM®
 PC PC/XT.

Coldblue fits
 in the IBM
 enclosure
 reducing
 operating
 temperatures
 as much as 27°F
 by increasing airflow
 across the card area.
 The one that really works!
\$185.

Mandrill Corp.
 PO Box 33848, San Antonio, TX 78265
 800-531-5314 Dealers inquiries welcome.

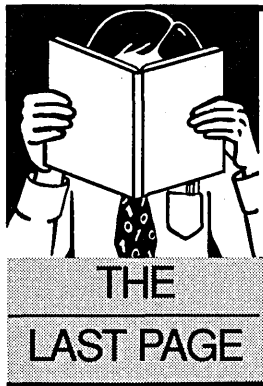
Reader Service Number 91

TurboCAD™
 The New Generation CAD LIST PRICE \$395

**"EASILY OUTCLASSES
 OTHER SIMILARLY
 PRICED CAD SOFTWARE"**
ONLINE

NOW ONLY \$99

Milan Systems America Inc.
 8351 Roswell Road (404) 642-4131
 Dunwoody, GA 30350



Machinery Of The Mind

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

In purest form, "Artificial Intelligence" researchers attempt to recreate human intelligence in machines (or computers). This form seems to have gotten lost in the commercialization of AI. At our level (user, hobbyist), AI seems to mean "fancy programming," which sells better.

Fancy programming has little more to do with "better programming" than it has to do with "Artificial Intelligence." Better programming obviously comes from better ideas and better ways of implementing them.

According to George Johnson, in his well-researched history of AI, *"Machinery Of The Mind"* (available from Microsoft Press)—

Artificial Intelligence is based on the assumption that the mind can be described as some kind of formal system manipulating symbols that stand for things in the world. Thus it doesn't matter what the brain is made of, or what it uses for tokens in the great game of thinking. Using an equivalent set of tokens and rules, we can do thinking with a digital computer, just as we can play chess using cups, salt and pepper shakers, knives, forks, and spoons. Using the right software, one system (the mind) can be mapped onto the other (the computer).

In this form, AI leads to interesting ideas about us, our minds, how we think.

Again, from Johnson—

Ultimately, when all the details and rhetoric are stripped away, most debates on whether or not AI is possible come down to a standoff between the reductionists and the holists—those who believe that mind can be explained as the sum of its parts, and those who believe that in any such explanation something will always elude analysis.

Intuition eludes analysis. It can't be explained at low levels, in the firing of neurons and the like, or at high levels, in rules and facts.

Douglas Hofstadter, believing that it's

neither necessary to model neurons, nor possible to capture the top level of the mind, suggests a middle level where rules emerge.

On this level, where we solve anagrams and the like, we aren't intentionally systematic.

According to Hofstadter, "When you first read a Jumble in a newspaper, you play around, rearranging, regrouping, reshuffling, in complex ways that you have no control over. In fact, it feels as if you throw the letters up into the air separately, and when they come down, they have somehow magically 'glommed' together in some English-like word! It's a marvelous feeling—and it's anything but cognitive, anything but conscious."

As a test of Micro C's intelligence, I sent Larry Fogg Raphael Robinson's puzzle. He likes puzzles, and I had high hopes. Just fill in the blanks, I said—

In this sentence, the number of occurrences of 0 is __, of 1 is __, of 2 is __, of 3 is __, of 4 is __, of 5 is __, of 6 is __, of 7 is __, of 8 is __, of 9 is __.

Feeling intelligent? If so, send us your method and solution (in English or, even better, as machine intelligence, in a program).

Meanwhile, I forget the question.

For more information—

- Johnson, George. *Machinery Of The Mind*. A Tempus Book from Microsoft Press. 1986.



NEW!

TOOLKITS FOR TURBO C & QUICK C from ZORTECH INC.

HOTKEY

A complete set of Terminate Stay Resident (TSR) functions that help you to write reliable 'pop-up' programs.

Now you can make your programs 'Sidekickable'. Two example programs are included, a 'pop-up Calculator' and a pop-up 'Critical Error Handler'.

The Hotkey toolkit handles all floating point functions in resident mode.

The 32 page manual includes an interesting discussion of the origin and history of undocumented MS-DOS function calls, together with a full explanation of the theory and practical use of TSR's.

Only \$49.95! (State Turbo C or Quick C version.)

COMMS

Do you need to incorporate serial communications into your applications? Yes! Then get this inexpensive but highly professional COMMS toolkit from Zortech Inc.

Look at the list of features: Xmodem, Kermit and ASCII file transfer, Hayes modem control, VT52, VT100 and ANSI terminal emulation, supports up to 8 serial ports, speeds up to 19.2k baud rate and higher.

Two demonstration programs are included, MINICOM and MAXICOM (like Procomm) together with the 120 page manual and full source code FREE!

Only \$49.95! (State Turbo C or Quick C version.)

GAMES

Have you ever wondered how to write a chess program? Now we reveal the secret algorithms and techniques of the masters with this dynamic Games toolkit.

The package comes complete with the full source code to three ready to play games of strategy - Chess, Backgammon and Wari (an ancient African game).

A comprehensive 150 page manual is provided giving an in depth look at the history, structure and program design of such 'Strategy Games'.

Only \$49.95!

(State Turbo C or Quick C version.)

SUPERTEXT

This is not simply an 'Editor' toolkit, but a full-blown, 'WordStar' compatible wordprocessor with the full source code.

As well as all the normal editing functions, you will also find 'do' commands and full printer control. The SuperText toolkit handles files of any size and allows full on-screen configuration.

Do you need to incorporate a wordprocessor into your application? Yes! Then get the SuperText toolkit complete with full source code and 150 page manual now!

Only \$49.95! (State Turbo C or Quick C version.)

PROSCREEN

Generate high quality data entry screens with the Pro-Screen - Screen Designer and Code Generator.

You can draw the data entry screen, define the input fields, define the input criteria, set screen colors and attributes, draw single or double lines, make boxes - press a few buttons and 'hey presto' Pro-Screen generates the C source code for your application!

Professional applications programmers will find this versatile utility and it's associated functions invaluable.

Comes complete with a substantial 78 page manual and demo programs.

Only \$49.95! (State Turbo C or Quick C version.)

**ONLY
\$49.95
EACH**

WINDOWS

Add super-fast text screen handling to your applications with the WINDOWS library from Zortech Inc.

Give your applications the professional look - with instant zooming and exploding windows. Incorporate drop-down menus and Lotus style menus with our easy to use functions.

Automatically handles memory saving and buffering of window text. Use any number of overlapping windows in your applications. Write to any window, read from any window, close any window, pull any window to the top.

Over 55 functions together with a big 85 page manual and remember, you get the full source code.

Only \$49.95! (State Turbo C or Quick C version.)

NEW! C VIDEO

- Now learn C the easy way!
- Get the 'Complete C Video Course' from Zortech Inc. together with our big 365 page workbook.
- Ten 1 hour tapes - 36 lessons!
- Easy to follow course, you get an excellent introduction to the C language.
- Takes you step-by-step up to the intermediate and advanced levels.
- Teach yourself at home or the office - at your own speed.

only ~~\$295.00~~ **\$199.95**

Yes!
Rush me
these items!

HOTKEY
 COMMS
 PRO-SCREEN

WINDOWS GAMES
 SUPERTEXT C VIDEO

FREE SHIPPING - VISA/MC/COD/CHECK

Name

Address

Phone

VISA or MC# Exp. Date

ZORTECH Inc. 361 Massachusetts Ave, Arlington, MA 02174

Orders & Enquiries Tel: (617) 646 6703

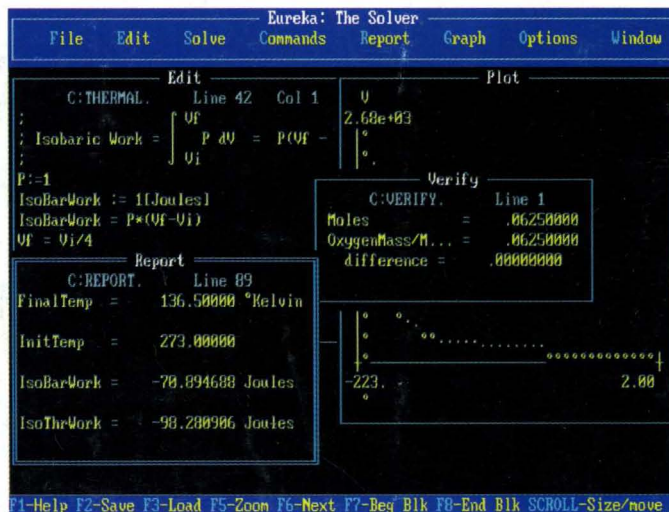
ORDER HOTLINE (800) 848-8408

Eureka: Instant Answers!

Now solving scientific and engineering equations goes from agonizing to easy!

Eureka™: The Solver and your PC can handle any problem you can hand them—instantly. Just type in any linear or non-linear equation, select "Solve," and look at your answer. But that's not all, because Eureka can do a lot more. Like evaluate your solution and plot a graph. Or generate a complete report, sending the output to your screen, disk file or printer.

You can check the equation itself or ask Eureka to continue searching iteratively for a satisfactory solution. Or, change the active variables or constants and search for an entirely different solution. Using Eureka's windows, you can solve up to 20 equations at once. It's all done with pull-down menus, full help screens and absolutely no agony. And all for just \$167.



Eureka instantly solved this Physics equation by immediately calculating how much work is required to compress isobarically 2 grams of oxygen initially at STP to 1/2 its original volume. In Science, Engineering, Finance and any application involving equations, Eureka gives you the right answer, right now!

Easy as a Calculator . . . Powerful as a Mainframe!

- Pull-down menus
- Full-screen text editor
- Context sensitive help
- On-screen calculator
- Automatic 8087 math coprocessor support
- Inequality constraints
- Powerful built-in and user-defined math and financial functions
- Report generation complete with plots and lists
- Polynomial root finder

“ Merely difficult problems Eureka solved virtually instantaneously; the almost impossible took a few seconds.
Stephen Randy Davis, PC Magazine ”



For the dealer nearest you
or to order call

(800) 543-7543

60-Day Money-back Guarantee*
Reader Service Number 1

System Requirements: For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles. PC-DOS (MS-DOS) 2.0 and later. 384K.

*Customer satisfaction is our main concern; if within 60 days of purchase this product does not perform in accordance with our claims, call our customer service department, and we will arrange a refund.
All Borland products are trademarks or registered trademarks of Borland International, Inc. Other brand and product names are trademarks or registered trademarks of their respective holders.
Copyright ©1987 Borland International, Inc.

BI 1179

BORLAND