

MICRO CORNUCOPIA

The Great C Issue

11 C Compilers page 8

Scott Ladd compares benchmarks, features, and generated code for 11 C compilers. (Including Microsoft C Vs 5.0 and Borland's Vs 1.5.)

Writing A Simple Parser In C page 20

Jack Purdum shows you how to use text files as program data.

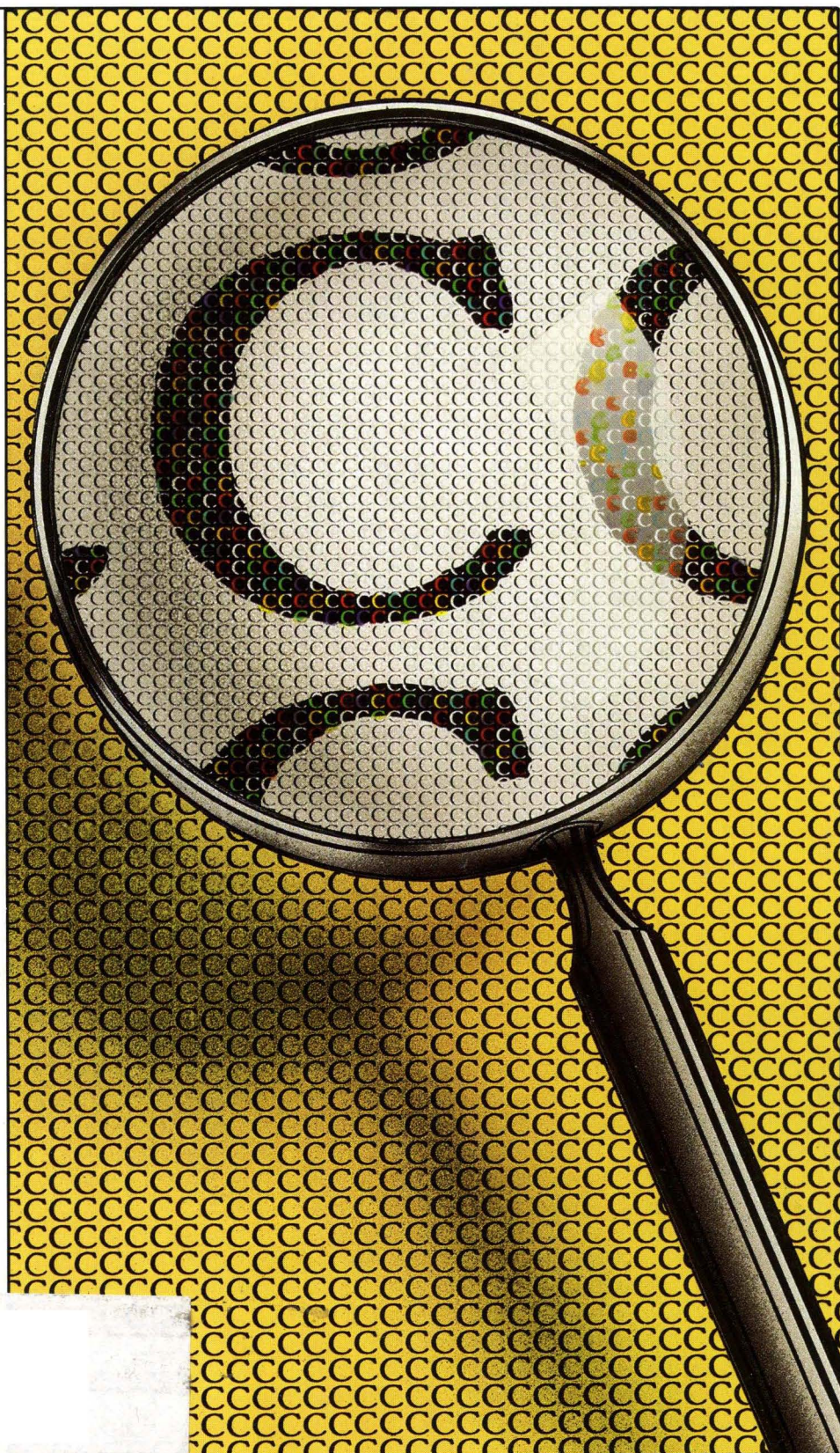
C++ , An Object Oriented C page 32

This fancy new language thinks it's C, Smalltalk, and more.

Source Level Debugger For Turbo C page 54

Every C needs a debugger. Here's a free one for Turbo written by Gary Mellor.

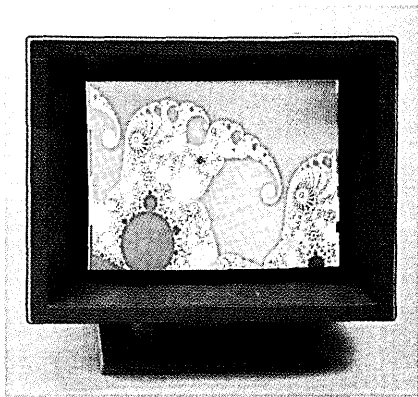
PLUS: In-depth looks at: A fast new AI language, a great new disassembler, PC Keyboards, the PC parallel IC, and much, much, more.



02

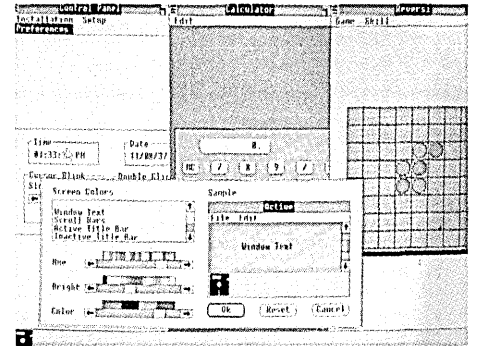
VERY HIGH RESOLUTION

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:

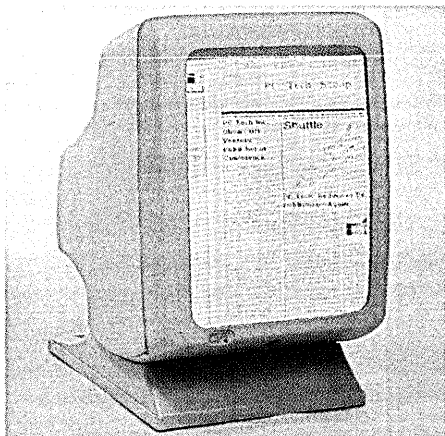
- Featured on the cover of Micro Cornucopia.
- From 800 x 512 through 1024 x 800 resolution (depending on monitor and configuration).
- 8 Bits per pixel for 256 simultaneous colors
- Hardware support for CGA/MDA emulation.
- PC, XT, and AT compatible



Available configurations:

- Basic Color 34010 processor board and software **\$995.00***
- "Loaded" Color 34010 processor board and software **\$1,200.00***
- Complete video sub-system (monitor, SW, Color 34010, cables)**
- Complete Color 34010 system based on the PC Tech X16**

** (Consult PC Tech for price, specifications, and other options)



Monochrome 34010 Video Processor:

- 800 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

Available configurations:

- Monochrome 34010 processor board and software **\$495.00***
- Complete video sub-system (monitor, SW, Monochrome 34010, cables) **\$995.00***
- Complete Monochrome 34010 system based on the PC Tech X16**

** (Consult PC Tech for price, specifications, and other options)

Introductory Special Window - \$995 complete mono system

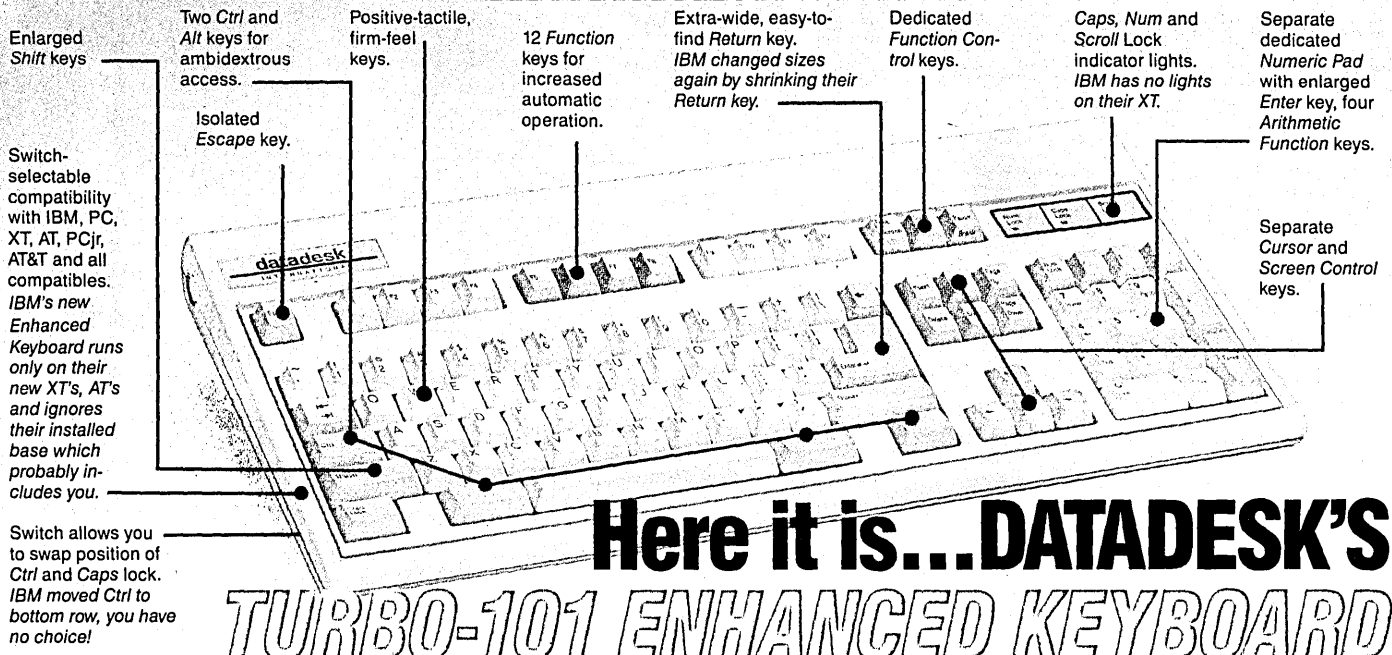
***SPECIAL LIMITED TIME OFFER:** For a LIMITED TIME PC Tech is offering Color and Monochrome adapters as well as complete systems at special introductory prices. All prices marked with an asterisk (*) reflect special prices in effect for a limited time.

Designed, Sold and Serviced By:



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

Borland's Turbo Lightning FREE



Here it is...DATADESK'S TURBO-101 ENHANCED KEYBOARD for the 10 million PC users IBM just ignored!

IBM just announced their new redesigned "standard" keyboard for personal computers. There's only one problem: it won't work on your IBM computer if it was purchased prior to June 1986 or on any PC compatible purchased at any time!

Not to worry. Our new Turbo-101 Enhanced Keyboard gives you the layout and enhancements of the IBM with some logical improvements (see above photo). And it works on your existing PC, XT, AT, PCjr, AT&T, Epson and virtually all compatibles!

Get Borland's Turbo Lightning™ For FREE!

To really turbocharge your productivity, we are including, free-of-charge, Borland's red-hot Turbo Lightning software with each keyboard. Now, when using SideKick, WordPerfect, Microsoft Word, 1-2-3 or most popular programs, our Turbo-101 Keyboard will check your spelling as you type, gives you instant access to Random House's 80,000-word Concise Dictionary and 60,000-word Thesaurus and much, much more!

"Lightning's good enough to make programmers and users cheer, executives of other software companies weep," says Jim Seymour of PC Week. Sold separately, Turbo Lightning retails for \$99.95!



The Turbo-101 is the best data entry tool since the pencil!

For users of spreadsheets like 1-2-3, the Turbo-101's separate cursor controls and numeric keypad makes entering numeric data into cells and moving from cell to cell as natural as moving your fingers. And for word-processing, the 'Selectric' typewriter layout makes the Turbo-101 as easy to use as a pencil; and with the extra large Enter, Shift & Control Keys, you'll make so few mistakes, you won't even need an eraser!

SPECIAL OFFER!

ONLY \$149.95* FOR BOTH
KEYBOARD & SOFTWARE

Includes 30-day money back guarantee and 2 year full warranty. To prove that we don't ignore you or your pocketbook, you get our Turbo-101 Enhanced Keyboard and Borland's Turbo Lightning for an astounding \$149.95.* No, you didn't read it wrong. During this amazing **Introductory Offer** you get both keyboard and software for less than most software programs by themselves! Now, if you're still feeling ignored, you can always do what you-know-who wants you to do...and buy a new computer to get their keyboard!

credit card orders call
(800) 826-5398
in CA call
(800) 592-9602

*Price does not include adaptor cables required by certain compatibles • A Limited offer—price subject to change without notification.



Up to now, DataDesk International may be one of the best kept secrets, but here's what's being said about our first end-user Keyboard/Borland software bundle:

"Who Can Pass Up a Deal? Department. Talk about an aggressive product!"
John C. Dvorak,
InfoWorld Mar 86

"It solves all of the problems exhibited by their regular PC/XT keyboard...it's a great bargain!"
PC Productivity Digest
May 86

"DataDesk Intl. has designed a sturdy and handsome keyboard that has tactile response...is the hardware bargain of the year" says Charles Humble, Oregonian
Jan 86

"The best part of the keyboard is the way it feels. It's ideal! And fast. I've never worked on a keyboard with a nicer touch."
Business Computer Digest
Aug 86

"It's a good keyboard. Good feel. The keys have tactile feedback. No mush at all. This is about as good a keyboard deal as you're likely to find...I have absolutely no hesitation in recommending the Model PC8700."
Jerry Pournelle,
Byte Magazine Sept. 86

"This keyboard is neat to type on and feels solid. It has tactile feedback keys...I can type much faster on it."
Test Drive Scorecard:
DataDesk-10 Key Ironics-9
Teleconnect Magazine
May 86

The Only Alternative
datadesk™
INTERNATIONAL

7650 Haskell Avenue
Van Nuys, California 91406 (818) 780-1673

Turbo-101 is a trademark of DataDesk International. Turbo Lightning is a trademark of Borland International. IBM and IBM AT are registered trademarks of International Business Machines, Inc.

Reader Service Number 8

**BOTH TURBO-101 ENHANCED™
KEYBOARD AND BORLAND'S TURBO
LIGHTNING™ SOFTWARE FOR ONLY:
\$149.95* LIMITED
OFFER**

*Plus \$10 Shipping & Handling.
California Residents
Add \$9.75 Per Unit Sales Tax

NO. UNITS: _____
PAYMENT: VISA MC CHECK
AMOUNT ENCLOSED: \$ _____

Enter

NAME: _____
ADDRESS: _____
CITY: _____ STATE: _____ ZIP: _____
PHONE: _____
CC NO.: _____ COMPUTER TYPE: _____
EXP: _____

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

XT KIT W/ 2 Floppy Drives.

Includes: 640 K 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 standard slide cabinet 649.00
10mhz with lock, LED, Reset & Turboswitch 699.00

XT KIT W/20MB Hard Drive.

Includes: 640 K 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 standard slide cabinet 949.00*
10mhz with lock, LED, Reset & Turboswitch 995.00*
*(For 30MB Miniscribe add \$50.00)



— Pictured keyboard is 5339 —

NEW!

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.

2875.00

80286 - AT KIT

Includes: 640K 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.

1795.00

CASES & POWER SUPPLY

150 Watt Power Supply (XT) 59.00
200 Watt Power Supply (AT) 99.00
XT Slide Case 32.00
XT Flip Top or XT Slide with Lock & LED 49.00
AT with Lock & LED 65.00

MONITORS

EGA/CGA (Auto Switch) 495.00
VGA/EGA/CGA Color 650.00
CGA Color 339.00
Amber 12" TTL 89.00
Green 12" TTL 89.00

VIDEO CARDS

Color/Graphics 55.00
Color/Graphics/Parallel 60.00
256K EGA Graphics 140.00
Mono/Graphics/Parallel 60.00
ATI Graphics Solution—
Mono, Herc. Color Emulation on Mono CGA (List 299) 150.00
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/clk/game/2 floppy 79.00
Parallel (printer) 19.00
Serial Port (RS232) 1 port 29.00
640K RAM (ØK 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 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

Free Instructions with Each System

MOTHERBOARDS

XT/Turbo 4.77/10mhz 119.00
AT 6/10 mhz (4 layer) Choice of Phoenix or DTK Bios 350.00
XT/Turbo 4.77/8 mhz (2 layer) 109.00
XT/Turbo 4.77/8 mhz (4 layer) 119.00
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 97.00
Toshiba 360K 99.00
Toshiba 1.2 MB 145.00
3½" Drive Kit 145.00

KEYBOARDS

5339 Professional XT-AT w/12 function key 79.00
5060 Keyboard AT Style 55.00

FROM THE INSIDE OUT: TECH TIP —

If you are considering an 80386 system, find a motherboard built in the USA. Due to chip shortage, Intel has been shipping seconds overseas. USA manufacturers get the better chips. If AMD starts making chips soon, the prices will drop.

HARD DRIVES & CONTROLLERS

AT 40 MB Seagate #251 450.00
AT Hard Drive & floppy controller (WD) 145.00
20 MB Miniscribe HD with controller 349.00
30 MB Miniscribe HD with controller 399.00
AT 40MB NEC HD 695.00
20 MB NEC HD with controller 449.00
30 MB NEC HD with controller 499.00

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

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

MARCH/APRIL 1988 - ISSUE NO. 40

FEATURES

8 Scott Ladd
11 MS-DOS C Compilers Compared

Are Borland and Microsoft the only real contestants in the C compiler battle? Nope. Scott Ladd gives us a blow by blow comparison of the whole field, complete with a look at the code they generate.

CCC

20 Jack Purdum
Programming For Change

Jack adds a simple parser to his programs so they can take data from ASCII text files.

26 David Thompson
Comdex, A Hacker's Computer Fix

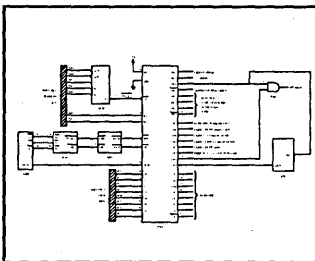
There was more than hype at the Fall Comdex. This year I found a great disassembler and an intriguing new language.

32 Bruce Eckel
A Programmer's Introduction To C++

An object oriented language that thinks it's still C? Yep.

36 Larry Fogg
Inside The PC's PPI

Larry winds down his special on the PC's smart chips (no dips) with a close look at the potentially boring 8255.



42 Eric Isaacson
C Vs. Assembly Language

This is a C issue, right? So we've asked Eric to help us bash assembly language, right? Wrong.

48 James H. Simpson
Turbo Pascal 4.0

You've enjoyed Turbo Pascal but found it was too limited for serious projects? Not anymore.

COLUMNS

54 C'ing Clearly

Turbo C doesn't have a debugger. At least not until now. Here's a source level pest chaser that's free. Really!

58 86 World

Laine comes back to reality (technically speaking) with an intimate look at PC keyboards.

68 On Your Own

Got reviews coming out on your zingy new product? Great. Should you place ads to run with the reviews? If so, how do you go about it?

76 Pascal Column

84 Shareware

93 Technical Tips

CP/M CORNER

72 Kaypro Column

FUTURE TENSE

80 Tidbits

96 Last Page

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

Comdex Past

Disk Cash

It's 9:30 p.m. Still early. The spoils of Comdex past tower over my desk in unsteady stacks. Interesting, uninteresting, interesting, uninteresting, very interesting, very, very interesting...

What's this... Why do I have a box of Verbatim disks?

Because Verbatim was giving away boxes of disks.

Why were they giving away boxes of disks?

Hold on, I'm looking.

Oh, yeah. Verbatim was bought out by Kodak and they've come up with a teflon coating that protects the media. You can supposedly spill anything on it, boiling hot coffee, white-out, catsup (or ketchup). At least that's what their P.R. person said. (See, it's right here in my notes.)

Okay, I'll cut open the disk's jacket and pull out the floppy little center (so I'll have a clean jacket to put the disk into after the fun).

Let's see, I want to get fingerprints all over the surface (this disk likes fingerprints). Where's the bottle of white-out? Darn, it's dried up. Some white indelible ink should work.

Now, to the kitchen. Aha, peanut butter, catsup, and hot sauce (VERY hot sauce). I haven't had this much fun since I edited an IBM punch card with a sewing machine.

Gentlemen: The Results

Anyway, it was great fun. I covered the disk with goeey edibles and then dropped it into a pan of boiling water (couldn't be any worse than hot coffee). After about a minute the edges curled up, so I flipped it over. (Equally done on both sides.)

Finally, I got some tissue from the kids' bathroom and cleaned off the mess (we've got to get creamy peanut butter before I test disks again), popped the disk back into the jacket and presto, a new disk. Well, pretty new.

I stuck it in drive A:

"Sector error."

I tried reformatting it.

"Are you kidding?" (My computer will be playing Hal in the next 2001.)

Okay, maybe I overdid it.

"That's hardly the word for it." (When computers

(Continued on page 87)

Blaise puts the Accent on C with C TOOLS PLUS/5.0™

Enhance your Microsoft C programming environment with C TOOLS PLUS/5.0™—a new, quintessential library of C functions. C TOOLS PLUS/5.0 from Blaise Computing Inc. puts a prime accent on quickly building professional applications using the full power of Microsoft C Version 5.0 and QuickC. Now you can concentrate on program creativity by having full control over DOS, menus, interrupt service routines, memory resident programs, printer and keyboard control, and more!

C TOOLS PLUS/5.0 prebuilt libraries are ready to use with either QuickC or the Microsoft C Version 5.0 command line environment. Complete documented source code is included so that you can study and adapt it to your specific needs. Blaise Computing's attention to detail, like the use of full function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes C TOOLS PLUS/5.0 the choice for experienced developers as well as newcomers to C.

Continuous refinement of Blaise Computing's library products has produced a collection of tools that are unsurpassed for reliability, functionality and ease of use. Built upon the widely acclaimed C TOOLS PLUS, C TOOLS PLUS/5.0 includes such highly-developed features as:

◆ WINDOWS

- Detachable, removable.
- Optional borders, cursor memory.
- Accept user input, formatted output.
- "printf" window-oriented output. **NEW!**

◆ INTERRUPT SERVICE ROUTINES

- Capture DOS critical errors and keystrokes.
- Install hardware interrupt handlers.

◆ RESIDENT SOFTWARE SUPPORT

- Install, detect and remove memory resident programs.

◆ MENUS

- Horizontal and pulldown. **NEW!**
- Lotus-style support. **NEW!**

◆ INTERVENTION CODE

- Schedule C functions at specified times, intervals or with a "hot key." **NEW!**
- Take full advantage of DOS, even from memory resident programs. **NEW!**

◆ FAST DIRECT VIDEO ACCESS

- All monitors, even EGA 43-line mode.

◆ PRINTER CONTROL

- Access BIOS print functions. **NEW!**
- Control the DOS PRINT utility. **NEW!**

◆ UTILITIES AND MACROS

- Take advantage of DOS file structure.
- Manipulate data types, far & near pointers. **NEW!**
- Access any memory areas with fast "peek" and "poke" macros. **NEW!**

C TOOLS PLUS/5.0 supports the Microsoft C Version 5.0 and QuickC compilers, requires DOS 2.00 or later and is just **\$129.00**.

C ASYNCH MANAGER™ Version 2.0 IMPROVED!

C ASYNCH MANAGER is a library of functions designed to help you incorporate asynchronous communication capabilities into your application programs. Version 2.0 has been rewritten especially for Microsoft C Version 5.0 and Borland's Turbo C. Simultaneous buffered input and output to both COM ports at speeds up to 9600 baud, XON/XOFF protocol, modem control and XMODEM file transfer are among the many features supported and is priced at just **\$175.00**.

Blaise computing Inc. has a full line of support products for both Pascal and C. Call today for your free information packet.

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Reader Service Number 5

Now, just \$129 and supports
Microsoft C 5.0 and QuickC

**THE BLAISE
E N U**

VIEW MANAGER \$275.00

General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

Turbo C TOOLS \$129.00

Windows; ISRs; intervention code; screen handling and EGA 43-line text mode support; direct screen access; DOS file handling and more. For Turbo C.

Turbo POWER SCREEN

COMING SOON! General screen management; paint screens; block mode data entry or field-by-field control with instant screen access. For Turbo Pascal.

Turbo POWERTOOLS PLUS \$129.00

Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. Now supports Turbo Pascal 4.0!

Turbo ASYNCH PLUS \$129.00

Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. Now supports Turbo Pascal 4.0!

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.

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. For MS-Pascal.

KeyPlayer \$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; flexible printer control; user-defined variables; index generation; general macro facility. Crafted in Turbo Pascal.

LIGHT TOOLS \$99.95

Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

TO ORDER CALL TOLL FREE

800-333-8087

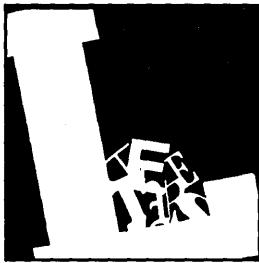
TELEX NUMBER-338139

Yes! Send me the prime accent!
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: _____ Zip: _____
City: _____ Exp. Date: _____
VISA or MC#: _____

Microsoft
is a registered trademark of
Microsoft Corporation. QuickC
is a trademark of Microsoft Corporation. Turbo C
is a registered trademark of Borland International.



Letters

In Defense Of Konan

Regarding your review of the Konan hard drive controller in Issue #38 of *Micro Cornucopia*: You didn't state which of the Konan boards you reviewed. I'd like to know if it was the KDC-230, KXP-230, or KXP-230Z?

My present system consists of a Leading Edge Model M, two Seagate 225s, and the Konan KXP-230Z. I use my computer both for developing software and as a hobby and have yet to see any problems that stem from the Konan board.

Some of the features you overlooked:

- Ability to configure drive from built in menu.
- Intelligent cache. You can keep most often used programs and data in memory.
- Supports drives up to 301 MB.
- Supports two different hard drives simultaneously.
- Can create one DOS and 15 EDISK (emulated disk) partitions on the first drive.
- Support diagnostics and utilities.
- Disk reorganization on-the-fly.
- Good telephone support.

While I haven't used my Konan in the high-speed configuration, or used various drives, I am quite pleased with my purchase. I highly recommend the Konan board.

Larry Kraemer
Route 2 Box 190
Jackson, MO 63755

Editor's note: Thanks for the comments Larry. I used the 230Z for the review. You mention the track buffering. I tried using the buffering but I turned it off after seeing how much memory it took. If you have expanded or extended memory these features make sense.

Also, I may have given the wrong im-

pression, I like the Konan. It's not perfect, but it has lots of features.

Oops!

I just moved to the Northwest and am really enjoying this part of the country (and your magazine). However, a recent issue contained an error pertaining to the Midwest (my former home).

The 86 World column in Issue #39 mentioned the "national teacher's hiring conference held every year at Iowa State University in Cedar Falls." Actually, the conference is held at the University of Northern Iowa in Cedar Falls, IA 50614. (I used to be on the computer staff there!)

This error is equivalent to confusing WSU with Western Washington!

Terry A. Ward
4269 148th NE #D-103
Bellevue, WA 98007

Editor's note: Your letter arrived the same day as the linebacker corps from UNI. (Now that we've placated you, we'll be tackling them.)

Desktop And Seagate

OK, OK, here's some bucks for another year. I couldn't miss another letter from Turkey (I'm not kidding).

Your articles on desktop publishing gave me the confidence to dive in. I purchased both Pagemaker and Ventura and although I'm hardly accomplished with either one yet, your evaluations seem pretty accurate.

For creating beautiful and internally consistent instruction manuals (primarily text but with some drawings), Ventura wins my vote. However, they need to do a lot of work on graphic input bugs and style sheets or someone will walk away with their market.

For brochures of one page or less I might tolerate Pagemaker, but it's awfully slow even on a 16 MHz 386 machine. Is that due to Windows? Pagemaker looks well suited to artsy-craftsy creations where you move things around until they look nice. (Why do I feel insulted by those stupid little icons? I thought I would get used to them, but after a month I *still* feel insulted.)

Here's another contribution to the hard drive discussion. I've used two Seagate ST-225s and a 4038 and none have lasted more than one year. My first Miniscribe is six years old and still sees daily use. A total of five 42 MByte Miniscribes acquired for home and work have also been flawless. Miniscribe makes an excellent product.

Bill Carver
25961 Ave Romero
San Juan Capistrano, CA 92675

More Nails In Seagate's Coffin

I really like your new cover and format. Desktop publishing is the way to go. You must have a graphic artist lurking somewhere.

In May I bought a Priam/Vertex VT 170. This 60 MB (formatted) drive has an access time of less than 28 msec. Set up with a DTC RLL controller, it runs all day, six days a week. I've had no problems except for what I perceive to be the controller misbehaving slightly.

Once in a hundred times it will boot up but fail to load AUTOEXEC.BAT. Rebooting cures the problem. Also, the controller is extremely finicky about which 360 KB disks it will choose to read. The Western digital controllers seem much less discriminating.

I use SpeedStore (partly because of your rave reviews). I found it very

(Continued on page 75)

Call or write
for the latest catalog

LIST OURS

TURBO PASCAL ADD-ONS	75	69
DOS/BIOS & MOUSE TOOLS	89	79
FLASH-UP	100	89
METRAYBYTE DATA ACQ. TOOLS	125	89
SCREEN SCULPTOR	150	129
SYSTEM BUILDER	100	89
IMPEX	130	115
REPORT BUILDER	60	49
T-DEBUG PLUS	99	69
TURBO ASM	129	99
TURBO ASYNCH PLUS	85	65
TURBO EXTENDER	99	79
TURBO HALO	199	179
TURBO MAGIC	75	65
TURBO OPTIMIZER	129	99
TURBO POWER TOOLS PLUS	95	79
TURBO POWER UTILITIES	95	79
TURBO PROFESSIONAL 4.0	NEW	99
TURBO WINDOW/PASCAL	95	79

FEATURED PRODUCTS

MICROSOFT WINDOWS/386 — Software for 386-based machines offering true multi-tasking. Run several programs within different windows on your screen. Cut and paste between them. Each program is given 640K of memory.
List: \$195 **Special Price: \$119**

MICRO FOCUS COBOL/2 — Compiler system that can exploit the full memory of 80286/386 machines under OS/2 or under PC-DOS. Conforms to the highest certifiable level of ANSI/85 COBOL. Includes the ANIMATOR source code debugger and offers full network support.
List: \$900 **Special Price: \$729**

PANEL/QC OR TC — This best selling screen management library is now available for both Quick C and Turbo C. Supports pop-up fields and windows, multi-line fields, horizontal and vertical field scrolling, menus, help boxes, and custom field validation. Generates C source code. No royalties.
List: \$129 **Special Price: \$89**

FETCH — New memory resident file librarian, for those of us who have trouble finding files, or forget what's in them. The user is prompted for a 255 character description anytime a file is created. Later, Fetch can scan the file description library with its fast pattern recognition capabilities, and will display the description, directory, and drive.
List: \$55 **Special Price: \$45**

OPERATING SYSTEMS		
MICROPORT SYSTEM V/AT	549	465
SCO XENIX SYSTEM V	1295	989
WENDIN-DOS	99	79
OTHER MICROPORT, SCO, WENDIN PRODUCTS	CALL	CALL

PASCAL COMPILERS			
MARSHAL PASCAL	189	155	
MICROSOFT PASCAL	304	185	
PASCAL 2	350	319	
TURBO PASCAL	NEW V. 4.0	100	65
TURBO PASCAL DEV. LIB.	NEW	395	259
BORLAND ADD-ONS	CALL	CALL	

SCREEN DISPLAY/WINDOWS			
C-SCAPE	279	265	
CURSES W/SOURCE CODE	250	169	
GREENLEAF DATA WINDOWS	225	155	
W/SOURCE CODE	395	259	
HI-SCREEN XL	149	119	
JVACC FORMAKER	495	449	
JVACC JAM	99	65	
MICROSOFT WINDOWS	500	309	
MS WINDOWS DEVELOPMENT KIT	495	395	
PANEL PLUS	129	89	
PANEL/QC OR TC	SPECIAL	129	89
QUICKSCREEN	195	175	
LINKSTAR W/SOURCE	195	169	
VITAMIN C	225	149	
VC SCREEN	99	79	
VIEW MANAGER	275	199	
WINDOWS FOR DATA	SPECIAL	295	229

HARDWARE PRODUCTS

AMDEK 722 MONITOR	750	499
AMDEK 730 MONITOR	899	569
AST ADVANTAGE PREMIUM W/512K	495	319
AST RAMPAGE! 286 W/512K	545	349
HERCULES GRAPHICS CARD PLUS	299	195
HERCULES IN COLOR CARD	499	329
IRMA 2	1195	779
ORCHID TURBO EGA	749	495
ORCHID TURBO PGA	1495	1099
PARADISE SYSTEMS		
VGA PROFESSIONAL	599	409
AUTOSWITCH EGA 480 CARD	349	169
VEGA DELUXE	379	259

ADDITIONAL PRODUCTS

ADVANTAGE VCMs	379	329
BASTOC	495	399
CARBON COPY PLUS	195	159
DAN BRICKLIN'S DEMO PROGRAM	75	59
DB2C	299	CALL
FLOW CHARTING II	229	205
MAGIC PC	195	179
MKS TOOLKIT	139	115
NORTON GUIDES	100	65
FINISH	395	209
POLYMAKE	149	125
POSTRONS PVCs	55	CALL
SOURCE PRINT	95	75
TREE DIAGRAMMER	77	69



Programmer's Paradise Gives You Superb Selection, Personal Service and Unbeatable Prices!

Welcome to Paradise. The microcomputer software source that caters to your programming needs. Discover the Many Advantages of Paradise...

- Lowest price guaranteed
- Latest versions
- Huge inventory, immediate shipment
- Knowledgeable sales staff
- Special orders
- 30-day money-back guarantee

Over 500 brand-name products in stock — if you don't see it, call!

We'll Match Any Nationally Advertised Price.

386 SOFTWARE			
ADVANTAGE 386 C OR PASCAL	895	799	
MICROPORT SYSTEM V/386 (COMPLETE)	799	679	
MS WINDOWS/386	SPECIAL, NEW	195	119
PHARLAP 386 ASM/LINK	495	419	
PHARLAP 386 DEBUG	195	155	
SCO XENIX SYS V 386 (COMPLETE)	195	119	
VM/386	SPECIAL	195	119
X-AM	595	535	
ARTIFICIAL INTELLIGENCE			
ARITY STANDARD PROLOG	95	79	
MULISP-87 INTERPRETER	300	199	
PC SCHEME	95	85	
SMALLTALK/V	NEW V. 2.0	99	85
TURBO PROLOG	100	65	
TURBO PROLOG TOOLBOX	100	65	
ASSEMBLERS/LINKERS			
ADVANTAGE DISASSEMBLER	295	269	
ADVANTAGE LINK	395	359	
ASMLIB	149	125	
EZ-ASM	70	65	
MS MACRO ASSEMBLER	150	95	
PASM86	195	109	
PLINK86PLUS	495	275	
RELSMS, UNIWARE X-ASMS	CALL	CALL	
VISIBLE COMPUTER 80286	100	89	

BASIC		
DB/LIB	139	119
FLASH-UP	89	79
MACH 2	75	59
MS QUICKBASIC	99	65
QUICKPAK	69	59
TRUE BASIC	100	69
TURBO BASIC	100	65
DATABASE TOOLBOX	100	65
EDITOR TOOLBOX	100	65
TELECOM TOOLBOX	100	65

XENIX/UNIX PRODUCTS

MICROPORT & SCO PRODUCTS	CALL	CALL
ADVANTAGE C++	695	CALL
BTRIEVE	595	455
C-TEP	498	379
INFORMIX ESQ/LC	749	CALL
INFORMIX 4GL	1500	CALL
INFORMIX SQL	995	CALL
KORN SHELL	125	115
MICROSOFT LANGUAGES	CALL	CALL
PANEL	625	535
PANEL PLUS	195	675
REAL TOOLS	149	89
RM/COBOL	1250	949
RM/FORTRAN	750	549
SCO MULTIVIEW (286)	395	319
SCO MULTIVIEW (386)	495	399

C++			
ADVANTAGE C++	495	479	
PFORCE++	395	209	
C COMPILERS			
C86PLUS	497	375	
LATTICE C	500	265	
MICROSOFT C	450	269	
QUICK C	SPECIAL	99	59
TURBO C	100	65	

C INTERPRETERS			
C-TEP	298	219	
INSTANT C	495	369	
RUN/C	120	79	
RUN/C PROFESSIONAL	250	155	
C LIBRARIES			
BASIC C	175	129	
C ASYNCH MANAGER	175	135	
C-FOOD SMORGASBORD	150	95	
W/SOURCE CODE	300	179	
C TOOLS PLUS/5.0	129	99	
C UTILITY LIBRARY	185	119	
C-XPERT	395	339	
ESSENTIAL COMMUNICATIONS	185	119	
COMMUNICATIONS PLUS	250	189	
GREENLEAF C SAMPLER	95	69	
GREENLEAF COMM LIBRARY	185	125	
GREENLEAF FUNCTIONS	185	125	
MULTI C	149	135	
PERICE	SPECIAL	285	199
RESIDENT C W/SOURCE	198	169	
TIMESLICER	295	265	
W/SOURCE CODE	1000	895	
TURBO C TOOLS	129	99	

COBOL			
E-Z PAGE	295	259	
MICRO FOCUS			
COBOL/2	SPECIAL	900	729
COBOL/2 TOOLSET	NEW	900	729
PC-CICS	1500	CALL	
LEVEL II COBOL	349	279	
PERSONAL COBOL	149	119	
OTHERS	CALL	CALL	
MICROSOFT COBOL	700	429	
MICROSOFT SORT	195	129	
OPT-TECH SORT	149	99	
REALICS	995	785	
REALIA COBOL	995	785	
W/REALMENU	1145	899	
RM/COBOL	950	759	
RM/COBOL-85	1250	999	
RM/SCREENS	395	315	
SCREENIO	400	379	

DEBUGGERS		
ADVANCED TRACE-86	175	115
C-SPRITE	175	119
PERISCOPE I	345	275
PERISCOPE II	175	139
PERISCOPE III 8 MHZ	995	799
PERISCOPE III 10 MHZ	1095	875
PPIX 86 PLUS	395	209
T-DEBUG PLUS	60	49
XVIEW86	60	49

DISK/DOS/KEYBOARD UTILITIES			
COMMAND PLUS	80	69	
DISK OPTIMIZER	60	55	
FETCH	SPECIAL	55	45
INTELLIGENT BACKUP	150	135	
NORTON COMMANDER	75	55	
ADVANCED NORTON UTILITIES	150	99	
PDISK	145	99	
VFEATURE	80	75	
VFEATURE DELUXE	120	110	

EDITORS		
BRIEF	195	CALL
W/DBRIEF	275	CALL
EDIX	195	155

EMACS			
EMULCON	295	265	
KEDIT	195	149	
PC/EDT	125	99	
PC/VI	250	229	
PI EDITOR	149	109	
PMATE	195	109	
SPEX/PC	195	149	
VEDIT PLUS	185	129	
FILE MANAGEMENT			
BTRIEVE	245	185	
XTRIEVE	245	185	
REPORT OPTION	99	145	
BTRIEVE/N	595	455	
XTRIEVE/N	595	455	
REPORT OPTION/N	345	269	
CBTREE	159	139	
C-TREE	395	315	
R-TREE	295	239	
C-TREE/R-TREE BUNDLE	650	519	
DBC III	250	169	
DBC III PLUS	750	595	
DB-VISTA OR DB-QUERY	195	159	
SINGLE USER W/SOURCE CODE	495	399	
MULTIUSER	495	399	
MULTIUSER W/SOURCE CODE	990	789	
INFORMIX PRODUCTS	CALL	CALL	
PHACT MANAGER	249	219	
XQL	NEW	795	599

FORTRAN COMPILERS		
LAHEY FORTRAN	477	CALL
LAHEY PERSONAL FORTRAN 77	95	89
MICROSOFT FORTRAN	450	269
RM/FORTRAN	595	479

FORTRAN UTILITIES/LIBRARIES		
DIAGRAMER OR DOCUMENT'ER	129	115
DIFF-E-Q	495	445
FORTRAN ADDENDA	165	139
GRAFATIC OR PLOTMATIC	135	119
MAGUS NUMERICAL ANALYST	295	249
MATHPAC	495	445
NO LIMIT	129	109
SPINDRIFT LIBRARY	149	135
SSP/PC	350	269

GRAPHICS			
ADVANTAGE GRAPHICS (C)	250	225	
ESSENTIAL GRAPHICS	250	185	
GSS GRAPHIC DEV. TOOLKIT	495	375	
HALO	SPECIAL	300	199
HALO (5 MICROSOFT LANG.)	595	389	
METAWINDOW PLUS	275	229	
TURBOWINDOW/C	95	79	
TURBO HALO (FOR TURBO C)	99	79	

LINT		
PC-LINT	139	99
PRE-C	295	155
MODULA-2		
LOGITECH MODULA-2	99	79
COMPILER PACK	249	199
DEVELOPMENT SYSTEM	169	139
TOOLKIT	49	39
WINDOW PACKAGE	299	239
ROM PACKAGE AND CROSS	89	75
RUNTIME DEBUGGER		
REPertoire		

Terms and Policies
 • We honor MC, VISA, AMERICAN EXPRESS
 No surcharge on credit card or C.O.D. Prepayment by check. New York State residents add applicable sales tax. Shipping and handling \$3.00 per item, sent via UPS ground. Rush service available, prevailing rates.
 • Programmer's Paradise will match any current nationally advertised price for the products listed in this ad.
 • Prices and Policies subject to change without notice.
 • Hours 9AM EST — 7PM EST
 *Ask for details. Some manufacturers will not allow returns once disk seals are broken.
Corporate Buyers — Call for special discounts and benefits!

1-800-445-7899
In NY: 914-332-4548
 Customer Service:
914-332-0869
 International Orders:
914-332-4548
 Telex: 510-601-7602

Programmer's Paradise
 A Division of Hudson Technologies, Inc.
 42 River Street, Tarrytown, NY 10591



Evaluating 11 MS-DOS C Compilers

C-ing The Forest For The Trees

The competition among vendors of C compilers has become quite intense over the last two years. What used to be considered optional is now essential. New features and benchmark speeds are touted as proudly as national flags at the Olympics.

The recent introduction of low-priced (under \$200) compilers offering many of the features of their \$400+ brothers has really warmed things up.

Reviewing 11 compilers is not as easy as I had originally expected. Benchmarks are a handy tool in these evaluations, but they only look at a few aspects of a compiler. What about ease of use, library completeness, and ANSI compatibility? Many of the products also come with extra utilities, including debuggers, assemblers, linkers, and librarians. All of these have to be taken into account when judging the contestants.

I've included the following compilers in this review:

Borland Turbo C v1.0D
C-Ware DeSmet C v3.03
Computer Innovations C86Plus v1.09g
Datalight Optimum-C v3.11
EcoSoft Eco-C88 v4.02
Lattice C v3.21
Manx Aztec C Developers Kit v4.10a
Mark Williams' Let's C v4.0.12
Metaware High C v1.4
Microsoft Optimizing C v5.00
Microsoft Quick C v1.00

Benchmarking

Creating benchmark programs is an arcane art. For example, traditional benchmark programs (such as the Fibonacci or Sieve tests) have been invalidated by modern "optimizing" compilers. These compilers recognize the

benchmarks and generate special code for them.

The benchmarks I wrote fall into two broad categories: "composite" and "specific." Composite benchmarks use many different types of functions, statements, and data-types in longer programs. Specific benchmarks test only one portion of a compiler, such as floating point operation.

Editor's note: Scott's benchmarks are available on the Micro C RBBS (503-382-7643 8-1-N 24 hrs) or on our Issue 40 disk (\$6 for subscribers, \$8 for non-subscribers and foreign, call 800-888-8087).

Composite Benchmarks

The composite benchmarks are DRYSTN, GRIND, and FXREF. DRYSTN is a slightly modified public domain dhrystone program, which contains a mix of statements corresponding to a typical program. Optimizing compilers which can recognize "dead" (un-executed) code should do well with DRYSTN.

GRIND was created to test floating point, looping, and file operations. It reads in 1000 floating point numbers, sorts them, and then calculates a table of trigonometric values. At completion, it writes the table to disk.

FXREF is a practical application. Designed as a filter, FXREF reads a file, displays it, builds a binary tree cross-reference table for text tokens, and prints the cross-reference. This tests dynamic memory allocation, looping, and file I/O.

Specific Benchmarks

The specific benchmarks are very short, and each tests only a few functions. Because they are limited, they give you a feel for the size of particular library functions (look at the size of the .EXE files).

CRUNCH reveals floating point cal-

culational and library performance. DISPLAY tests video I/O using printf(). DISKIO reads and writes a large disk file, while ALLOCMEM tests dynamic memory allocation and deallocation. Raw loop speed is checked by LOOPS, and SORT arranges a very large array of reverse-ordered numbers using a recursive QuickSort algorithm.

Figure 1 shows the results of the benchmark tests. In the case of compilers with optimizing passes, results are shown for compiles made with standard options and those made with full optimization. I ran all the benchmarks on an 8Mhz, V20-based, Kaypro Professional Computer with a fast Miniscribe 42-megabyte hard disk. Stack checking code was disabled for ALL compilers.

In some instances I modified a benchmark program to get it to compile. This was because of limited ANSI compatibility. For example, SORT uses function prototypes, which are not implemented in the DeSmet or Mark Williams compilers.

I also used a special benchmark program, CODEQUAL. It is a very short, simple program from which I recovered an assembler listing of each compiler's code. This gave me a comparison of the code each was generating.

As I pointed out earlier, benchmarks are only a part of the story. Figure 2 shows basic compiler features. Note that although the ANSI C standard has yet to be finalized, individual features of it (e.g., function prototypes, void, etc.) are firmly set.

Let's take a general look at each package and its features.

Borland Turbo C v1.0D

Turbo C is a spirited newcomer, arriving amid considerable anticipation and fanfare. Incorporating parts of the

well-regarded Wizard C compiler (purchased by Borland), Turbo C continues the Borland tradition of excellent performance and slick packaging at a relatively small price.

Unlike most other MS-DOS C compilers, Turbo C does not use environment variables to locate its header and library files; instead, it uses configuration files.

Installation of the product is simple and well-documented, and it will run on a two floppy-disk PC. However, they recommend a hard disk.

Documentation comes in two trade-sized paperback books, a user's guide and a reference. I found it difficult to find certain information in the manuals; they seem to be somewhat disorganized. Functions are documented in alphabetical order, but are not "one function per page" as in many manuals, so some are split across page boundaries.

Borland pioneered the integrated editor/compiler with Turbo Pascal, and has refined it for Turbo C. You can edit, compile, and run a program without leaving the window environment. If you prefer your own editor, Borland has included a standard command line version of both the compiler and linker.

Turbo C showed excellent compile speeds, good overall benchmark scores, and extraordinary floating-point performance. On the other hand, tests indicate slow display (especially printf()) and dynamic memory allocation functions.

The extensive library contains many procedures for accessing the PC BIOS and MS-DOS. The object modules use Microsoft conventions so you can use Turbo C with many commercial libraries.

Borland provides minimal support for third-party debuggers, but it's currently testing its own symbolic model (to be released sometime in 1988). Other

missing elements, such as an object module librarian and a graphics library, will be included in version 1.5. (See sidebar for the latest on version 1.5)

Turbo C generates very intelligent assembler code which makes good use of shifts and registers. The initialization code for the call to putchar() in CODEQUAL is rather large though. (See Figure 3 for a listing of CODEQUAL.C and the assembler output of each compiler.)

C-Ware DeSmet C v3.03

This compiler has been around for

quite some time and has a loyal following. It was one of the first lower-cost C compilers for MS-DOS. Just recently, C-Ware repackaged their compiler with additional libraries and utilities.

Documentation comes in a simple three-ring binder and appears complete, though somewhat sparse.

Installation is not automatic but it's quite simple and the package lives happily on a two-floppy system. (It takes less than 500K bytes on a hard disk.)

An editor (SEE), assembler, linker, and debugger all come in the package. DeSmet uses a nonstandard object

Turbo C 1.5 - A Better Contender

As this article went to press, Borland was beta testing version 1.5 of Turbo C. The manual with my advance copy included a 100+ page addendum detailing the improvements and additions.

The compiler and environment are virtually identical to version 1.0's though all known bugs have been fixed. However, they've made several changes to the library.

The new graphics library is VERY similar to the one included with Turbo Pascal v4.0 and includes direct video display functions. It supports a wide range of graphics displays, including the Hercules, VGA, and AT&T 6300.

The graphics initialization function can be told to auto-detect the current display adapter. It then loads the correct graphics driver from disk. I assume Borland will let you distribute the drivers with your programs at no charge (as they did with Turbo Pascal 4.0).

Other additions include:

- A new librarian (TLIB) which accomplishes everything Microsoft's LIB does, and uses a similar (but not identical) syntax.
- A version of the UNIX utility GREP.
- A program to convert the graphics drivers to .OBJ modules, which can then be directly linked into a program.

By the time you read this review, Turbo C 1.5 should be on the market. Borland is talking about a price of \$35.00 for the upgrade (assuming you already own Turbo C).

What does all this mean? I like the Turbo C graphics library better than the one included with the Microsoft compilers; Turbo C supports more graphics adapters, and has auto-sensing.

Does Turbo C 1.5 change my recommendations (see article)? Well, I waffle a bit, but I would say yes. The lack of a debugger is a minus, but Borland is supposed to be working on that. And, of course, the price is right. I don't think you'll be disappointed with Turbo C 1.5.

module format, although a utility is available (at extra cost) which converts its object modules to standard Intel format.

The lightning-fast compiler (easily the fastest in the group) generates quite small executable programs. However, run-times were average (memory allocation was fast but disk I/O was slow).

The standard library adds a few extra functions such as graphics and fast video display routines.

The code generated by DeSmet C88 shows a lack of support for register variables. Neither register variable is stored in a register, and SI and DI are never used in the code. Unlike most of the other compilers, DeSmet does not push the address of an I/O buffer onto the stack as a parameter for putchar().

Computer Innovations C86Plus v1.09g

The now-famous file archive management program ARC (from Systems Enhancements Associates) was written using this compiler.

The installation program is quick and simple. The documentation is very nice, housed in two colorful three-ring binders. These are some of the friendliest manuals I looked into, well-organized and with a touch of humor.

Compile times were abysmally slow, slower than with any other compiler reviewed. Computer Innovations patterned their compiler switches after those used by Microsoft.

C86Plus comes with a linker and librarian, and they've reserved a chapter in the manual for a (future) symbolic debugger. The extensive library includes such functions as serial port I/O and fast screen displays.

Unfortunately, the product did not live up to its documentation. Two of the benchmark programs, FXREF and CRUNCH, would compile but did not execute properly.

There was some problem with recognizing end of file in the gets() function, so FXREF simply read past the end of file into never-never land.

As for CRUNCH, it sent my PC into a hard crash after doing the equivalent of a Control-PrtSc.

Execution times of the other benchmark programs were much slower than average and programs were quite large.

In the code, we see a number of assembler instructions used in the initialization of the putchar() I/O buffers (as in TurboC and MS C). Upon return from the call to putchar(), this compiler uses a pair of POPs to clear parameters

Figure 1 - Results Of C Compiler Tests

results are: os: <obj size in bytes>
 es: <exe size in bytes>
 ct: <comp time in seconds>
 rt: <run time in seconds>

Compiler*		DryStn	Grind	Crunch	FXRef	Display	DiskIO	AllocMem	Loops	Sort
Borland	os:	2,086	2,516	1,084	1,672	320	450	228	225	459
Turbo C	es:	6,720	23,062	20,060	7,520	19,294	7,432	1,886	5,408	1,934
v 1.0D	ct:	9.3	10.9	7.2	10.1	5.3	5.6	3.6	3.6	6.3
(standard)	rt:	64.8	50.5	5.2	41.7	31.3	21.0	15.4	0.6	7.0
Borland	os:	2,035	2,511	1,084	1,640	320	463	230	226	458
Turbo C	es:	6,688	23,062	20,060	7,504	19,294	7,448	1,886	5,408	1,934
v 1.0D	ct:	9.1	10.6	6.7	9.7	4.8	5.1	3.1	3.0	6.1
(optimized)	rt:	63.0	50.5	5.2	41.6	31.3	20.7	15.3	0.5	6.9
C-Ware	os:	2,410	2,919	1,211	1,996	232	407	133	132	452
DeSmet C	es:	9,728	13,824	10,752	11,776	7,168	10,752	2,048	7,168	2,048
v 3.03	ct:	7.2	7.0	4.8	6.6	4.1	4.4	4.0	4.0	4.4
	rt:	87.9	66.4	11.8	41.2	26.9	48.0	8.5	1.8	9.9
Comp. Innov.	os:	2,724	2,882	1,226	2,268	563	569	357	366	739
C86	es:	12,480	35,790	31,658	12,934	29,462	13,246	7,340	10,472	6,522
v 1.09g	ct:	148.5	101.5	42.5	128.9	24.0	32.7	17.9	16.9	41.2
(no opt.)	rt:	100.8	137.5	----	----	37.6	32.5	16.7	2.4	14.6
Comp. Innov.	os:	2,217	2,531	1,258	12,502	563	550	315	319	606
C86	es:	11,984	35,470	31,664	1,824	29,462	13,214	7,292	10,424	6,378
v 1.09g	ct:	147.4	105.0	45.7	130.3	24.3	33.0	17.3	15.7	41.9
(optimized)	rt:	74.5	134.7	----	----	37.4	32.5	14.9	0.4	7.6
Datalight	os:	2,724	3,046	1,263	2,359	293	511	256	236	471
Optimum C	es:	8,170	19,111	18,454	8,110	9,512	7,427	4,060	6,832	4,076
v 3.12	ct:	12.4	11.4	8.4	12.9	6.9	7.3	5.9	5.9	7.0
(standard)	rt:	65.9	64.7	13.7	30.8	23.6	18.9	11.1	1.8	8.8
Datalight	os:	2,474	3,049	1,246	2,414	289	500	206	199	463
Optimum C	es:	7,946	19,079	18,438	8,078	9,512	7,411	3,836	6,816	4,060
v 3.12	ct:	37.1	28.0	15.0	31.5	10.8	13.6	9.8	9.7	17.3
(optimized)	rt:	52.1	64.3	13.6	30.7	23.6	18.2	9.6	0.6	7.5
EcoSoft	os:	2,559	2,983	1,358	2,216	517	642	454	442	728
Eco-C88	es:	10,664	16,404	16,914	11,896	9,334	11,132	2,426	9,256	1,970
v 4.02	ct:	21.5	17.9	11.4	23.4	9.1	10.0	7.4	7.4	9.3
	rt:	81.6	72.9	16.0	43.2	30.4	36.8	17.9	2.9	10.2
Lattice	os:	2,679	2,896	1,396	2,228	507	674	419	405	687
C	es:	9,926	19,760	20,976	10,258	12,480	10,320	3,826	8,368	3,234
v 3.21	ct:	30.2	25.6	15.4	30.1	10.1	12.0	7.8	7.7	12.7
	rt:	86.8	65.2	10.0	49.3	36.4	41.7	15.0	1.1	9.9

from the stack. On an 8088 this takes 24 clock cycles. The traditional method (adding to the stack pointer) takes only 4 cycles.

Datalight Optimum-C v3.11

Optimizing compilers are the current rage, and Datalight had one of the first on the market.

Installation can be done manually or

with a supplied batch file.

The three-ring manual is probably the weakest part of this package. It has numerous spelling and syntactical errors, terse function descriptions, and code printed in a proportional font, making it hard to read.

The package includes an editor, linker, and object-module disassembler and supports third-party debuggers.

You can use a "big" version of the compiler with very large source files. Several environment variables help the compiler locate headers, libraries, and tools.

Optimum C compiles very quickly and produces fast programs. Optimization is done by an optional compiler pass. This slows the compiler immensely but, of course, can be turned off

Compiler*		DryStn	Grind	Crunch	FXRef	Display	DiskIO	AllocMem	Loops	Sort
Manx	os:	2,176	2,701	1,258	1,834	666	374	175	177	448
Aztec C86-c	es:	5,474	10,870	11,174	5,716	7,058	4,966	2,374	4,034	2,178
v 4.10a	ct:	25.4	23.5	11.4	21.4	8.2	8.7	6.4	6.5	9.7
	rt:	67.8	69.8	13.1	45.0	34.5	23.7	10.3	1.2	7.6
Mark Wms	os:	2,559	2,846	1,133	2,130	438	579	316	305	570
Let's C	es:	12,674	47,024	14,864	7,362	11,408	6,722	4,018	5,872	44,224
v 4.0.6	ct:	20.0	21.5	13.4	17.3	7.4	8.0	5.6	5.5	8.0
	rt:	84.3	67.9	12.3	28.1	23.2	23.4	9.1	1.3	7.5
MetaWare	os:	1,905	2,473	1,288	1,859	450	586	320	266	560
High C	es:	24,704	57,824	23,952	21,792	18,160	21,328	5,856	18,064	46,064
v 1.4	ct:	51.5	48.1	39.7	53.9	34.2	34.6	29.6	27.5	31.5
	rt:	63.2	87.1	15.4	33.2	25.2	45.0	----	1.0	9.5
Microsoft	os:	2,507	2,806	1,233	2,088	482	577	317	310	614
C	es:	8,735	26,616	23,584	9,295	21,324	9,081	3,003	7,099	2,581
v 5.00	ct:	29.7	25.0	13.9	26.6	9.3	11.0	7.7	7.5	12.1
(no optim)	rt:	77.7	58.8	7.6	29.0	20.6	18.1	9.7	0.8	8.3
Microsoft	os:	2,174	2,740	894	1,946	451	544	304	285	639
C	es:	8,383	26,504	23,264	9,135	21,308	9,049	2,987	7,083	2,613
v 5.00	ct:	40.0	34.7	14.3	35.3	9.7	12.6	8.3	7.8	17.2
(max opt.)	rt:	56.2	59.2	6.9	28.7	20.6	17.4	8.9	0.2	7.2
Microsoft	os:	3,260	3,859	1,709	2,831	810	945	672	676	1,011
Quick_C	es:	8,815	26,824	23,552	9,359	21,324	9,065	3,003	7,115	2,613
v 1.0	ct:	13.0	12.9	9.9	14.7	8.1	8.2	6.1	6.2	7.1
(no optim)	rt:	80.3	63.2	8.0	29.0	20.7	17.9	9.8	1.3	9.0
Microsoft	os:	2,923	3,527	1,654	2,570	771	870	660	650	947
Quick-C	es:	8,591	26,740	23,612	9,215	21,436	9,049	3,003	7,099	2,549
v 1.0	ct:	22.4	21.6	17.4	26.2	14.2	14.4	10.3	10.3	12.0
(max opt)	rt:	67.6	63.2	8.0	28.7	20.7	17.9	9.3	0.7	7.6

Execution Timings Summary

Compiler*		DryStn	Grind	Crunch	FXRef	Display	DiskIO	AllocMem	Loops	Sort
Turbo C	rt:	63.0	50.5	5.2	41.6	31.3	20.7	15.3	0.5	6.9
DeSmet	rt:	87.9	66.4	11.8	41.2	26.9	48.0	8.5	1.8	9.9
Comp. Innov.	rt:	74.5	134.7	----	----	37.4	32.5	14.9	0.4	7.6
Datalight	rt:	54.4	71.9	13.5	30.8	23.9	19.9	10.0	0.7	8.8
Eco	rt:	81.6	72.9	16.0	43.2	30.4	36.8	17.9	2.9	10.2
Lattice	rt:	86.8	65.2	10.0	49.3	36.4	41.7	15.0	1.1	9.9
Aztec	rt:	67.8	69.8	13.1	45.0	34.5	23.7	10.3	1.2	7.6
Mark Wms	rt:	84.3	67.9	12.3	28.1	23.2	23.4	9.1	1.3	7.5
Metaware	rt:	63.2	87.1	15.4	33.2	25.2	45.0	----	1.0	9.5
Microsoft	rt:	56.2	59.2	6.9	28.7	20.6	17.4	8.9	0.2	7.2
Quick C	rt:	67.6	63.2	8.0	28.7	20.7	17.9	9.3	0.7	7.6

Final Ratings (based on execution speed):

Place	DryStn	Grind	Crunch	FXRef	Display	DiskIO	AllocMem	Loops	Sort
1st	Datalight	Borland	Borland	Mark Wms	Microsoft	Microsoft	DeSmet	Microsoft	Borland
2nd	Microsoft	Microsoft	Microsoft	Microsoft	MS QuickC	MS QuickC	Microsoft	Mark Wms	Microsoft
3rd	Borland	MS QuickC	MS QuickC	MS QuickC	Mark Wms	Datalight	Mark Wms	Borland	Mark Wms
4th	Metaware	Datalight	Lattice	Datalight	Datalight	Borland	MS QuickC	Datalight	Datalight
5th	MS QuickC	Lattice	DeSmet	Metaware	Metaware	Aztec	Datalight	MS QuickC	MS-QuickC
6th	Comp Inn	DeSmet	Mark Wms	DeSmet	DeSmet	Mark Wms	Aztec	Metaware	Aztec
7th	Aztec	Mark Wms	Aztec	Borland	Eco	Comp Inn	Comp Inn	Lattice	Comp Inn
8th	Eco	Aztec	Datalight	Eco	Borland	Eco	Lattice	Aztec	Metaware
9th	Mark Wms	Eco	Eco	Aztec	Aztec	Lattice	Borland	Mark Wms	Lattice*
10th	Lattice	Metaware	Metaware	Lattice	Lattice	Metaware	Eco	DeSmet	DeSmet*
11th	DeSmet	Comp Inn			Comp Inn	DeSmet		Eco	Eco

Figure 2 - Compiler Features

Feature	Borlnd TurboC	CompIn C86	C-Ware DeSmet	Datalt Opt-C	EcoSft EcoC88	Lattice C	Manx AztecC	MrkWms Let'sC	Metwre High-C	Mcrsft C	Mcrsft QuickC
price	\$100	\$497	\$209	\$139	\$140	\$500	\$499	\$75	\$495	\$450	\$99 (11)
utilities:											
linker	Y	Y	Y	Y	N	N	Y	N	N	Y	Y
librarian	N	Y	Y	N	N	Y	Y	Y	N	Y	Y
debugger	N	N	Y	N	Y	(3)	Y	Y	N	Y	Y
editor	Y	N	Y	Y	Y	(3)	Y	Y	N	Y	Y
assembler	N	N	Y	N	N	N	Y	Y	N	(3)	(3)
make	Y	Y	N	Y	(7)	N	Y	Y	N	Y	Y
profiler	N	N	Y	N	N	N	Y	N	N	N	N
.OBJ disasm.	N	N	N	Y	N	Y	N	N	N	N	N
ANSI support:											
void	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
const	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y
volatile	Y	Y	N	Y	Y	Y	Y	N	Y	(10)	(10)
enum	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
prototypes	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y
models (4):											
Tiny (.COM)	Y	N	N	(6)	N	N	N	N	N	N	N
Small	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Medium	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y
Compact	Y	Y	N	Y	Y	Y	Y	N	Y	Y	Y
Large	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
Huge	Y	Y	N	N	N	Y	Y	N	N	Y	Y
Mixed	Y	Y	N	Y	N	Y	Y	N	N	Y	Y
code output:											
.OBJ files	Y	Y	(5)	Y	Y	Y	(5)	Y	Y	Y	Y
80x87 support	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
8087 sensing	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y
186 support	Y	Y	N	N	N	Y	Y	Y	Y	Y	N
286 support(2)	Y	Y	N	N	N	Y	Y	N	Y	Y	Y
386 support(2)	N	Y	N	N	N	N	N	N	(9)	N	N
ROMable	N	(3)	N	(3)	N	N	Y	N	N	Y	N
Assembler	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
source code:											
library	(1)	Y	N	Y	N	(8)	Y	N	N	(1)	(1)
start-up	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
documentation:											
tutorial	Y	N	N	N	N	N	Y	Y	N	Y	Y
lang. ref.	Y	Y	N	N	N	Y	Y	Y	Y	Y	Y
other:											
Req's HD	N	Y	N	N	Y	N	N	N	Y	Y	N

(1) Library source available for \$150.00

(2) None of the compilers reviewed support protected-mode programming -- only support for the special instructions available in real mode on these chips.

(3) Available at extra cost.

(4) Models are as follows:

- Tiny -- code and data both in 64K (.COM model)
- Small -- small (up to 64k) code, small data
- Medium -- large (up to 1Mb) code, small data
- Compact -- small code, large data
- Large -- large code, large data
- Huge -- large code, large data, individual data items can be > 64k

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

(6) .COM programs can be produced by using the EXE2BIN program (provided with most versions of DOS) on some small model programs.

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

(8) Library source available for \$400.00

(9) 80386 protected mode support is available in a separate product

(10) Although the volatile qualifier is accepted by Microsoft compilers, they perform no special processing because of it. It does nothing.

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

during code development.

Optimum C performed very well on most benchmarks, producing the fastest DRYSTN program. I/O operations were quick. Only on those benchmarks featuring floating-point operations did it fall behind (nearly to last place).

Unique among the compilers reviewed, Datalight provides a list of known compiler bugs, with possible work-arounds. No compiler is truly bug-free, and I commend Datalight for their honesty.

You get complete library source code free with the system. The library is complete (to UNIX V standard) but lacks many of the newer functions included in the ANSI draft. It also includes several special functions such as fast video display and mouse interfacing. Since Datalight produces Lattice-compatible object modules, several third-party libraries are available.

Code wise, Datalight does some unique things, like stripping local variables from the stack by ADDING to the stack, rather than using BP to store the stack status. Datalight does automatic register allocation when optimizing. So it assigns the variable *k* to a register (DI) (rather than the inner loop variable *j*). Why the compiler then moves DI to AX, shifts AX for the multiply, and then moves the value back to DI, is a mystery. It would be much faster to simply shift DI.

EcoSoft Eco-C88 v4.02

This inexpensive compiler features excellent compatibility with the emerging ANSI standard.

Although installing the system was relatively easy, defining the proper environment variables was not (due to several spelling and syntax errors in the manual). The manual comes in a thick, three-ring binder. It's well-organized, but it needs proofreading.

This compiler has many interesting traits. For example, its "picky" level tells the compiler how strict it should be when interpreting nonstandard code. It comes with an editor, and a source-level debugger is available at extra cost.

In terms of performance, it didn't do well. Both compile and execution times were the slowest tested. Executable code size was smaller than average, however.

The library contains an acceptable number of functions with both UNIX and ANSI compatibility.

Eco-C88 used a lot of data segments (seven). Otherwise, the code is good, though it doesn't recognize register

variable declarations.

Lattice C v3.21

Lattice was the leader in MS-DOS C compilers during the early 80s. Although it seems popular to knock the package now, it remains competitive. This was the only compiler reviewed which sends both 5.25" and 3.5" disks in the standard package.

Documentation resides in spiral-bound manuals, and it's clear that Lattice does not rely on packaging to sell the compiler. However, I found no glaring errors in the manuals with their detailed and accurate descriptions of utility programs and functions. The disks contained a very long READ.ME file which described new compiler features and otherwise undocumented options.

Lattice throws in both a linker and a librarian. You can purchase a simple source-level debugger, C-Sprite, and an editor separately.

At Comdex, Lattice displayed a fancy, windowed, source-level debugger. They say it will compete directly with Microsoft's Codeview.

Compiles are slow, and execution times are slightly longer than average. Floating point performance is an exception, being very quick. Lattice had the worst display I/O of the group.

Lattice includes a large library which has many useful functions for data conversion and manipulation. There are many third-party libraries available, too. Interestingly, Lattice is the only non-Microsoft compiler in the set which is compatible with the Microsoft Windows Developer's Kit.

In the CODEQUAL test of code generation, Lattice showed it was missing some common optimizations. Although it does use a SHL instead of a multiply in the line "*k* = *k* * 4", it employs the shift using the CX register (after loading CX with 2), rather than the faster pair of single shifts (SHL *x*,1). Like several other compilers, there is a significant amount of initialization code before the putchar() call.

Manx Aztec C Developers Kit v4.10a

Manx produces C compilers for everything from MS-DOS to Amiga to Apple, along with cross-compilers. They even sell older versions of their compilers at reduced prices.

The manual comes in an ill-fitting three-ring binder. The main manual was written for a much older version of the compiler, so they've added a large ad-

dendum. Unfortunately, the addendum cannot be integrated into the manual. So you may find the information you seek in the main manual, the addendum, or both.

A linker, assembler, editor, source-level debugger, and many other tools make this the most complete package of them all. They even include utilities to convert Aztec's nonstandard object modules to Intel format.

Aztec produces the smallest executable files by far. Compile speeds are fast, but program execution is among the slowest.

The code generated by this compiler was very good, with effective use of register variables.

Mark Williams' Let's C v4.0.12

The Mark Williams Company has long been involved in the production of C compilers. With Let's C they compete head-to-head with Borland's Turbo C and Microsoft's QuickC.

Documentation comes in two paperback books, one for the compiler and the other for the csd symbolic debugger. You'll find most of the information in a "lexicon," or encyclopedic, reference section (topics are listed in alphabetical order).

References to utilities and environment variables are mixed with function definitions and glossary information. Although I wasn't fond of this format at first, I seemed to be able to find what I wanted quickly and easily.

This is not an ANSI standard compiler. It adheres to the K&R C very closely but does not include new features such as const or function prototypes.

They've included only a few header files so you often have to add your own function declarations (for such functions as malloc() and strcpy()). The library is small but adequate.

This very complete package, with an editor, assembler, and debugger, even includes full source code for the microEMACS editor. Let's C does rely upon the DOS linker, though.

One especially neat feature is their "shell" program, which lets you edit and compile without going back to MS-DOS. Compilation errors show up when you get back into the editor so you can make quick corrections.

Performance was very good. This compiler turned in the fastest run of FXREF and did better than many compilers advertising "advanced optimization." Code size was small. Due to the

lack of support for function prototypes, FXREF, SORT, and GRIND needed very minor modifications in order to compile.

The first thing which caught my eye in Let's C's code generation was the implementation of the `putchar()` function. It looks as though a near call is done based on an absolute address stored in the data segment. SUBtracts are used to initialize registers to zero—just as fast as the more commonly used XORs, and much faster than the often used direct MOVs.

Metaware High C v1.4

This compiler felt massive when I used it. For example, the compiler itself takes up a single 600+K executable file which uses dynamic overlays. High C has been used in the past to create some well-known programs, such as dBASE III+.

Installation was automated but *very* slow. It's a disk hog, requiring more than three megabytes if you install the complete system. Needless to say, it can't run on floppies.

The documentation suffers from a too-small binder; pages kept popping out. Organization of this two-inch thick manual, however, is fairly good.

A special "reviewers" section describes how to disable features, such as stack-checking, to improve compiler performance on benchmarks. There are also several unique features to this compiler, such as its ability to generate code for the NEC V-series of microprocessors. (I didn't use this feature for these benchmarks.)

You get none of the extras such as linkers and debuggers. However, the compiler does support Microsoft's Codeview, and it's the only non-Microsoft compiler in the review to do so. The somewhat sparse library has very few nonstandard bell-and-whistle functions. It is very ANSI compatible.

Performance was mediocre on most tests and compilation speeds were *very* slow. Floating point and file test were the slowest in the review suite. According to Metaware, the file speeds are affected by the use of small (512 byte) buffers.

The compiler did do quite well on the DRYSTN and DISPLAY tests but the ALLOCMEM benchmark generated an infinite loop, locking up the PC.

High C ignored register variable declarations. This is yet another compiler which generates a significant chunk of code to initialize buffers before a call to

`putchar()`.

Microsoft Optimizing C v5.00

Microsoft C has been one of the leading compilers for some time and is considered by many to be the "best" MS-DOS C compiler.

The automatic installation program is well-documented and very flexible.

Environmental variables help the compiler find its components and libraries (and one variable even stores default compiler switches).

This compiler uses over two megabytes of hard drive when fully installed, but it can also be installed for floppies. (I wouldn't recommend it if you don't enjoy disk swapping exercises.)

Microsoft distributes generous documentation. You get five manuals, four in three-ring binders, and one paperback. They're well-organized, with classy print and many examples. Of the group, this is the most professionally documented.

This package includes the Codeview symbolic source-level debugger, LINK, LIB, and several miscellaneous utilities. Codeview is currently the premiere MS-DOS debugger, although that may change as other companies (such as Lattice and Borland) create their own.

The code was outstanding, better than the rest of the group on most benchmarks. It never came in lower than second place in executable code speed. Compile times were slow, but Quick C (see below), included in the package, compiles very quickly.

Microsoft does not implement some of the features found in the proposed ANSI standard. For example, although it recognizes the keyword "volatile," the compiler does not change its output to support the term. The ANSI standard pre-defined macros, such as `__TIME__` and `__DATE__` are also not supported.

For all of its touted "optimizations", Microsoft still doesn't know that using two one-bit shifts is faster than doing one shift using CX to indicate the shift size.

It seems to have optimized the multiply in the line "`i = i + i * j`" to a series of ADDs. The compiler recognized that in each loop, `i` increases by `i`.

Microsoft Quick C v1.00

You can get Quick C two ways—as part of Microsoft's Optimizing C package—or by itself.

Quick C is Microsoft's answer to competitors like Borland. It's an inex-

pensive, full-featured compiler in an integrated environment. The environment includes the compiler, an editor, and a subset of the CodeView debugger. In the environment, it compiles only a medium (small data, large code) program, but the command line version supports all five standard Microsoft models.

Unlike its large sibling, Quick C is comfortable on a floppy-based PC. This compiler does not use a great deal of disk space, but the integrated version consumes memory at an alarming rate. The command line version uses a subset of the command line switches from its bigger brother, and operates very much like a traditional compiler.

The package contains three manuals, all in trade-paperback format. The Quick C Programmer's Guide felt somewhat disorganized. However, the C tutorial got me off to a good start. It isn't detailed enough, though, for those of you who want to really sink your teeth in.

Microsoft has made some odd choices with this compiler. First of all, it does not use the standard .LIB library file format; rather, it uses a "Quick Library." The Quick Library provided contains most of the "standard" C functions.

Many programs I have would not compile in the integrated environment if I didn't first create a "make" file. In that file I had to include the names of the function libraries. (Interestingly, the command line version uses the same models and libraries as the MS-C v5.0.)

The limited debugger gives you no tracepoints and only a limited number of watch variables. The documentation for the debugger was obscure, although as a user of CodeView I had no major difficulties.

Code performance was very good. Compile speeds were also very good, although not anywhere near the fastest. Execution times always came in slightly slower than MS-C v5.0. According to Microsoft, Quick C uses the MS-C v4.0 optimizer. This approach makes sense, since I'm sure Microsoft does not want Quick C to compete with MS-C v5.0.

Quick C produced code much like most of the other compilers. It did use two one-bit shifts rather than the CL-based shift, an optimization its larger brother missed.

Conclusions

The quality of MS-DOS C compilers has certainly improved and, if anything,

I was surprised at the general quality of all these compilers. (Though documentation hasn't always kept up.)

It all comes down to this—which compiler should you buy? The answer depends on what you want to do and how much you want to spend.

Cheap Recommendations

In the low-cost arena (under \$200), I like Mark Williams, Borland, and Datalight. I'm also impressed with Let's C, although its lack of many proposed ANSI features is a minus.

The documentation for the Datalight compiler needs a revision, but the product produces excellent code for a paltry \$129.

Optimum-C includes source for its library. A big plus.

Turbo C has a slick environment and a large, full-featured library.

I'd recommend Optimum-C to professionals who need Lattice compatibility, and either Mark Williams or Turbo C to beginning C'ers.

Although not as easy to work with as Turbo C, Quick C also deserves honorable mention. (Its claim to consideration is its debugger, a requirement for developing large systems.)

Not So Cheap Recommendations

On the high end, above \$200, I would choose the Microsoft C v5.0 for its excellent documentation, CodeView debugger, and extensive library. It includes Quick C, so you're getting a fast compiler along with an optimizing one.

If you're working on machines other than PCs or clones, Lattice and Aztec would also be good choices.

I hope this review has helped you in making a decision. By the time this sees print, there may already be new versions of some of these compilers. All this competition should keep newer and better products coming.

Editor's note: Scott worked on this article for nearly three months. During that time, most of the packages he was reviewing went through at least one (and sometimes two or three) revision. Even a week after article deadline, he was still receiving (and including) new versions. If this review isn't current, you can't blame Scott.

Meanwhile, fire up your compilers and enjoy. Those of you struggling along without a debugger should check out the Turbo C debugger article in this issue. So your C'ing will be that much more fun (without making a spectacle of yourself)

Companies

Borland International
4585 Scotts Valley Drive
Scotts Valley, CA 95066
(800) 255-8008
(800) 742-1133 (In California)

Computer Innovations
980 Shrewsbury Avenue
Tinton Falls, NJ 07724
(800) 922-0169

C Ware Corporation
P.O. Box 428
Paso Robles, CA 93447
(805) 239-4620

Datalight
17505 68th NE Ste. 304
Bothell, WA 98011
(800) 221-6630

Ecosoft Incorporated
6413 N. College Avenue
Indianapolis, IN 46220
(800) 952-0472

Lattice Incorporated
2500 S. Highland
Lombard, IL 60148
(800) 533-3577
(312) 916-1600 (In Illinois)

Manx Software Systems
P.O. Box 55
Shrewsbury, NJ 07701
(800) 221-0440
(201) 542-2121 (In New Jersey)

Mark Williams Company
1430 Wrightwood Avenue
Chicago, IL 60614
(800) 692-1700
(312) 472-6659 (In Illinois)

MetaWare Incorporated
903 Pacific Ave. Ste. 201
Santa Cruz, CA 95060
(408) 429-6382

Microsoft
16011 NE 36th Way
Redmond, WA 98052
(800) 426-9400
(206) 882-8088 (In Washington)

Got Software Problems?

Get Turbo GhostWriter

Now \$99⁰⁰

	Full Version	Starter Version	Cost of Upgrade for \$99 version
B-Tree File Manager	Yes	Yes	N/A
Context-Sensitive Help	Yes	Yes	50
Command Window	Yes	Yes	50
Relational Model	Yes	Yes	150
Free Support	No Limit	1 Hour	100
Manual	Cloth D-ring	Paperback	50
All Upgrades	225

Orders & Information 800 227-7681
30 day money-back guarantee (less \$14 s/h)

YES! Send me _____ copies of Turbo GhostWriter today.

Name _____
Address _____
City, State, Zip _____
Phone _____
MC, Visa, or Choice Card _____
#(or write COD) _____
Expiration Date _____
Circle version \$99 \$289
ASCII - 3239 Mill Run - Raleigh, NC 27612
800 227-7681

EVERYTIME THE PHONE RINGS IT'S A CUSTOMER WITH A BUG!

NOVICE PROGRAMMERS!!! ... CAN'T SLEEP AT NIGHT!

I CAN'T PRODUCE CUSTOM SOFTWARE COMPETITIVELY!

Turbo GhostWriter is an application generator that creates 80% of your custom application automatically -- as a Turbo Pascal Program. That 80% is done swiftly in just 10% of the time it takes to code it yourself. That 80% contains all of the "hooks" to add custom features such as table verification, security, currency conversion, importing files from other languages, etc...

Now it's up to you to write the code to relate one file to another. The relational model shows you how and it only takes a few minutes. End to end, all programming for 5 relational files should take less than 30 minutes. It's far superior to doing the whole thing from scratch. Your user sees a consistent interface, all programs function the same way, and your code is 100% error-free Turbo Pascal.

ASCII wants to give you a chance to try Turbo GhostWriter. For a limited time you can buy a fully functional version for just \$99. This version will create flat files such as mailing lists, telephone files, inventory files, simple payroll files, etc. The utilities are the same as for the full-blown product, but are modestly packaged, and support is limited. It's the perfect way to see if an application generator is for you without risking a cent.

Reader Service Number 48

Figure 3 - CodeQual Program (Code Quality)

```

/*
Program: CodeQual (Code Quality)
Version: 1.00 Date: September 22, 1987
Language: Generic ANSI C

This program should be disassembled after a
compile to show how "intelligently" the
compiler generates object code.
Copyright 1987 Scott Robert Ladd.
Source code released into the public domain.*/

```

```

#include "stdio.h"

main() /* line 18 */
{ /* line 19 */
    register int i, j; /* line 20 */
    int k = 2048, l = 0; /* line 21 */
    /* line 22 */
    for (i = 0; i < 2; ++i) /* line 23 */
    { /* line 24 */
        for (j = 0; j < 3; ++j) /* line 25 */
        { /* line 26 */
            putchar('.'); /* line 27 */
            k = k * 4; /* line 28 */
            l = l + i * j; /* line 29 */
            /* line 30 */
        } /* line 31 */
    } /* line 32 */
}

```

BORLAND TURBO C

```

_text segment byte public 'code'
dgroup group data, bss
assume cs:_text, ds:dgroup, ss:dgroup
_text ends
_data segment word public 'data'
_d@ label byte
_data ends
_bss segment word public 'bss'
_b@ label byte
_bss ends
_text segment byte public 'code'
_main near ;line 19
push si
push di
push bp
mov bp, sp
sub sp, 4
mov word ptr [bp-4], 2048 ;line 21
mov word ptr [bp-2], 0
xor di, di ;line 23
jmp short @5
@4: xor si, si ;line 25
jmp short @9
@8: ;line 27
mov bx, offset dgroup: __streams+14
inc word ptr [bx]
jge @11
mov al, 46
mov bx, offset dgroup: __streams+24
inc word ptr [bx]
mov bx, word ptr [bx]
mov byte ptr [bx-1], al
mov ah, 0
jmp short @10
@11: mov ax, offset dgroup: __streams+14
push ax
mov al, 46
push ax
call near ptr __fputc
add sp, 4
@10: mov ax, word ptr [bp-4] ;line 28
shl ax, 1
shl ax, 1
mov word ptr [bp-4], ax
mov ax, di ;line 29
mul si
add ax, word ptr [bp-2]
mov word ptr [bp-2], ax
inc si ;line 30
@9: cmp si, 3
jl @8
inc di ;line 31
@5: cmp di, 2

```

```

jl @4
mov sp, bp ;line 32
pop bp
pop di
pop si
ret
_main endp
_text ends
_data segment word public 'data'
_s@ label byte
_data ends
extrn __streams:word
_text segment byte public 'code'
extrn __fputc:near
public _main
_text ends
end

```

C-WARE DESMET C

```

CSEG
PUBLIC main
PUBLIC putchar
main: PUSH BP ;line 19
MOV BP, SP
SUB SP, 8
MOV WORD [BP-6], 2048 ;line 21
MOV WORD [BP-8], 0
MOV WORD [BP-2], 0 ;line 23
_L1: CMP WORD [BP-2], 2
JGE _L2
MOV WORD [BP-4], 0 ;line 25
_L4: CMP WORD [BP-4], 3
JGE _L5
MOV AX, 46 ;line 27
PUSH AX
CALL putchar
ADD SP, 2
MOV AX, WORD [BP-6] ;line 28
SHL AX, 1
SHL AX, 1
MOV WORD [BP-6], AX
MOV AX, WORD [BP-2] ;line 29
IMUL WORD [BP-4]
ADD AX, WORD [BP-8]
MOV WORD [BP-8], AX
_L6: INC WORD [BP-4] ;line 30
JMP _L4
_L5: INC WORD [BP-2]
_L3: JMP _L1 ;line 31
_L2: MOV SP, BP ;line 32
POP BP
RET
END

```

COMPUTER INNOVATIONS C86

```

.8086
extrn __fputc:near
extrn __iob:byte
dgroup group CONST, DATA, BSS
_TEXT segment byte public 'CODE'
_TEXT ends
CONST segment word public 'CONST'
assume ds:dgroup
CONST ends
_DATA segment word public 'DATA'
assume ds:dgroup
_DATA ends
_BSS segment word public 'BSS'
assume ds:dgroup
_BSS ends
assume cs:_TEXT, ds:dgroup
_TEXT segment byte public 'CODE'
extrn __acrtused:near ;line 19
_public _main
_main proc near
push bp
mov bp, sp
sub sp, 4
push si
push di
mov word ptr _k[bp], 2048 ;line 21
mov word ptr _l[bp], 0

```



```

xor     si,si
L3@2:  jmp     short L12@2           ;line 23
xor     di,di                   ;line 25
jmp     short L11@2
L0@2:  dec     word ptr dgroup: __iob+10;line 27
mov     ax,word ptr dgroup: __iob+10
or      ax,ax
jl      L6@2
mov     bx,word ptr dgroup: __iob+8
inc     word ptr dgroup: __iob+8
mov     al,+46
mov     mov     byte ptr [bx],al
xor     ah,ah
jmp     short L10@2
L6@2:  mov     ax,offset dgroup: __iob+8
push   ax
mov     bx,+46
push   bx
call   near ptr __flsbuf
pop     bx
pop     bx
L10@2: mov     ax,word ptr _k@1[bp]
sal     ax,1
sal     ax,1
mov     cx,ax                   ;line 28
mov     ax,si
mov     word ptr _i@1[bp],ax
mov     word ptr _j@1[bp],di
imul   di
add     word ptr _l@1[bp],ax    ;line 29
mov     si,word ptr _i@1[bp]   ;line 30
mov     word ptr _k@1[bp],cx
L1@2:  inc     di
L11@2: cmp     di,+3
jl      L0@2
L2@2:
L4@2:  inc     si                   ;line 31
L12@2: cmp     si,+2
jl      L3@2
L5@2:
L13@2: pop     di                   ;line 32
pop     si
mov     sp,bp
pop     bp
ret
_main  endp
_TEXT  ends
end

```

DATALIGHT OPTIMUM-C

```

public main
extrn fputc
extrn _iob
main:  push   BP                   ;line 19
sub    SP,8
mov    BP,SP
mov    4[BP],0800h             ;line 21
xor    AX,AX
mov    6[BP],AX
mov    0[BP],AX               ;line 23
L13:  mov    2[BP],0             ;line 25
L18:  mov    AX,offset _iob[0Eh] ;line 27
push   AX
mov    AX,02Eh
push   AX
calln  fputc
mov    SP,BP
mov    AX,4[BP]               ;line 28
shl   AX,1
shl   AX,1
mov    4[BP],AX
mov    AX,0[BP]               ;line 29
imul  2[BP]
add   AX,6[BP]
mov   6[BP],AX
inc   2[BP]                   ;line 31
cmp   2[BP],3
jl    L18
inc   0[BP]                   ;line 32
cmp   0[BP],2
jl    L13
add   SP,8
pop   BP
ret

```

ECOSOFT ECO-C88

```

DGROUP group  $c$strtseg,$d$dataseg,$e$usdseg
DGROUP group  $f$udseg,$g$uedseg
DGROUP group  $h$stkseg,$i$endseg
PGROUP group  $a$chain,$b$prog
$a$chain      segment public 'code'
$a$chain      ends
$b$prog       segment word public 'code'
$b$prog       ends
$c$strtseg    segment public 'data1'
$c$strtseg    ends
$d$dataseg    segment word public 'data2'
$d$dataseg    ends
$e$usdseg     segment word public 'data3'
$e$usdseg     ends
$f$udseg      segment word public 'data4'
$f$udseg      ends
$g$uedseg     segment word public 'data5'
$g$uedseg     ends
$h$stkseg     segment word stack 'data6'
$h$stkseg     ends
$i$endseg     segment word public 'data7'
$i$endseg     ends
assume cs:PGROUP,ds:DGROUP
assume es:DGROUP,ss:DGROUP
extrn _putchar:near
extrn $start:near
$b$prog       segment 'code'
$b$prog       ends
$d$dataseg    segment 'data2'
$d$dataseg    ends
$b$prog       segment 'code'

```

```

_main public main
proc near ;line 19
push bp
mov bp,sp
add sp,-8
mov word ptr [bp][-6],2048 ;line 21
mov word ptr [bp][-8],0
mov word ptr [bp][-2],0 ;line 23
L72e label near
cmp word ptr [bp][-2],2
jge L72d
mov word ptr [bp][-4],0 ;line 25
L736 label near
cmp word ptr [bp][-4],3
jge L735
mov dx,46 ;line 27
push dx
call _putchar
add sp,2
mov ax,word ptr [bp][-6] ;line 28
shl ax,1
shl ax,1
mov word ptr [bp][-6],ax ;line 29
mov ax,word ptr [bp][-4]
imul word ptr [bp][-2]
add ax,word ptr [bp][-8]
mov word ptr [bp][-8],ax
L737 label near
inc word ptr [bp][-4]
jmp short L736
L735 label near ;line 30
L72f label near
inc word ptr [bp][-2]
jmp short L72e
L72d label near ;line 31
mov ax,0 ;line 32
mov sp,bp
pop bp
ret
_main endp
$b$prog ends
end

```

LATTICE C

```

MOV AX,FFF6 ;line 19
CALL CXSTKP
MOV W,[BP+06],0800 ;line 21
XOR AX,AX
MOV [BP+02],AX
MOV [BP+08],AX ;line 23
CMP W,[BP+02],02

```

```

JGE 0064
MOV W,[BP+04],0000 ;line 25
CMP W,[BP+04],03
JGE 005F
DEC W,[_IOB+0012] ;line 27
MOV AX,[_IOB+0012]
TEST AX,AX
JS 003F
MOV SI,[_IOB+000E]
INC W,[_IOB+000E]
MOV AL,2E
MOV [SI],AL
XOR AH,AH
JMP 004C
MOV AX,_IOB+000E
PUSH AX
MOV AX,002E
PUSH AX
CALL _FLSBF
MOV SP,BP
MOV CL,02 ;line 28
SHL W,[BP+06],CL
MOV AX,[BP+02] ;line 29
IMUL W,[BP+04]
ADD [BP+08],AX
INC W,[BP+04] ;line 30
JMP 001E
INC W,[BP+02] ;line 31
JMP 0013
ADD SP,0A ;line 32
POP BP
RET

```

MANX AZTEC C

```

codeseg segment para public 'code'
databseg segment para public 'data'
databseg ends
assume cs:codeseg,ds:databseg
assume es:databseg,ss:databseg
extrn $begin:near
$3 public main_ ;line 19
main_proc near
push bp
mov bp,sp
ifdef $20001
add sp,$20001
endif
ifdef $20002
push di
endif
ifdef $20003
push si
endif
mov word ptr -2[bp],0800H ;line 20
mov word ptr -4[bp],00H ;line 23
$6: mov di,00H ;line 25
$9: mov ax,offset Cbufs_+13 ;line 27
push ax
mov ax,02eH
push ax
call near ptr aputc_
add sp,4
mov ax,word ptr -2[bp] ;line 28
shl ax,1
shl ax,1
mov word ptr -2[bp],ax
mov ax,si ;line 29
imul di
add word ptr -4[bp],ax
$7: inc di ;line 30
cmp di,03H
blt $9
$8:
$4: inc si ;line 31
cmp si,02H
blt $6
$5:
$20004:
ifdef $20003 ;line 32
pop si
endif
ifdef $20002
pop di
endif

```

```

mov sp,bp
pop bp
ret
$20001 = -6
$20003 equ 1
$20002 equ 1
main_endp
$3: extrn aputc_:near
codeseg ends
databseg segment word public 'data'
extrn Cbufs_:word
databseg ends
end

```

MARK WILLIAMS LET'S C

```

.shri
.globl main_
main_ : push si ;line 19
push di
push bp
mov bp,sp
sub sp,$4
mov -2(bp),$2048 ;line 21
mov -4(bp),$0
sub si,si ;line 23
jmp L4
L3: inc si
L4: cmp si,$2
jge L1
sub di,di ;line 25
jmp L7
L6: inc di
L7: cmp di,$3
jge L3
mov ax,$_stdout_ ;line 27
push ax
mov ax,$46
push ax
icall _stdout_+10
add sp,$4
mov ax,-2(bp) ;line 28
sal ax,$1
sal ax,$1
mov -2(bp),ax
mov ax,di ;line 29
imul si
add ax,-4(bp)
mov -4(bp),ax ;line 30
L1: mov sp,bp ;line 32
pop bp
pop di
pop si
ret

```

METAWARE HIGH C

```

extrn _mwininit,_iob,fputc
MODEL segment 'DATA'
db 83
MODEL ends
CODEQUAL segment 'CODE'
.L0000: public main ;line 19
main proc near
push bp
mov bp,sp
sub sp,8
mov word ptr -6[bp],2048 ;line 21
sub ax,ax
mov -8[bp],ax ;line 23
mov -2[bp],ax
.L0013: cmp word ptr -2[bp],2
jnl 0067
mov word ptr -4[bp],0 ;line 25
.L001e: cmp word ptr -4[bp],3
jnl 0062
mov ax,_iob+14 ;line 27
and ax,ax
jle 003c
dec word ptr _iob+14
mov si,_iob+12
inc word ptr _iob+12
mov byte ptr [si],46

```



```

    jmp     004a
.L003c: mov     ax,offset _iob+12
    push   ax
    mov     ax,46
    push   ax
    call   fputc
    add     sp,4
.L004a: mov     ax,-6[bp]           ;line 28
    shl    ax,1
    shl    ax,1
    mov     -6[bp],ax
    mov     ax,-2[bp]           ;line 29
    imul   word ptr -4[bp]
    add     -8[bp],ax
    inc    word ptr -4[bp]
    jmp     001e
.L0062: inc    word ptr -2[bp]
    jmp     0013
.L0067: mov     sp,bp           ;line 32
    pop    bp
    rets
main    endp
CODEQUAL ends

```

MICROSOFT C v5.0

```

    .8087
_TEXT  SEGMENT WORD PUBLIC 'CODE'
_TEXT  ENDS
_DATA  SEGMENT WORD PUBLIC 'DATA'
_DATA  ENDS
CONST  SEGMENT WORD PUBLIC 'CONST'
CONST  ENDS
_BSS   SEGMENT WORD PUBLIC 'BSS'
_BSS   ENDS
DGROUP GROUP CONST, _BSS, _DATA
ASSUME CS: TEXT, DS:DGROUP, SS:DGROUP
EXTRN  __acrtused:ABS
EXTRN  __flsbuf:NEAR
EXTRN  __iob:BYTE
_TEXT  SEGMENT
ASSUME CS: TEXT
PUBLIC _main           ;line 19
_main  PROC NEAR
    push   bp
    mov     bp,sp
    sub     sp,12
    push   di
    push   si
    mov     WORD PTR [bp-6],2048 ;line 20
    mov     WORD PTR [bp-8],0
    sub     si,si           ;line 23
    sub     di,di           ;line 25
    mov     WORD PTR [bp-10],di
    mov     WORD PTR [bp-12],3
    add     di,3
    $L20002:dec WORD PTR __iob+10 ;line 27
    js     $L20000
    mov     al,46
    mov     bx,WORD PTR __iob+8
    inc    WORD PTR __iob+8
    mov     BYTE PTR [bx],al
    jmp     SHORT $L20001
    nop
    $L20000:mov ax,OFFSET __iob+8
    push   ax
    mov     ax,46
    push   ax
    call   __flsbuf
    add     sp,4
    $L20001:mov cl,2           ;line 28
    shl    WORD PTR [bp-6],cl ;k
    mov     ax,WORD PTR [bp-10] ;line 29
    add     WORD PTR [bp-8],ax ;l
    add     WORD PTR [bp-10],si ;line 30
    dec    WORD PTR [bp-12]
    jne    $L20002
    inc    si           ;line 31
    cmp    si,2
    jl     $L20003
    pop    si           ;line 32
    pop    di
    mov     sp,bp
    pop    bp
    ret

```

```

_main  ENDP
_TEXT  ENDS
END

MICROSOFT QUICK C

    extrn  __acrtused
    extrn  __iob
    extrn  __flsbuf
public  _main
_main:  push   BP           ;line 19
    mov    BP,SP
    sub    SP,8
    push   DI
    push   SI
    mov    OFFFAh[BP],0800h ;line 21
    mov    OFFF8h[BP],0
    mov    SI,0           ;line 23
    jmp    L19
L19:    nop
L1A:    mov    DI,0           ;line 25
    jmp    L20
L20:    dec    __iob[0Ah] ;line 27
    mov    AX,__iob[0Ah]
    and    AX,AX
    jl     L2C
L2C:    mov    BX,__iob[08h]
    inc    __iob[08h]
    mov    [BX],02Eh
    mov    AL,02Eh
    and    AX,0FFh
    jmps   L3F
L3F:    nop
    mov    AX,offset __iob[8]
    push   AX
    mov    AX,02Eh
    push   AX
    calln  __flsbuf
    add    SP,4
    mov    AX,OFFFAh[BP] ;line 28
    shl    AX,1
    shl    AX,1
    mov    OFFFAh[BP],AX ;line 29
    mov    AX,SI
    imul   DI
    add    AX,OFFF8h[BP]
    mov    OFFF8h[BP],AX ;line 30
    inc    DI
    cmp    DI,3
    jge    L6B
    jmp    L20
L6B:    inc    SI           ;line 31
    cmp    SI,2
    jge    L74
    jmp    L1A
L74:    pop    SI           ;line 32
    pop    DI
    mov    SP,BP
    pop    BP
    ret

```

End of Figure 3

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

Redh

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

Programming For Change:

Using Data Files In Lieu Of Program Data

You've always wanted to write a parser, right? A simple but useful parser. (Of course, that means saving the certified ADA parser for our second project.)

Anyway, it'd be nice to write something that would accept a text file and do something with it. Understand it. Act on it. Work a little magic.

I guess I'm pretty lucky; my family has progressed from tolerating my computer activities to participating in them. My wife does most of her reports and my daughter does her homework on my system. That's the good part.

On the down side, I really don't want them playing around with the stuff I'm working on, even accidentally. So I created a directory for each of them to use. That's where the problem begins.

The Problem

My hard disk has a directory structure that's confusing to anyone, including myself. One of these days, of course, I'm going to get organized. But in the meantime, I shouldn't be too surprised that neither wife nor daughter are excited about learning how neat the MS-DOS hierarchical directory structure is.

We programmers, in general, assume it's obvious to everyone (including non-programmers) how to move between directories on a hard disk. Not so. Although learning to run a word processor was easy enough for my family, getting to the correct directory was a boggle, confusing. I kept telling them how easy it was; all they had to remember was to type:

```
D:
cd\family\words\name
```

where name is either Karol or Katie.

In their defense, I'll admit there is an incredible number of combinations in the above sequence. What the problem called for was a programming solution: a tool to simplify moving among directories.

The more I thought about it, the more I was convinced I wouldn't mind an easier way to move between directories and subdirectories myself. Maybe I could get organized.

It's not uncommon for me to have a pathname something like:

```
C:\eco88\tools\source\project1
```

I know I'm working on project1, but I'm not always sure which subdirectory holds it. Also, it's a hassle to type in the pathname for project1. Yet, I don't want to move the project from its (logical?) place.

Designing A Solution

When I first started thinking about how to address the problem, several solutions seemed feasible. The most direct is a batch file for each user that contains the sequence of MS-DOS commands that would place the user in the proper directory. That's the easiest solution.

It would solve my family's immediate problem, but we'd need a batch file for each directory pathname we wanted to use. Again some form of program seemed to be the answer.

First, what information does MS-DOS need to change directories?

MS-DOS needs to know only two things: 1) the full pathname, and 2) possibly a drive name. Given what I wanted to do, I needed a third piece of information, an abbreviation for the full pathname, as well. For the pathname mentioned earlier, the information might be:

```
(1)\eco88\tools\source\project1
```

```
(2)C:
(3)project1
```

Therefore, if I type in "project1" from any directory on any drive in my system, I should end up in the directory C:\eco88\tools\source\project1.

Second, how can I access the program from anywhere on the disk?

Most programmers are familiar with the "environment" area available in MS-DOS. It's a small area of memory that MS-DOS can examine for information to be used with MS-DOS commands. Of interest to us is the PATH command which tells MS-DOS which pathnames to search if a file isn't in the current working directory.

The PATH environment variable lets us make a single copy of a program accessible from any directory on the disk. This saves disk space.

At present, my path is set to:

```
E:;C:\bin;C:\;
```

which says: Search drive "E:" (a RAM disk) first, and if you don't find the file there, try "C:\bin". If that fails, try "C:\". If all three path names fail, MS-DOS gives up and issues an error message. Notice that each path name in the PATH environment area is separated by a semicolon.

To set the PATH environment variable, all you do is type in:

```
C>PATH=E:;C:\bin;C:\;
```

If you type:

```
C>set
```

and press Enter, you'll see the status of your environment variables. The use of the PATH environment variable solves one of our design problems: accessing the program from any place in the sys-

tem. I use the "bin" directory off the root directory to hold all of my executable files, so that's where I'll store the directory program. (When I finish writing it.)

Data Alternatives

Next, how do I want to access the information needed for each directory entry?

My first thought was to code the directory paths as a series of pointers to chars, something like:

```
char *paths[] = {
    'Karol D: \family\words\karol'',
    'Katie D: \family\words\katie'',
    'project1 C:
    \eco88\tools\source\project1'',
    0
};
```

This data structure would work okay, but I'd have to recompile the program each time I want to add to, or otherwise change, the abbreviations for a directory path. (The last entry is zero so I would be able to sense the end of the list easily.)

By now, the final solution should be obvious. Because we need a flexible means of storing directory path names, we'll use a text file.

Now that we know: 1) the data requirements for the program, 2) how to access that data from any point on the disk, and 3) how the data will be stored, writing the program is fairly easy.

The Solution

The program in Figure 1 is the solution to the problem. I wrote it with the Eco-C88 Rel. 4.0 C compiler, but it should be compatible with almost any MS-DOS C compiler. I've used prototyping in the program, but you can easily change the prototypes to the standard K&R format.

The program begins with header files and definitions. Note that "get.dat" is the data file.

Finding The Data

Once in main(), the program makes sure there are two arguments on the command line. Therefore, if I want to go to my test project, I would enter:

```
C>get project1
```

If I'd just entered "get" the program would quit with an error message. Note: the program assumes that MS-DOS's PATH environment variable points to the program (get.exe) and the data file (get.dat).

The call to open_data_file() does more than simply open the data file. As I mentioned above, the PATH variable can have multiple paths. Therefore, it is necessary to read, or parse, the PATH environment data to find which path holds the data file we need.

Inside open_data_file(), the call to getenv() returns a pointer to the PATH environment variable. If PATH is not set, a NULL pointer returns and the program ends. Assuming that PATH has been set, the pointer will be non-NULL. On my system, ptr would point to:

```
E::C:\bin;C:;
```

The problem now is to separate the paths to find which one holds our data file. To do this, we copy the PATH information from ptr into the p[] character array by a call to strcpy().

Next, we use strtok() to break the PATH string into its sub-parts (watching for the semicolon delimiter). We won't use strtok() that often, but it's wonderful when we need it. It's also part of the (proposed) ANSI and UNIX System V standard libraries.

In short, strtok() searches a string (the first argument of the function) using a list of one or more characters from the second string. Our first call to strtok() searches p[] using the semicolon as a delimiter. If strtok() finds a semicolon, it forms a string of what was read up to that point and returns a pointer to it.

If a non-null pointer is returned from strtok(), it will have returned a pointer to a string. In our case, it would be:

```
ptr = 'E:'
```

This is copied into temp[] by a call to strcpy() and a backslash is then appended to it. The second call to strcat() adds the data file name to the string with the result:

```
temp[] = E:\get.dat
```

We then try to open a file using the full pathname just built. If we're in the wrong directory for the get.dat data file, a NULL pointer is returned. Since we may still have more PATH directories to examine, we clear out temp[] and call strtok() again.

Because the first argument to strtok() is cast to a NULL character pointer, strtok() continues to parse the previous string. If you follow through the second pass, temp[] will hold:

```
temp[] = C:\bin\get.dat
```

and the third pass would be:

```
temp[] = C:\get.dat
```

Assuming that one of the PATH strings does hold the get.dat data file, FILE pointer fpin will be non-NULL and we break out of the while loop. (If we exhaust the PATH possibilities,

strtok() will return a NULL pointer and the while loop ends. The next if test would find a NULL in fpin and the program would end.)

Assuming open_data_file() returns a valid FILE pointer, the call to search_file() passes the FILE pointer and the command line abbreviation for the directory path wanted by the user (e.g., "project1"). The search_file() function reads the get.dat data file one line at a time, using the newline character ('\n') to sense the end of each directory name. The program assumes the data is stored in the form:

```
abbreviation drive-name
directory-pathname
```

or

```
'project1 C:
\eco88\tools\source\project1'',
```

with one blank space between each of the three data items. Note that we use strtok() again to parse the string read from the data file, using a blank space as the delimiter this time. After the first call, ptr would point to "project1".

In the if statement, strcmp() checks ptr against "where" for a match. Since "where" is the command line abbreviation for the directory path wanted by the user, we're checking to see if the abbreviation exists in the get.dat data file. If we find a match, further calls to strtok() break the string up into the drive (drive[]) and path (path[]) information and return a MATCH flag. If no match is found, the FAIL flag gets set.

If MATCH is found, we call do_switch(). The system() function performs a command.com-type directive to MS-DOS. Therefore, if drive[] contains "D:", it's as if that were typed in at the keyboard, thus moving control to drive D.

The chdir() function changes the current working directory to the one specified by the argument passed to chdir(). Since path[] has the full path name for our desired directory, chdir() puts us in that directory and returns control to main(). If chdir() doesn't find the directory, it displays an error message. A successful chdir() ends the program and you're in the desired directory.

Parsing Thoughts

Although the program itself is useful, the idea of replacing program data with a data file is particularly interest-

ing. Not too long ago, I wanted to write a CAI (computer aided instruction) on the C language.

I started out writing printf()'s but Tim Leslie, a colleague and friend, convinced me that a better approach would be to write a Pilot-like interpreter instead.

He was right.

While the initial costs were higher (it took almost a month to write the interpreter), the result is a program that lets me write and test new CAI's in weeks rather than months.

The interpreter is very similar to what I've presented here; parsing program lines, finding out what was read in the parse, and then acting on it. (If you think about it, interpreters are little more than parsers with a symbol table.)

Using ASCII text files for storing data shouldn't be limited to large database-type projects either. Tim used

this approach to write a simple metric-to-decimal conversion program for his father.

Then when his father wanted to add new conversions, he used his word processor to add them. Try that when the data is buried inside the program code.

◆ ◆ ◆



Come to
SOG VII!

July 14 - July 16

Bend, Oregon

Figure 1 - Program To Parse Data File For Directory String

```
/* program get.exe
Oct. 23, 1987
Version 0.2
JJP

This program moves the user
from the current working directory
to a new directory.

Copyright (c) 1987 by Jack Purdum

Each user is allowed to distribute up to 11,563
copies for non-commercial use. */

#include <stdio.h>
#include <stdlib.h> /* Has prototypes for "un-headered" functions */
#include <string.h> /* Need for string functions */

#define DATAFILE "get.dat" /* Name of directory-info file */
#define MAXBUFF 256
#define MATCH 0
#define FAIL 1

char buff[MAXBUFF], /* Input buffer for file reads */
path[MAXBUFF], /* Buffer for full path name */
drive[3]; /* Buffer for disk drive */

/* Prototypes for non-standard functions */

int search_file(FILE *fp, char *where);
int do_switch(void);
FILE *open_data_file(char *name);

int main(int argc, char **argv)
{
    int flag;
    FILE *fpin;

    if (argc != 2) {
        fprintf(stderr, "Usage: get directory_abbreviation_name");
        exit(0);
    }

    fpin = open_data_file(DATAFILE); /* Open the data file */
    flag = search_file(fpin, argv[1]); /* Find the pseudo-name */
    fclose(fpin); /* We're done reading */

    if (flag == FAIL) {
        fprintf(stderr, "\nCannot find %s in data file.\n", argv[1]);
    }
}
```



```

    exit(0);
}

flag = do_switch();                /* Try the change */
if (flag != MATCH) {
    fprintf(stderr, "\nCannot find directory %s\n", path);
    exit(0);
}
}

/*
do_switch()
Function to do the work.
Argument list:    void
Return value:    int    0 if successful, 1 on error
*/

int do_switch(void)
{
    int flag;
    system(drive);                /* Function to issue command to command.com */
                                /* and places us on the correct drive. */
    flag = chdir(path);          /* We should be there now. */
    return flag;
}

/*
search_file()

Function reads the data file that holds the information about the
directory locations, looking for a match on the string passed in.
The format for the input file is:

abbreviation drive full_pathname\n

where:
abbreviation -- the abbreviation for the full path name
drive -- the disk drive designator for the search
full_pathname -- the full MSDOS pathname desired
\n -- a newline terminates the data for each
possible directory

Example:

tut c \lessons\prog\tutorial

When the user types: go tut, the user is moved to the \lessons\prog\tutorial
directory on drive C:

Argument list:    FILE *fp    a FILE pointer to the open file.
                  char *where a string constant that is the short
                              name for the directory desired.

Return value:    int    0 if successful, 1 on no match
*/

int search_file(FILE *fp, char *where)
{
    char *ptr, temp[MAXBUFF];
    int c, i;

    i = 0;
    while ((c = fgetc(fp)) != EOF) {
        if (c != '\n') {
            buff[i++] = (char) c;
            continue;
        }
        buff[i] = '\0';
        strcpy(temp, buff);
        ptr = strtok(temp, " ");
        if ( strcmp(ptr, where) == MATCH) {
            ptr = strtok( (char *) 0, " "); /* Get the disk drive */
            strcpy(drive, ptr);
            strcat(drive, ":");
            ptr = strtok( (char *) 0, " "); /* Get the full path name */
            strcpy(path, ptr);
            return MATCH; /* ...and we're done */
        }
        i = 0; /* No match, start over */
    }
    return FAIL;
}

/*
open_data_file()

Function attempts to open the data file that holds the information
about the directory locations. It does this by searching the PATH

```

Turbo C Programmers, Developers or Students

Advanced, Intermediate, Novice

BAS/C® Pack

Business Applications
Source code in C

A REVOLUTIONARY PRODUCT !!
Contains the complete Turbo
C source code for 8 MAJOR
Business Applications in
modular form.

•General Ledger •Inventory
•Accounts Receivable •Payroll
•Amortization •Order Entry
•Purchase Orders •Accounts
Payable

FIRST TIME EVER !! Quality
source code for major business
applications at BARGAIN prices.

SAVE THOUSANDS of coding
hours and MONTHS of development !

INCLUDES HUNDREDS of TESTED
Accounting routines.

INCLUDES MORE THAN 60
Report formats, Check formats,
Invoices, Credit Memos, Statements.

EDUCATIONAL: Learn how an
accounting system is designed.

USE MODULES or ROUTINES as
is or modify as required.

EXTENSIVELY COMMENTED
Code for ease of use.

TRANSPORTABLE TO UNIX
systems. Uses ANSI screen handling.

PASSWORD PROTECTED at
users option for all APPLICATIONS.

PAY NO ROYALTIES ! Use any
or all modules or routines in
your systems.

Introductory
Offer !! **\$179.00**

What you get: More than 70 Business
Applications Programs in the BAS/C Pack.
More than 90,000 lines of Turbo C code.
Shipped on 5 1/4" diskettes in MS-DOS
360K format. 300+ pages of manuals in
text files for easy editing. Numerous
programmers' utilities and batch files
for compiling/linking. C Function Run
Time library. Data file layouts.
Programmer's notes. **BONUS:** For a
limited time we will include the BASIC
source code that these programs were
derived from AT NO EXTRA CHARGE !!

For VISA/MC Orders CALL:
1-800-824-2222 Ext. 26

ORDER FORM

BAS/C Pack @ \$179.00 _____
Texas Residents add 7% tax _____
Shipping \$7.50 _____
TOTAL _____

Name: _____
Address: _____
City: _____ State: _____
Zip: _____ Phone: _____

BAS/C® software
13810 Champion Forest Dr. #235MC
Houston, Texas 77069 (713) 580-0104

Reader Service Number 74

Come to
SOG VII!
 July 14 - July 16
 Bend, Oregon



Registration
 begins
 May 1st!

1- (503)-382-5060
 Come join the fun!

environment variable. The program assumes that the dat file is in the PATH directory.

Argument list: char *name a string constant that is the name of the file containing the directory information.

Return value: FILE *fp a FILE pointer to the open file if successful, program aborts on error.

```

*/
FILE *open_data_file(char *name)
{
  char *ptr, p[MAXBUFF], temp[MAXBUFF];
  FILE *fpin;

  ptr = getenv("PATH");          /* Find out where go.dat is. */
                                /* NOTE: PATH must be in caps */

  if (ptr == NULL) {
    fprintf(stderr, "\nPATH not set. Program and data file must \
    be on PATH.\n");
    exit(0);
  }
  strcpy(p, ptr);
  ptr = strtok(p, ";");          /* Find first PATH */
  while (ptr != NULL) {
    strcpy(temp, ptr);          /* Save a copy of substring */
    strcat(temp, "\\");        /* Form a path and file name */
    strcat(temp, name);
    if ( (fpin = fopen(temp, "r") ) == NULL) {
      temp[0] = '\0';          /* Start over again */
      ptr = strtok( (char *) 0, ";"); /* Try next path */
    } else
      break;                    /* Must have a good fpin */
  }

  if (fpin == NULL) {
    fprintf(stderr, "\nCannot find %s on default path(s).\n", name);
    exit(0);
  } else
    return fpin;
}

```

What have YOU been missing?

C

Programmer's Toolbox Volumes I & II

Create better, faster, higher quality and easier to read programs in a fraction of the time. Let your system do the work for you. With 23 powerful, state of the art tools for the IBM PC and compatibles, both beginners and experts will find programming a breeze.

Easy to use. Unlimited program sizes. Online documentation...

- CFlow™:** Determine program hierarchy, external R/T library functions, etc.;
- CLint™:** More rigorously check program syntax;
- CPrint™:** Beautify source programs to user selected formats;
- CXref™:** Cross reference and check symbol usage;
- CritPath™:** Determine a program's critical path;
- PMon™:** Monitor program/OS execution; and more...

At \$79.95 per volume or \$130 for both with a 30 day money back guarantee, the Toolbox is simply the best value today. The Toolbox works with your existing C compiler(s) and enhances your development environment.

Call and Order Today
 Visa, MasterCard Accepted



MMC AD Systems
 Box 360845 Milpitas, California 95035
 (408) 263-0781

"The C Tool Specialists"

Reader Service Number 57

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.



SOFTWARE SYSTEMS CORP.

BOX 209 - MT. KISCO, NY 10549
 914/241-0287

Reader Service Number 40



Finally —
Affordable Intelligence.

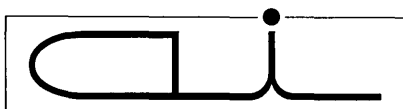
TINY EINSTEIN

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)

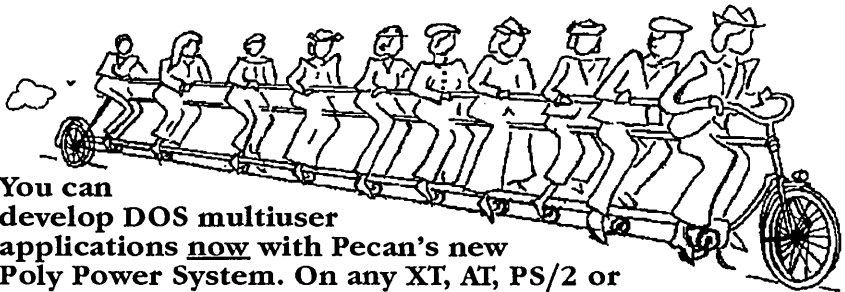
Designed & implemented by
Gary Entsminger & Larry Fogg



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

Reader Service Number 72

MULTIUSER DOS NOW!



You can develop DOS multiuser applications now with Pecan's new Poly Power System. On any XT, AT, PS/2 or compatible. Under any version of DOS 2.0 or later.

All you need is Pecan's Poly Power System, one or more of our compilers: UCSD Pascal, Modula-2, FORTRAN-77, BASIC, Assembler (C is coming soon), and inexpensive dumb terminals on serial ports.

You can even have multiple programmers develop on a single machine.

Poly Power System, \$399.95. Compilers, \$79.95 each. Call us for full details and to place your order.

PECAN
The UCSD Pascal Company
1-800-63-PECAN

Pecan Software Systems, Inc.
1410 39th Street
Brooklyn, NY 11218
718-851-3100

Pecan Software Europe, Ltd.
MGM House, Oakfield Grove
Clifton, Bristol, BS8 2BN, England
0272 733633

Reader Service Number 29

FULL AT&T C++: ANNOUNCING VERSION 1.2

Guidelines announces its port of version 1.2 of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, inheritance, member functions, constructors and destructors, data hiding, and data abstraction. "Object-oriented" means that C++ code is more readable, more reliable and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is Bell Labs' answer to Ada and Modula 2. C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: \$295

Requires IBM PC/XT/AT or compatible with 640K and a hard disk.
Note: C++ is a *translator*, and requires the use of Microsoft C 3.0 or later.

Here is what you get for \$295:

- The full AT&T v1.2 C++ translator.
- Libraries for stream I/O and complex math.
- The C++ Programming Language, the definitive 327-page tutorial and description by Bjarne Stroustrup, designer of C++.
- Sample programs written in C++.
- Improved installation guide and documentation.
- 30-day money-back guarantee.

To Order:

send check or money order to:

GUIDELINES SOFTWARE, INC.
P.O. Box 749, #MCC
Orinda, CA 94563

To order with Visa or MC,
phone (800) 634-7779;
in CA, (415) 254-9393.
(CA residents add 6% tax.)

C++ is ported to the PC by Guidelines under license from AT&T.
Call or write for a free C++ information package.

Reader Service Number 55

Comdex, A Hacker's Computer Fix

And A Manufacturer's Future

I won't tell you all about Comdex. Sure, I could have listed the booths, rewritten the releases, and thrown in a few sanguine comments from company presidents, but that wouldn't be Comdex.

Comdex is parties and face-to-face discussions and information. It's friendships made and renewed, backs patted and stabbed. And, finally, it's money. Lots and lots of money.

This year, Comdex was the place where more than a few small designer companies were writing contracts, selling product, doing business. For instance, Matech Computer Systems (of Bedford, OH) was showing its new 80286 card. It's a 10 or 12 MHz AT on an XT sized plug-in card. On board there are: 1 to 6 meg ram, floppy controller, two serial, one parallel, keyboard port, clock & battery, 80287 socket, and provision for a daughter board (with hard disk and VGA controller). Retail price is \$795 (10 MHz, 1 meg RAM). The card has an Award BIOS and only works with empty mother boards.

They couldn't take real orders because they hadn't received FCC approval but tentative orders totalled 160,000 units. (And they weren't selling onesies.)

Tracking The Market

First, a look at what's happening and why.

Computers are getting faster, memories are getting larger, and there's more software. (There, I said it. Again.)

What's driving the market is changing, however. Instead of discussing whether 64K of RAM is really necessary we're talking about displaying graphics at 1,024 by 1,024 with 64 colors or 256 colors or 32,000 colors or...

Meanwhile, scanner manufacturers

are releasing the first units which handle grey tones at 4 or more bits per pixel. (No longer content to dither.)

Now, lets see. 1K by 1K at 1 bit per pixel is 1 megabit. At 8 bits per pixel it's 1 megabyte. At 32 bits per pixel it's... And that's just a single screen image.

Tie together image scanning, editing, display, incorporation into a document, and printing and you're talking about heavy duty processing. Really heavy duty.

There was a small, out of the way room at the convention center. TI rented it and lined it shoulder to shoulder with living, breathing (and sweating) designers. They were showing off new products using the 34010 graphics processor. (You know, the one Dean Klein wrote about in Issue #39.)

The race for the standard graphics engine has been between Intel's dedicated graphics controller which can rotate, size, fill... in a flash—and TI's 34010, a real microprocessor with very complete instruction set (including floating point primitives), a 16-bit data bus, and a 40 MHz clock. The 34010 also contains a souped up 6845-style video and sync generator.

From the activity in the room and from talking to other folks around the show (some of whom had designed boards with both the TI and Intel chips) it looks like TI's winning the graphics engine competition.

OK, what difference does it make to you and me whether there's a super screamer graphics processor hidden in our new graphics card? Well, there are some very interesting high-level graphics which are relatively easy for programs and CPUs to support. Graphics languages include PostScript, DGIS, Halo, Nova CGI, Gem, Windows, Borland's new graphics interface, and so on. (PostScript is a relatively high-level interface. DGIS lies at the low level end.)

Your graphics editor and desktop package, for instance, might output graphics and text in PostScript form. Your scanner might also generate PostScript.

Meanwhile, if your video monitor also understood PostScript then everyone would be speaking the same language. Want to redirect the image? Redirect the PostScript.

Timely

If you've ever watched AutoCad rotate a complex image, you know what a struggle it is for a CPU (even a fast CPU with a math co-processor) to manipulate graphics (simple 1-bit per pixel monochrome graphics).

What if the CPU could simply instruct the graphics processor to rotate the image 3 degrees clockwise? And what if that graphics processor loved doing 3-degree rotations? (You get the picture.)

You know how slow Windows is? TI has developed 34010 firmware which speeds up Windows by an order of magnitude.

Anyway, the 34010 and it's successor (5 MIPs, 32-bit data bus, 50 MHz) look like real answers to graphics horsepower problems. Plus, as true processors, they should make it possible for current graphics devices to support new graphics languages. (After all, it'll just require software!)

For more information on the 34010 contact:

Texas Instruments
PO Box 809066
Dallas, Texas 75380
800-232-3200

A Disassembler

While the 34010 is a fancy chip, produced by a large corporation as an answer to an industry-wide problem, I

found another product at Comdex that helps solve a very old problem for a relatively small group. For me, this is as interesting a product as the 34010.

It's called Sourcer. Feed it an 8088 (or V20, or 80186, or 80286, or 80386) .COM or .EXE file and turns it into source. Well-commented source. (Hey now, how is a disassembler going to comment the code?)

It can comment the code because it assumes the code will run on a compatible (OK, that provides the DOS and BIOS calls). It also makes up to five passes through the code, digging out data (including index tables), marking calls to subroutines, and in a crude way, actually running the code.

I asked Frank van Gilluwe, the author, how he knew so much about every program I'd disassembled.

"For one thing, I keep tabs on the contents of each register as I walk through the object code. That way I can specify in the comment what value is being output to a port, or I can tell where an index call will wind up. If the program has several segments I can tell when code in segment 1 uses data in segment 3 (or vice versa).

"Also, I try to explain the complex instructions well enough in the comments so that novice assembly language programmers can understand them."

As far as I know, this is the only product which automatically disassembles indexed calls (used a lot in BIOS calls), the only product which lists who's calling a subroutine at the beginning of the subroutine (rather than in a table at the end of the listing), and this is the only disassembler I've ever used where I simply SR, the object filename, a 'G' and 20 seconds later have a complete, commented listing.

This all got started when Frank wrote a program to generate commented source of IBM's PS/2 BIOS. (IBM isn't

releasing the source for PS/2.) He markets that product as the "BIOS Pre-processor." Just run it and it generates a commented listing of your BIOS ROM whether it be in a PC, XT, AT, PS/2, or clone.

(It had no trouble generating a thoroughly commented source of the X-16's BIOS ROM. Over 200K in a few minutes. Dean and Earl at PC-Tech will be quite impressed (concerned?) when they see my commented disassembly of their code.)

I asked Frank what people have been doing with the program.

"One outfit mentioned they had lost all their source files for a current product. They used sourcer to recreate the source. Another outfit bought it after they noticed their programmer's comments were unintelligible. An individual used it to find and remove an obnoxious bell from his favorite game.

"A number of companies have purchased it to dig into Sidekick. They have to be compatible with the program and they can't get specific details or help from Borland so they disassembled it.

"Usually, though, people don't tell me what they're doing. They may have the feeling they're doing something wrong, getting in and peeking at someone else's code. But it's really OK."

The BIOS Pre-processor and Sourcer are available from V Communications. The Sourcer alone is \$99.95, with the BIOS Pre-processor, it's \$139.95. Frank is working on an MS-DOS preprocessor/disassembler. It should handle most of the 2.X and 3.X versions and should be available by the time you've read this. (Depending on how fast you read, of course.)

V Communications
3031 Tisch Way, 2nd floor
San Jose, CA 95128
408-296-4224

A Language

I know why I haven't picked up Prolog. It's hard to understand. I'm process oriented, so as far as I'm concerned a language needs a goesinta and a goesouta and inbetween it should make some kind of sense.

Pascal, C, even BASIC work like I do. But these languages don't think (backtrack). You can't just throw a problem at them and ask them to muddle through.

At this year's Silicon Northwest Press Reception (yep, another party) I was within 10 feet of the supply of fresh crab when I found myself face to face with Paul Voda. He was standing at an otherwise deserted table holding up a small plain manual and smiling.

"What's that?" I asked.

"Trilogy," he beamed.

"What's that?" I asked.

My timing has never been good. This time it was terrible. If I'd loaded up with crab and then returned for a Q & A session it would have been a perfect evening. But no:

"It's a language that has the power of Prolog and C and the structure of Pascal."

(Oh boy, another language. I'll let him send me a copy if it doesn't have GOTOs.)

"It's a complete programming environment, editor, compiler, debugger, library, linker... It can run just like Prolog, programs can ask questions, contain symbolic variables whose values are determined by backtracking."

"Unreadable," I thought.

"But we're a lot like Pascal with a concise nested program structure rather than Prolog's hard to read clauses. Plus we have 'case' and 'if' clauses. You can just as easily write fast, non-backtracking code, that reads like Pascal. And, even our backtracking programs run 10 to 100 times as fast as Prolog."

"Maybe I could fudge and just check

out the Pascal part," I thought.

"And we have arrays, structures, lists, and unions. All as concisely implemented as they are in C."

"Unions and structures?" I'd almost forgotten the disappearing crab.

"Plus a few things that aren't in any other language such as a decision procedure to set constraints on variables so the program doesn't waste time trying ridiculous values. A database file query system that reads dBase files. Plus we have incremental compiling, linking, and loading of program modules as well as new data types like 'injections' and 'relations.'"

"Injections and relations sound as appealing as getting a tetanus shot from my father," I mused.

But, he'd already made enough good points, so I asked for a copy.

Back at my office, I surrounded myself with piles. Releases here, notes and business cards there, books here, software there, the really interesting stuff front and center. On top of the really interesting stuff was Trilogy.

Two hours with the tutorial (this is a truly great manual) proved everything Paul had told me. (These technical folks, you can almost trust them.)

It's a simple language with only eight constructs: and, or, not, if, case, all, relations, and predicate.

It's an incredibly fast language, even when doing Prolog style problems. By having you define inputs, outputs, and variables, and by doing some sophisticated range checking, it produces iterations of the 8 queens problem so fast that you have to write in a delay in order to see the results.

Paul has done a lot to make the code more readable than other AI languages. For example, math is standard algebraic format (e.g. $c = 5 + 3$) and the structure of the search is much simpler. It's also a complete programming environment (editor, windows...) quite similar to the Turbo series.

Everything I saw in this language told me that its author was no lightweight. And it was obvious, even in the manual, that this had been a long-term project. (I can name a lot of good compilers that have bad manuals—and a lot of bad compilers that have bad manuals—but I know of no bad compilers that have really good manuals.)

I called Paul and asked him to write a simple (20 lines max) program in Pascal, Prolog, and Trilogy to give you a feeling for the language. Here is what he sent me:

Fig 1: Pascal Equation

```
program Equation;
var a,b,c:integer;
procedure Eq(a:integer;b:integer;c:integer);
var x,y:integer;      { a*x + b*y = c }
begin
  if (a > 0) and (b > 0) then
    for x := 0 to c div a do
      for y := 0 to c div b do
        if a*x + b*y = c then
          writeln('x = ',x,' y = ',y);
end;
begin readln(a,b,c); Eq(a,b,c); end.
```

Fig 2: Prolog Equation

```
eq(A,X,B,Y,C) <-
  A > 0 , B > 0 ,
  Xu = C/A, gen(X,Xu),
  Yu = C/B, gen(Y,Yu),
  A*X + B*Y = C.
gen(X,N) <-
  N > 0, N1 = N-1 , gen(X,N1) .
gen(N,N) <- N >= 0.
```

Fig 3: Trilogy Equation

```
pred Eq(a:<L[1..], x::L[0..], b:<L[1..], y::L[0..], c:<L) iff
  a*x + b*y = c
```

"Dave: As we have discussed over the phone, I have prepared three very short programs (in Pascal, Prolog, and Trilogy) 'naively' solving Diophantine equations. I know they are longer than 'Hello world' but they solve a real problem.

"The basic differences among the languages:

- Pascal: explicit control backtracking by loops.
- Prolog: no explicit loops but values must be 'guessed' by the programmer.
- Trilogy: no loops no guessing.

Problem:

"Given an equation $a*x+b*y = c$ with integer coefficients a, b, c (a and b are positive) find all non-negative integer solutions x and y . Rather than encoding the algorithm for Diophantine equation solving, solve the problem by a trial and error (generate and backtrack).

"In Pascal you have to read, generate, and print the results explicitly by a loop. (See Figure 1.)

"In Prolog the reading, printing and looping (via backtracking) is automatic. You simply enter a query, for instance:

```
<-> eq(6,X,8,Y,46)
```

to get two sets of results $X=1, Y=5$ and $X=5, Y=2$. Note that the generation part

must be explicitly programmed by the recursive predicate "gen". (See Figure 2.)

"In Trilogy you also enter a query

```
all Eq(6,x,8,y,46)
```

to get the same results without any backtracking. The constraints make it possible to generate a direct solution. (See Figure 3.)

"Note, for instance, the unknown x is constrained by its parameter type to be an (long: L) integer such that $x \geq 0$."

Long Time Writing

It turns out that Paul Voda has been writing compilers for 20 years. He and some of his students (finishing up their PhDs) have been working on Trilogy for three years. Even the manual writer is a CS PhD candidate.

Paul is familiar with the folks who created LISP and Prolog. These languages were also based on predicate calculus but the people who finally implemented the compilers added extra-logical features. These additions made the languages much more difficult to predict mathematically.

The idea is if you can prove the compiler is correct and prove your own source code is correct, then you can prove the final object code. This moves the software reliability problem out of

the dominion of endless tests and into the laps of mathematicians.

Since Paul and his students are both theoreticians and programmers, Trilogy has avoided the plague of hackers. It's a theoretically valid language that at a cursory glance appears very, very usable.

At this point the language is unknown outside of Vancouver, BC. That will change very quickly. This is too good a language and too good a development environment. (Now if I could just find some good fresh crab.)

Trilogy \$99.95 + \$5.00 Shipping (U.S. funds)

Complete Logic Systems
741 Blueridge Ave
North Vancouver, B.C. V7R 2J5
604-986-3234

P.S. Paul Voda has agreed to come to SOG VII to talk about designing the language, using the language, and he'll do a more general presentation on designing compilers using predicate calculus. See you at SOG VII, July 14-16 in Bend.

Postscript

Several companies are still trying to

muddy the page definition language scene. I saw folks pushing DDL and ACE like there was no tomorrow (which is pretty close).

Genicom says its ACE is much faster than PostScript. Ten times faster. At least with their image processor on their laser printer.

Interesting.

PostScript printers are slow turning out the first page of a book, etc., because the 68000 is busy translating font outlines into pixel definitions and storing them in memory. Once the translation is complete the pace really picks up. I asked if ACE defined type faces via outlines.

"Yes."

And didn't the image processor have to translate the outlines into pixel representations before the characters could be used?

"Yes."

So why was ACE faster than PostScript?

It turns out that they've added a bitslice processor to speed up the outline-to-pixel translation. OK, so we're talking a lot more about hardware differences than language differences.

I asked if ACE had all the features of PostScript.

"No, but that's not important. Most of the market is less than power users and when people need the additional features they can get PostScript."

I don't know how you'd feel being called a "less than power user" (a power user is someone who's just discovered BASIC), but if someone called me that, I'd byte him.

Anyway, I'd think twice about paying a premium for a printer with a limited, non-standard page description language. No matter how quick.

Right Ace?

◆ ◆ ◆

TRIOLOGY

A Powerful Procedural, Database, and Declarative Language.

SPEED — Where Prolog must backtrack, Trilogy can often solve the problem logically. Trilogy 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 — Trilogy uses an intuitive, Pascal-like, program structure.

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

MODULARITY — Trilogy 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 . . .

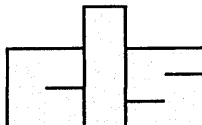
A TRUE COMPILER — Trilogy is an interactive compiler which produces native code for the 8086/8087.

LOGICAL PURITY — Trilogy was designed from scratch as a simple language with a completely logical foundation. Trilogy'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 — Trilogy 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:

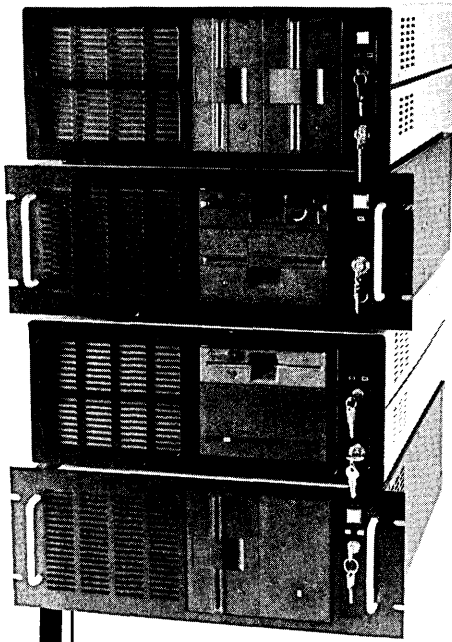


**COMPLETE
LOGIC
SYSTEMS**

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

ONLY \$99.95

Reader Service Number 71



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

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.

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
SBA51C Column Begins
Upgrading Kaypro II to 4
Upgrading Kaypro 4 to 8
48 pages

ISSUE #16 (2/84)
Xerox 820 Column Restarts
BBI Double Density
BBII 5 1/8" Interface Fix
Kaypro ZCPR Patch
Adding Joystick To Color Graphics
Recovering Text From Memory
52 pages

ISSUE #17 (4/84)
Voice Synthesizer
820 RAM Disk
Kaypro Morse Code Interface
68000-Based System Review
Inside CP/M 86
56 pages

ISSUE #18 (6/84)
Kaypro EPROM Programmer
I/O Byte: A Primer
Kaypro Joystick
Serial To Parallel Interface
Business COBOL
60 pages

ISSUE #19 (8/84)
Adding Winchester To BBII
6 MHz On the BBI
Bulletin Boards
Track Buffering On Slicer
4 MHz For The 820-1
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' C
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 Device 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

ISSUE #35 (4/87)
Building An 8-channel Temperature
scanner
Designing An Expert System
Teaching Your PC To Beep
Who's Making Great Hard Drives?
Learning Assembly Language
96 pages

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

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.

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

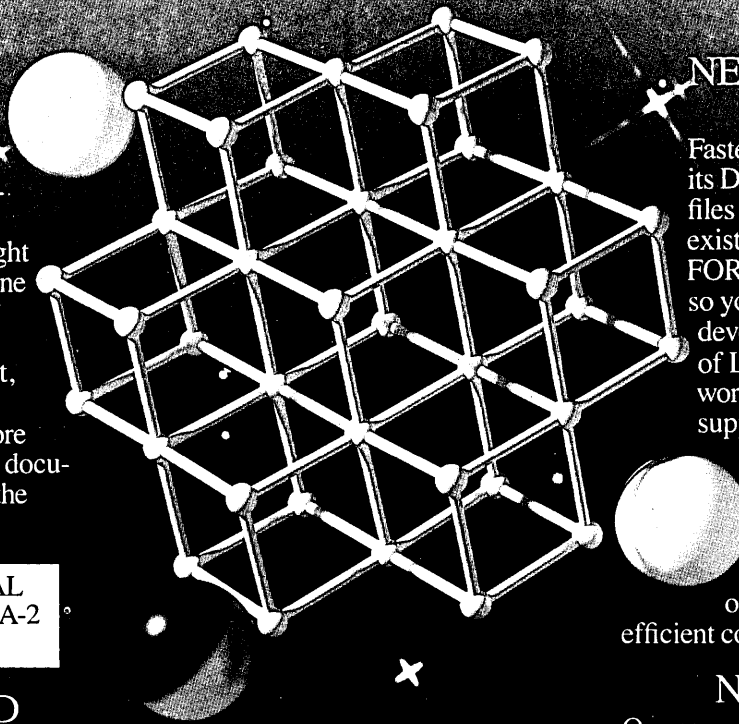


project throughput. The Post Mortem Debugger analyzes the status of a program after it has terminated while the dynamic,

Run Time Debugger monitors the execution of a program with user-defined breakpoints. 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

A Programmer's Introduction To C++

C Grows Up

Here's C with extensibility and objects. That should make the FORTH and Smalltalk people happy. It also accepts standard C programs so C folks should like it.

At this point, however, it's only available as a preprocessor for standard C—just one of the growing pains of this very new language.

You're in a bookstore, searching for a science fiction novel, one with a stimulating picture on the cover. The cover, of course, has nothing to do with the content, but you're here looking for escape and vulnerable to suggestion. And publishers know these things. So you find one and begin the acid test: you open to the middle and start reading.

"High Thizbin!" exclaimed Wicna, "this is a matter of Feeblum! There will be oompla between the Snorts and the Bingos if we don't show our flexmuscle by using the Whambammer!" The Thizbin looked thoughtful as he absently inserted his digiplex into a zitplorg....

It's a new language, and one (you suspect) created as a smokescreen for bad writing. Mumbling something about literary license and good ol' English, you jam the book back onto the shelf and move on.

It's a familiar case—bad something trying to disguise bad something else. But sometimes a concept is good and sometimes it genuinely needs a new language.

Another Programming Language

In November I attended the first Usenix (Unix User's Group) C++ Workshop. There was a lot of arguing between two subgroups: one wanting to extend the language and one thinking it was already too big. A third, hardly visible, subgroup is the majority of programmers who haven't heard of

C++. (And the programmers who are happy with what they're using and think there are already enough languages in the world.)

In the middle of all this is C++'s creator, Bjarne Stroustrup.

"Bee-yarn-eh," he says. "And you have to be a Dane to pronounce my last name."

He makes a sound like clearing his throat. And then he makes a case for keeping C++ small: "small languages have been more successful."

What's C++?

(1) A better C. Many of the ANSI C improvements have been taken from C++ (for example, function prototyping). You can take old C code and compile it with C++. C programmers can use the language immediately and learn new features as they need them.

(2) C++ supports user-defined types. This is different from C's "typedef," which simply aliases one thing for another. A true user-defined type is indistinguishable from a native type (like int or float). Plus, it follows normal scoping rules and naming conventions.

With user-defined types, the compiler performs all the error checking it does for any other type. You can define operations on types: a matrix type can

have the "*" operator overloaded to perform matrix multiplication, for example.

User-defined typing (or data abstraction) is an important advance. Once we've packaged and debugged a type, we can use it without learning all the details of the internals—you tell it *what* to do, not *how*; and if you make a mistake, the compiler tells you.

(3) C++ is object-oriented. This means you can not only define new types, but you can define a new type as an old type with some features added. You don't have to know how the old type works, just what it does.

For example, suppose someone has defined a matrix type and you want to create a special type based on that matrix. You can say: "my_type is a matrix with some extra information about location, time, and operations (or methods) to set and extract that information." This is called inheritance, and it's one of the most powerful concepts in C++ (or any true object-oriented programming language).

If someone has gone to the trouble of defining and debugging a class, and you need something like it (but different), you don't have to throw it away and start from scratch. You simply say: "I'm creating a new object, which is like the old object with these changes." You make the changes and debug those.

This works even if you only have a linkable module for the object rather than the source code. Most of the problems with buying someone else's plain C object modules occur because you can't change the modules. In C++, you can still clean up their work even if they don't distribute source.

Real Objects?

You often hear that a product is good if it's object-oriented. Not necessarily, and for your own defense you should know that object-oriented isn't just the

Sometimes a concept is good and sometimes it genuinely needs a new language.

ability to create user-defined types. It's user-defined types combined with the ability to inherit from an older type.

From this definition, only Smalltalk, Lisp LOOPS, and C++ are truly object-oriented. (Stroustrup didn't mention Objective-C or Eiffel.) And, of these, C++ is the only object-oriented language with the deliberate aim of runtime efficiency.

Virtual Functions

Before I was introduced to virtual functions, the concept had never occurred to me. Now they seem indispensable. A lot of features in C++ are like that. It's very difficult to go back to plain C once you've had a taste.

A virtual function says, in effect, "there's a function here, but I don't know exactly what it is yet." The function is only defined in a derived class. A popular example is the "shape" class. Everything that's basic to shapes is defined in the base shape class.

When you want to make a specific shape (say, a square), you derive it from the base class by saying: "a square is a shape with some extra information about squareness."

But there's something else to know about shapes: they can be drawn, rotated, filled, etc. You want the base shape class to know that these functions are available (so you can, for instance, go through a list of shapes and rotate them all 45 degrees without knowing or caring what kind of shapes they are).

So you declare virtual functions in the base class. Then when you derive a square, the part about how it's drawn, filled, or rotated is already defined.

Using C++

The goal of C++ is to create libraries of reusable objects. When you need a certain kind of object, you include the object's header file, declare an instance of that object, and use it.

Figure 1 - Windows Object Definition

```
#include ``windows.h``
.....
main() {
.....
    /* create an instance of the object with specifications */
    window winl(x_corner, y_corner, x_size, y_size, color, title);

    /* Use ``methods`` of the object (the header file lists them) */

    winl.write(``this is a string in the window``);
    winl.gotoxy(12,24);
    winl.write(``press any key to continue``);
    getch();
    winl.clear();
}
```

Figure 2 - Windows Class Definition

```
class window {
    unsigned char old_screen[80 * 24 * 2];
    /* a place to put portions of the old screen */
    int x_corner, y_corner, x_size, y_size, color;
    /* Data above can only be modified by ``member`` functions declared
    below: */
public:
    /* how to make a new window object */
    window(int x = 0, int y = 0, int x_s = 0, int y_s = 0,
           int col = BLUE | BLACK_BACK, char * title = ````);
    ~window(); /* how to destroy an old window object */
    void write(char * st); /* how to write a string */
    void write(char c); /* how to write a char */
    void write(int i); /* how to write an int */
    void gotoxy(int x = 0, int y = 0); /* moving around */
    void clear(); /* how to clear yourself */
} /* end of class definition */
```

This is significantly simpler than using conventional C libraries—objects initialize themselves instead of requiring you to declare and initialize some data structure with marginally meaningful values.

For example, you can use a windows object someone else has written, with the header in WINDOWS.H (see Figure 1). Notice the appearance of getch(), an or-

dinary C function.

A Class (Object) Definition

The ability to define classes really distinguishes C++ from C.

A class is a C structure with a twist: it contains functions as well as objects. A windows class definition might look like Figure 2.

Note in the code: we see functions as

just another kind of structure member. And there are three definitions for "write." This is called overloading and clarifies your code. For example, when writing something, you don't want to worry about WHAT you're trying to write. You just want to write it. So for our window, we can say:

```
win1.write(123);
win1.write('a string');
win1.write('c');
```

Another nice feature is the default values in gotoxy(). This allows us to say:

```
win1.gotoxy();
```

to automatically mean go to $x = 0, y = 0$.

Operator Overloading

Operators (+, -, *, ==, etc.) can also be overloaded with new meanings. These meanings are invoked automatically when the operator is applied (adding two complex numbers is different than adding two reals).

For example, the language has replaced the clumsy and ugly C printf() function with a concept called the stream. The "<<" and ">>" have been overloaded for this purpose (although they retain their usual meaning when working with integers). Instead of stdin, stdout and stderr, we write to cin, cout and cerr as follows:

```
cout << 'an integer:'
< int_num << 'a float:'
< floatnum;
```

No more format statements. No barfing if you don't have the right number of arguments. So it's much more readable. There are many other features of streams which make file handling much easier than with old C.

Extensibility

Why is all this so useful? For one thing, it helps you organize your code (Modula-2 users may see some familiar ideas). More importantly, once a class is completely defined, it becomes part of your compiling environment—you have truly extended the compiler, and it will check errors for you as if your class were built-in, like an int or a float.

Extensible languages have gotten a lot of bad press, and rightfully so. Nowhere in his book (*The C++ Programming Language*, by Bjarne Stroustrup, Addison Wesley, 1986) does Stroustrup use the word "extensible," and I think

it's because he doesn't want C++ besmirched with FORTH's reputation.

FORTH is a fascinating language. You create new keywords with the same status as the originals. Creating keywords is what FORTH is all about—you just build keywords until you have one that does what you want and call that your program.

Once a new keyword submerges in the bubbling soup, it becomes part of the morass. There isn't a soup, there's only

Nowhere... does Stroustrup use the word "extensible," and I think it's because he doesn't want C++ besmirched with FORTH's reputation.

your soup. You make your own keywords which have meaning to you, and you write your programs in your own personal "Captain Midnight Decoder Ring" programming language. This is why FORTH is called "write only"; no one has the patience to go back and decode what you did.

I think FORTH might make a dynamite assembly language, now that it's being cast in silicon for microprocessors. But for general-purpose programming, it's too out-of-control.

C++ has enough restrictions that extensibility is manageable. The syntax is consistent, you must write header files (so the user can see the functions you're using) and the compiler checks your syntax when you're using a class. It's more work than defining a new keyword in FORTH, but it does a lot more for you.

Introducing DAIMS

DAIMS (Data Analysis and Interactive Modeling System) is a project funded by the Institute of Naval Oceanography, with principal inves-

tigators Tom Keffer and Dale Haidvogel, and one programmer (me).

Why are we doing this? Scientists are *still* programming in Fortran. They clutch with white-knuckle fierceness to the object which really slows them down.

It's understandable. Fortran was one of the original non-assembly programming languages, and scientists have been using it since the fifties. As a result, there are tons of debugged scientific Fortran routines out there, and it's easy to think "all I have to do is link this package, and that package, and the other one, and it's done."

Of course, it's never that easy. We do need to tweak, and some things just won't talk to others. The scientist kills a lot of time struggling with the language, and not much time struggling with science. Fortran 1, scientist 0.

Our goal is to leapfrog all that stuff and make it "that easy." To do this, we need to lure people away from Fortran. It's been tried before. So the lure has to be powerful.

We chose C++ because it offers the power of object-oriented programming without losing the efficiency of C.

We plan to produce the following:

(1) a standard, simple, data storage and retrieval system, along with C++ classes and utilities to perform these operations;

(2) as many C++ classes as we can create (including interfaces to existing Fortran packages) or collect from public domain sources;

(3) an interpreter including the most useful classes, and we're porting version 1 of the interpreter to the PC.

I've tried to make the interpreter a cross between BASIC and C. It will evaluate arbitrarily complex expressions involving integers, double floating-point numbers, and matrices of any size (which it easily reads from disk files). All the results, along with source code, will be public domain.

C++ Compilers For The PC

There are at least two C++ translators for the PC, and I've heard rumors of a third. Note I'm calling them translators; generally your code is translated into standard C code, which you run through your C compiler. The current compilers of choice are Microsoft and Lattice, but I suspect Turbo C might work with some tweaking if you use the Microsoft linker.

I had no trouble getting the translator (usually called "cfront") from Guidelines, but getting the Advantage

C++ was much more difficult.

At the C++ workshop, I met the marketing group (Oasys, who sell the PC version through Lifeboat) and the developers (Glockenspiel from Ireland). They explained their reluctance in handing out Advantage for review.

The two compilers have been reviewed in *Byte*, *Computer Language*, and *PC Tech Journal*. Apparently, the reviewers were not C++ programmers. They ran some benchmarks, compared prices (Guidelines is \$195 while Advantage is \$495), and said, "might as well get the cheaper one."

Glockenspiel (Advantage) claims to have made some 200 bug fixes to the AT&T code. Also, he claimed Advantage was better "adapted" to the Intel architecture.

I ported the AT&T version of the C++ translator to my Sun 3 Workstation. Then I took working source code from the Sun and moved it over to the PC. In two rather critical cases the Guidelines compiler broke.

Problems With Guidelines

In my parser (which I can't split up into smaller pieces) Guidelines runs out of memory. In my lexical analyzer, it just hangs up the machine.

The technical help line has a recorder which plays "I'll get back to you." It lies.

The Guidelines compiler seemed to work fine for ordinary C++ (although it wouldn't accept comments on the same line as a "#include").

But I want to take everything I do on the Sun and move it over to the PC, so I can't count on Guidelines (Note: to run either compiler, you *must* have 640K and a hard disk).

Advantages

The Advantage package was a refreshing change. The documentation was professional (although the "=" flag doesn't work as documented) and it includes a good introduction to the language (with a summary card).

Plus, they're very aware of the memory problem and are working on it as I write.

The next beta release will use free memory between 640K and 1meg for the compiler's internal stack. I think that will solve my problem.

You also get all the source code for the examples from Stroustrup's book and for the examples in the Advantage manual.

By following the suggestions in the guide, I was able to get Advantage to

compile my lexer. The parser, however, still ran out of internal stack. When I called Lifeboat, I got a real, knowledgeable, person who told me they would send the next release as soon as it's out.

I do have one question about their newest version: why stop at 1 meg? EMS 4.0 would use all the memory I could stuff into the machine. Then I'd *never* have to worry about the size of my parser.

On To The Compiler

Microsoft C version 4.0 was also a problem. Besides being clunky (I'm wishing for a C++ which works with Turbo C), it kept truncating long variable names.

The cfront translator tends to create long names, and while the truncation didn't break anything, it made me nervous. (Version 5.0 is supposed to fix this problem.) Microsoft's C is also tediously slow (after using GNU-emacs on the Sun).

If you want to do serious C++ programming on the PC, I think you need to spend the bucks and go for Advantage. Their support, manual, and examples are worth it. It's also the better choice for beginners despite Guidelines' lower price. Advantage has good examples, good documentation, and it's dependable.

Macintosh C++

Apple is developing a C++ translator for the Mac. The problem here is to define all the objects necessary to make Mac programming easy. I like all the built-in Mac stuff, but I have no desire to get on that learning curve. C++ could really make the difference here. We will port DAIMS to the Mac using a beta version.

Object Packages

So far, there's only one package of pre-defined objects for the PC. It's Pfor-Ce++ from Phoenix (the ROM BIOS People). I found many problems with the package and at \$395 list, it seemed like a bust. But then I discovered they had included source code.

So, at \$209 (Programmer's Paradise price) it might be worth it, but I don't trust the code.

(Or you can get the DAIMS source code on the Micro C bulletin board, and the Advantage people are preparing their own public-domain package—another reason I like them.)

The Book

There's only one book about C++. Stroustrup said he had to choose between writing a novice, expert, or implementer's guide. He wrote an expert's guide.

Chapter one is probably the most difficult. It uses concepts which are introduced much later in the book.

I suspect the author wanted to catch your attention by introducing all the powerful features right off, but it's confusing. I recommend starting at chapter two (two through four are a review of C and C enhancements). Then tackle chapters five through eight. They cover C++ specific features. Then try chapter one.

Stroustrup claims that people are working on six books, most of them introductory.

The Future

What we need is a system which helps the programmer organize his code in a sensible manner without adding a burden. C++ is an ideal candidate since it has the object-oriented code organization of languages like Smalltalk, but it retains the efficiency of C.

Stroustrup claims the C++ user base is doubling every eight months. I think the emergence of C++ will come from the grass roots, like its predecessor. It's the language I'll use whenever I can.

Guidelines Software

18 Evergreen Drive
Orinda, CA 94563
(800) 634-7779
(415) 254-9393 (In California)

Programmer's Paradise

42 River Street
Tarrytown, NY 10591
(800) 445-7899
(914) 332-4548 (In New York)

Phoenix Technologies

320 Norwood Parks
Norwood, MA 02062
(800) 344-7200
(617) 769-7020 (In Massachusetts)

◆ ◆ ◆

Inside The PC's PPI

Just Another Dumb Chip (Well, Just About)

There isn't much more boring than a parallel chip. However, before you say anything nasty about this grumpy frog remember, warts and all, it exercises a lot of influence. Make friends with this simpleton and you've got control. Lots of control.

What a drag. It's coming on Christmas—I should be high in the Cascade Mountains scouting a location for this New Year's snow cave gala. But here I sit, stuck writing about the PPI.

I know it's been a long time since I said this article was next. But quite frankly, just about everything else has been more interesting than this dive into Intel's decidedly unglamorous

Programmable Peripheral Interface (PPI) chip, the 8255.

The PPI won't paint pretty pictures on the screen, it won't talk on the phone, and it won't take out the trash. But it does perform some vital functions in the PC so it deserves a little attention.

General Description

As I've said before, just saying "PPI" makes me want to scamper off to the men's room. But Intel saw fit to call their parallel I/O chip the PPI, so in the interests of conformity and world peace, I'll stick with their name.

With 24 programmable I/O pins, the PPI is a general purpose workhorse. Like the rest of the smart chips under the PC's hood, it can do much more than the PC asks of it.

The 24 pins break down to three 8-

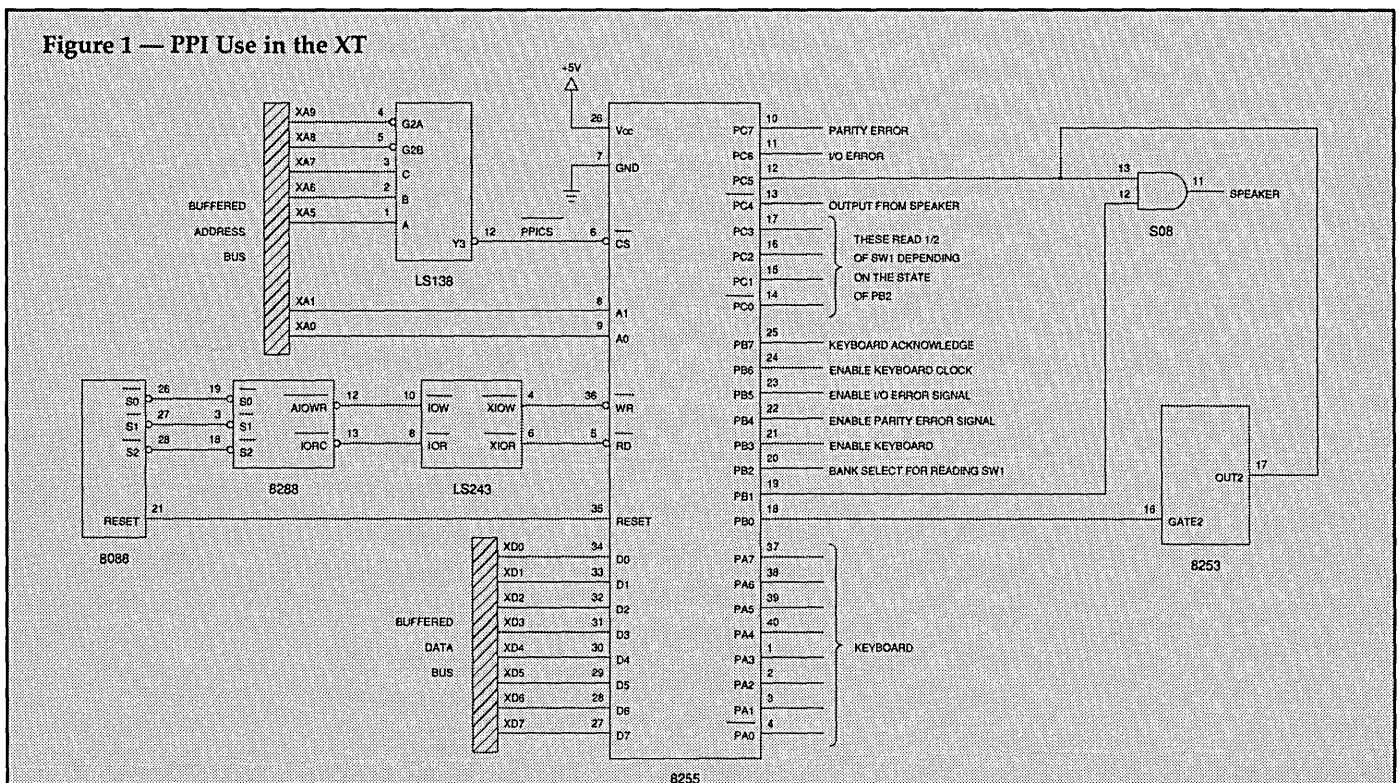
bit ports, Port A through Port C. You can program any of the ports for either input or output. Port A has the additional capability of bidirectional operation with Port C handling the handshaking.

Along with the three ports, the PPI contains a control register. This register sets the individual ports for either input or output, specifies the operational mode for each port, and allows setting and resetting bits in Port C. The latter function finds use in systems which use Port C for handshaking or interrupts. Setting or resetting an individual bit disables or enables the corresponding interrupt. The PC doesn't use this function.

Operational Modes

The PPI has three modes. Mode 0 is

Figure 1 — PPI Use in the XT



the basic I/O configuration. (The PC uses this simple mode.) Mode 0 allows no handshaking, only port reads and writes. Any port can be input or output (according to the control register setting), the outputs are latched, and the inputs are not latched.

By the way, "input" and "output" are from the CPU's point of view. A PPI output port accepts data from the processor and outputs it to the outside world (or whatever).

Mode 1 takes advantage of handshaking for more controlled, strobed I/O. Ports A and B each use one nibble of Port C for transmitting status and receiving control signals. Ports A and B can be either input or output with both inputs and outputs latched.

A nice feature of this mode (and Mode 2) is that when interrupts are enabled, an input device need only strobe its data into the PPI. The PPI will take care of generating an interrupt request signal for the CPU.

Mode 2 does strobed bi-directional I/O. It looks a lot like Mode 1 but data is received as well as transmitted. Five bits of Port C transfer control and status information between the PPI and the CPU. Only Port A may use Mode 2.

The PPI Pinout

The 8255 interfaces to the CPU in much the same manner as the interrupt controller (8259) and the DMA controller (8237). (Figure 1 shows a schematic of the 8255 as it's used in XT's and clones. Older PCs are a bit different. I'll detail those differences shortly.)

The active-low chip select (CS) lets the 8255 talk to the processor. Chip select signals for the PC's smart chips come from an LS138 3-to-8 decoder. Looking at TI's *TTL Data Book*, we see that any port address with bits nine, eight, and seven low, and bits six and five high, will select the PPI. At the

same time, the A1 and A0 address inputs to the PPI determine which of the PPI's registers will be on line.

The contents of A1 and A0 decode as follows: 00—Port A, 01—Port B, 10—Port C, and 11—control port. Rolling these together with the address input required for CS gives port addresses of

60h through 62h for Port A through Port C, and 63h for the control port.

The read (RD) and write (WR) inputs are active low signals which specify the direction of data and control flow. An 8288 bus controller decodes the S1, S2, and S3 outputs from the 8088 and generates I/O read (IOR) and I/O write

Figure 2 - PPI I/O Ports

Port A

In New PCs : Port A reads keyboard scan code
In Old PCs :
if PB7 = 0 : Port A reads keyboard scan code
if PB7 = 1 : Port A reads SW1

Port B

PB0 = 0 : disable 8253 Timer 2
= 1 : enable Timer 2
PB1 = 0 : disable speaker
= 1 : enable speaker
PB2 = 0 : PC0-PC3 reads top of SW1 or SW2
= 1 : PC0-PC3 reads bottom of SW1 or SW2
(old PCs read SW2, XT's read SW1)
PB3 = 0 : Turn cassette motor on (old PCs)
= 1 : Turn cassette motor off, or...
= 0 : Keyboard enabled (XT's and clones)
= 1 : Keyboard disabled
PB4 = 0 : Enable RAM parity error signals
= 1 : Disable RAM parity error signals
PB5 = 0 : Enable I/O bus error signals
= 1 : Disable I/O bus error signals
PB6 = 0 : Disable keyboard clock
= 1 : Enable keyboard clock
PB7 = 0 : Port A reads keyboard scan code
= 1 : Port A reads SW1 (old PCs), or...
= 0 : Keyboard enabled (XT's and clones)
= 1 : Keyboard acknowledge

Port C

PC0-PC3 : determined by PB2
if PB2 = 0 : Low nibble reads top of SW1 or SW2
= 1 : Low nibble reads bottom of switch
PC4 = Data from cassette tape (old PCs) or...
= Output from speaker (XT's and clones)
PC5 = Output from Timer 2
PC6 = 1 : I/O bus error
PC7 = 1 : RAM parity error
(note: if both PC6 and PC7 are low during NMI,
the interrupt source is the 8087)

(IOW) signals. An LS243 then drives these signals out to the smart chips. The combination of either RD or WR along with an active CS lets the 8088 read from and write to the PPI.

The RESET pin responds to a high input by clearing the control register and setting all ports to input.

PPI Tasks On The PC

So what does the PPI do to earn its keep? Quite a bit it turns out. (See Figure 2 for a complete list of 8255 port usage.)

Old PCs use input Port A to perform one of two functions, depending on the value of PB7 (Port B, bit seven). If PB7 equals zero, Port A accepts scan codes from the keyboard. A PB7 of one forces the PPI to read the first of two system configuration DIP switches (SW1).

Newer PCs, XT's, and clones use Port A for keyboard input only. These machines have just one configuration switch (SW1). This difference in switch configuration between PCs and XT's illustrates the danger of reading the DIP switches directly. Use interrupt 11h instead. It does the job much more easily and reliably.

We can get into all sorts of mischief with output Port B. PB0 provides a gate signal for Timer 2 on the 8253. A low on this bit turns off the counter and prevents the 8253 from making any sound.

PB1 provides another means of fooling with sound since the output of Timer 2 gets ANDed with PB1 before going to the speaker. See *A Bleeping PC* in Issue #35 (April-May 1987) for a complete look at sound generation.

On old PCs, PB2 determines which half of the second configuration DIP switch (SW2) is seen by the low order nibble of Port C. The bottom half of this switch has to do with the amount of RAM installed in the expansion slots. The top half of the switch is designated spare. Since the newer machines don't even have SW2, you shouldn't use this method to check for RAM. Call interrupt 12h instead.

XT's and clones use PB2 in a similar manner but they look at SW1 instead of SW2.

Back in the dark ages of computing (old PCs again), PB3 controlled the cassette motor. These days not too many folks care about using cassette tape storage, so this bit is up for grabs. I use it to switch clock rates on my sped-up clone.

XT's can use PB3 to *attempt* to disable

Figure 3 - PPI Control Port

Bit 0 = 0	: Low order nibble of Port C is output
= 1	: Low order nibble of Port C is input
Bit 1 = 0	: Port B set for output
= 1	: Port B set for input
Bit 2 = 0	: Port B is Mode 0
= 1	: Port B is Mode 1
Bit 3 = 0	: High order nibble of Port C is output
= 1	: High order nibble of Port C is input
Bits 6,5 = 00	: Port A is Mode 0
= 01	: Port A is Mode 1
= 10 or 11	: Port A is Mode 2
Bit 7 = 0	: Set/reset individual bits in Port C
= 1	: Program PPI ports

the keyboard. But this depends on how the user has set jumper E5. So you can't depend on PB3 for this function. Also, depending on the system, this method may allow the keyboard buffer to fill anyway. Then, when you reenable the keyboard, all those nasty keystrokes will come back to haunt you.

PB4 and PB5 enable or disable RAM parity and I/O bus error signals, respectively. A low on PB6 disables the keyboard clock. We'll use PB6 later on in a function which turns off the keyboard. As stated above, PB7 controls the behavior of Port A for old PCs. New machines use it strictly as a keyboard acknowledge signal.

The low order half of input Port C

causing problems. PC7 and PC6 allow the handler (a section of code in the ROM BIOS) to differentiate between parity, I/O, and 8087 errors.

PPI Programming On The PC

The PC uses the 8255 in its default mode (mode 0). So all 24 pins are set for I/O with no handshaking or interrupt capability. Let's look at the control register to see how the individual ports get programmed. (See Figure 3.)

Bit zero controls the direction of Port C's low order nibble. We want Port C to read one of the DIP switches, so bit zero gets a one. Bit one controls Port B's direction. A zero here allows the processor to output the control signals

Figure 4 - Keyboard Disable/Enable Function

```

PROCEDURE KeyboardOff (KbdOff: Boolean);
CONST
  PPIb = $61;           { PPI Port B address }
BEGIN
  IF KbdOff
  THEN { turn off PB6 to kill Kbd clock }
    Port [PPIb] := Port [PPIb] AND $bf
  ELSE { turn on PB6 to enable Kbd clock }
    Port [PPIb] := Port [PPIb] OR $40;
END; { KeyboardOff }

```

reads either SW1 (XT's and clones) or SW2 (old PCs) a nibble at a time. PB2 selects the nibble.

PC4 feeds cassette data to the processor in old PCs. Again, cassettes are pretty useless so this bit would be a candidate for any project that needs input to the CPU. XT's listen to the speaker output with PC4. I'm not sure what use this serves (other than keeping the PPI and the CPU entertained).

PC5 watches the output of Timer 2 before it gets ANDed with PB1.

When things go all to hell and a non-maskable interrupt (NMI) occurs, the NMI handler needs to know who's

described above. Bit two sets Port B's mode. This port is constrained to either mode 0 or mode 1. We'll reset the bit to 0 for mode 0.

Port C's high order nibble gets its direction from bit three. Setting bit three to one allows the 8088 to input NMI and Timer 2 information. Port A's direction comes from bit four. A one sets this keyboard/configuration port to input.

Port A requires two bits to specify its mode since it may operate in any of the three available modes. We'll send bits six and five a 00b to select mode 0. A 01b sets mode 1, and either 10b or 11b means mode 2.

Finally, a one in bit seven means that we're programming the PPI ports for mode and direction. Resetting bit seven to zero causes the PPI to use the low order nibble of the control byte to tweak individual bits in Port C. Again, this capability is used for handshaking and interrupts—*verboden* on the PC.

So, sending a control byte of 10011001b (99h) to I/O port 63h sets up the PPI for operation in the PC.

Power On Test

During power on self test (POST), the BIOS uses the PPI in a different manner.

During the first four checkpoints (BIOS checksum, Timer one (memory refresh) verification, DMA test, and memory test), Port A is an output port. The BIOS loads Port A with the checkpoint number (1 - 4) prior to each test. Failure of any of these four tests leads to an immediate HLT (halt) instruction.

At this point the intrepid repair per-

son can examine the status of Port A directly with a logic probe. The value in Port A shows how far the system got before it puked. A very handy diagnostic since during the first four checkpoints, the system doesn't know how to beep, much less write error messages to the screen.

Some PPI Utilities

So you've put up a "Don't fool with me, I'm thinking" message on the screen, but you know that some (ab)user will bollix up the works by leaning on the keyboard at an inopportune moment. It would certainly be nice to be able to kill the keyboard from time to time.

Figure 4 shows a routine which takes advantage of PB6 to shut down the keyboard. We could just as easily have used the keyboard acknowledge bit (PB7), but turning off the clock signal to the keyboard seems a more obvious method. Since this is our "C" issue, I

chose Pascal for the examples.

KeyboardOff won't take any crap from the user—not even Control/Alt/Delete. The operator's only recourse is something that plugs into the wall.

Anyone taking over interrupt 9 (keyboard) to install a hot key for a resident program will be dealing directly with the PPI. See the PC speedup article in issue #31 (August-September 1986) for a simple-minded example of the process. Also, as mentioned above, you can use PB1 for sound generation.

System Configuration

The PPI is instrumental in determining system configuration, although we don't have to look at it directly. Interrupt 11h does all the work for us. Figure 5 shows a function which performs this task.

As an aside, on power up the BIOS checks out the switches and performs other tests on the system. It then loads the results of these equipment checks into memory locations 0:410h and 0:411h. So you could examine these equipment flags directly rather than use interrupt 11h. But that's not the *approved* method and far be it from me to counsel inappropriate behavior.

Further notes on interrupt 11: It's limited. It *has* been handy for determining which video card is installed. And it seems reliable in returning the number of floppy drives. But it's unclear if you can get good information about the math coprocessor. It probably depends on your BIOS.

The reported number of printers and RS-232 cards is suspect as well. Again, it seems to be a function of the BIOS. Dave recently got a hold of a very interesting disassembler from V Communications called Sourcer. (See his Comdex article in this issue.) I'll be using it to look at the various BIOSes we have at Micro C. If I come up with anything, I'll let you know.

We're Almost Done

Well, that wasn't nearly as bad as I'd expected. We even got some useful routines out of it. Next time around, I'll wrap up this series with a close look at the brains of the outfit, the 8088.

Editor's note: Assuming there really are brains in this outfit.

We'll also take a step back from the small details and try to tie all these smart chips together into a coherent whole.

◆ ◆ ◆

Figure 5 - Check Installed Equipment

```

TYPE { standard register declaration }
ReqRecord = RECORD CASE Integer OF
    1: (AX, BX, CX, DX, BP, SI,
        DI, DS, ES, Flags: Integer);
    2: (AL, AH, BL, BH, CL, CH, DL, DH: Byte);
END;

PROCEDURE CheckEquip;
{ uses int 11h to determine equipment installed }

VAR
    Regs: ReqRecord;
    Temp: Integer; { makes things easier to read }

BEGIN
    Intr ($11, Regs); { do equip check interrupt }
    WITH Regs DO { result returned in AX }
        BEGIN
            Temp := AX AND 1; { any disk drives? }
            IF (Temp = 1)
                THEN Writeln ('Got some drives')
                ELSE Writeln ('No drives');
            { bit 1 not used }
            Temp := (AX AND $0c) SHR 2; {how much RAM}
            Writeln (Temp+1, ' block(s) RAM'); {old PCs}
            Temp := (AX AND $30) SHR 4; { video }
            CASE Temp OF
                0: Writeln ('EGA');
                1: Writeln ('40 column CGA');
                2: Writeln ('80 column CGA');
                3: Writeln ('80 column mono');
            END;
            Temp := (AX AND $c0) SHR 6; { # of drives }
            Writeln (Temp + 1, ' drive(s)');
            { bit8 senses DMA controller on PCjr only }
            Temp := (AX AND $0e00) SHR 9; {# serial ports}
            Writeln (Temp, ' RS232 port(s)');
            Temp := (AX AND $1000) SHR 12; {game port?}
            IF (Temp = 0)
                THEN Writeln ('Game port')
                ELSE Writeln ('No game port');
            { bit13 senses serial printer on PCjr only}
            Temp := (AX AND $c000) SHR 14; {# printers}
            Writeln (Temp, ' printer(s)');
        END; { WITH Regs }
    END; { CheckEquip }

```

HALTED

SERVING NORTHERN CALIFORNIA
SINCE 1963!

SPECIALTIES

WE'VE MOVED!

PLEASE COME AND VISIT OUR NEW ENLARGED LOCATION!

KURTA SERIES ONE GRAPHICS TABLET

INCLUDES:

- MANUAL
- CORD
- BASIC PROGRAM LISTING

\$299⁰⁰
With Stylus

\$349⁰⁰
With Puck

AT/XT 4 SLOT MOTHER BOARD EXPANDER

INCLUDES:

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

\$69⁹⁵

—IC PROGRAMMERS— XT/AT COMPATIBLE

ALL USE SINGLE SLOT IN YOUR XT/AT AND INCLUDE
EXTERNAL PODS WITH ZERO INSERTION FORCE
SOCKET(S)

PAL PROGRAMMER *269.95

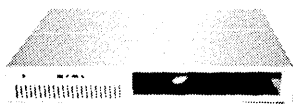
—PROGRAMS MOST 20 & 24 PIN PALS
—VERIFY, PROGRAM & BURN SECURITY FUSE LINK
8748/49 PROGRAMMER *199.85

—EPROM PROGRAMMERS—

PROGRAMS 27XX to 27512
12.5, 21, 25V DEVICES INCLUDES SOFTWARE FOR
STANDARD & INTEL HEX FORMATS

SINGLE GANG *109.95
FOUR GANG *169.95
TEN GANG *309.95

IF YOU DON'T SEE WHAT YOU WANT, CALL US!



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
- 60 Watt P.S. w/DC Fan and Drive Cables

\$89⁹⁵/\$29⁹⁵
complete chassis only

HALTED SPECIALTIES offers its customers
Unique "SUPERMARKET STYLE" shopping
for ANY and ALL ELECTRONIC NEEDS. We
stock literally THOUSANDS of parts— from the
newest IC's to some of the first transistors.
ALSO, for the electronic enthusiast, we carry a
full line of computer accessories, test
equipment, tools, R+D supplies and much
more! PLEASE CALL OR VISIT ONE OF OUR
THREE RETAIL STORES TODAY!

XEBEC 1410 SASI HARD DISK CONTROLLER

\$89⁰⁰

YOUR COMPLETE ELECTRONIC SUPPLY STORE!

LITTON TRACKBALL

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

\$49⁹⁵

• NEW!! TI 787 Terminal —

- 120 CPS Thermal Printer
- 110 to 9600 BAUD MODEM: Bell 212, 103 + Racal 3400 Compatible.
- Acoustic/Direct Connect
- RS-232 Interface, Includes Manuals, Cables & Paper.

\$ 249.00

ANDERSON/JACOBSEN Stand-Alone MODEM— • 1200 Baud Racal 3400 Compatible • Acoustic/direct Connect • Originate Only • INCLUDES MANUAL + CABLES.....

\$ 29.95

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

MAIL ORDERS

Call NOW! (408) 732-1573

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

WE SHIP C.O.D.

TERMS: Minimum order \$10. California residents add 7% sales tax. Prepaid orders send freight COD or call for charges. Shipping will be added to credit card and COD 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

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!

V COMMUNICATIONS

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

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

Reader Service Number 62

The Dream-88

30 Mb XT Compatible

XT Turbo, 4.77-8 MHz motherboard
640 K of on board RAM
1 5.25" 360K Fujitsu floppy drive
30 Mb Seagate ST-238 Hard drive
Floppy controller card (controls 2)
Hard disk controller card
12" amber Samsung monitor (tilt/swivel)
Monographics card with parallel port
AT style keyboard
XT slide case (UL and FCC approved)

Complete! \$795⁰⁰

The Dream-286

40 Mb AT Compatible

AT 6-10 MHz 0 wait state motherboard
1 Megabyte of on board RAM
1 5.25" TEAC 1.2M floppy drive
40 Mb Seagate ST-251 Hard drive
Hard/floppy disk controller card
12" amber Samsung monitor (tilt/swivel)
Monographics card with parallel port
AT style keyboard
AT case (UL and FCC approved)

Complete! \$1495⁰⁰

All systems can be customized to your exact specifications.

Everex 1200 Baud Modem	\$ 95
Everex 2400 Baud Modem	\$ 195
Magnavox Multimode EGA Monitor	\$ 567
Genoa Super EGA High Res (800x600)	\$ 275
Micropolis 71 Mb Hard Disk	\$ 849
Mountain 40 Mb Tape Backup Unit	\$ 455
Toshiba 720K 3.5" Floppy Drive	\$ 135
Ventura Desktop Publisher	\$ 465
Word Perfect Version 4.2	\$ 239
Norton Utilities (Advanced)	\$ 89
Turbo Basic	\$ 65
Turbo Pascal	\$ 65
Turbo C	\$ 65
Citizen 120D / 180D Printer	\$ 189 / 215
Citizen MSP40 / MSP45 Printer	\$ 379 / 495
Citizen Overture 10 page/min laser (laser printer includes toner)	\$ 1595

DreamTech (408) 996-2373

5175 Moorpark Avenue San Jose, CA 95129

Reader Service Number 16

MICRO CORNUCOPIA, #40, Mar-Apr 1988 41

C Vs. Assembly Language

The Choice Is Obvious

This article came with a short note from the author: "You may wish to point out that I'm a little prejudiced. I'm the author of the A86 assembler package, reviewed on page 66 of your Nov-Dec '87 issue (#38)."

Duly noted. But, considering his background what else can he say? A lot. Read on and you'll have a much better understanding of what your C compiler (or any compiler) does for you. And to you.

What follows is a memo I wrote to a client (two years ago). At that time, he was considering translating one of my assembly language programs into C. He thought coding the program in a high-level language would improve it.

One of my main arguments in the memo was the lack of usable debugging facilities for Xenix-based C. In today's MS-DOS environment, the situation is not so grim: good debuggers such as Codeview and Periscope make C more attractive.

I mentioned that VisiCalc and Lotus 1-2-3 were coded in assembly language. That was based on hearsay.

I've since had the chance to look at part of 1-2-3 with a debugger; the part I inspected was apparently generated by a high-level compiler. I'd appreciate hearing from anyone who knows what percent of VisiCalc and 1-2-3 were written in assembly language.

The Memo:

It is my opinion that 8086 assembly language is superior to C.

I realize my opinion runs against conventional wisdom, and I shall do my best to point out theoretical, practical, and historical flaws in this common "knowledge."

Practical Examples

Ask any knowledgeable person to name the two most successful programs in the history of microcomputer software. They are likely to name VisiCalc and Lotus 1-2-3; probably the two most money-making applications programs ever written.

Both programs are complex and sophisticated—perfect examples of code that must be written in high-level language. Yet their authors chose to write substantial portions in assembly language. Were they misguided in doing so? I think not.

Lisa

In 1979 Apple Computers was buoyed by the success of its Apple II computer in the business market (due mostly to the success of VisiCalc). Apple appeared to be in a perfect position to dominate the market before IBM gained a foothold.

So Apple set about building the ultimate business computer; and they were going to do it right. This meant coding every line in a high-level language. So they hired hundreds of programmers who dutifully wrote in Pascal, the darling high-level language of the day.

Years later, and years late, the computer came to market.

Lisa featured state-of-the-art hardware. The microprocessor was the fastest; its instruction set was the best for high-level language. Yet the software, after taking years to develop and debug, was abysmally slow. Lisa became the Edsel of the computer industry; and IBM, aided by a plethora of independently-developed programs like Lotus 1-2-3, took over the business market.

Fairly late in the Lisa project, some Apple engineers, seeing the impending Lisa disaster, started a new project. With

a small fraction of Lisa's manpower, in a fraction of the time, they developed the Macintosh. What is the major difference between Lisa and Macintosh? The Macintosh's operating system was written in assembly language.

Direct Testimony

In 1977 I was assigned to create the first 8086 assembler, Intel's ASM86. I was part of a two-man team, to which a third man was added six months later. Ninety-eight percent of the assembler's source was PL/M. The rest was assembly language. It took us 33 man-months to write that assembler.

Some years later, on my own, I wrote another 8086 assembler. I wrote it in assembly language and it took me six months. (Actually, I worked on it on-and-off for about a year.)

I admit there were other factors in this quick development: I changed the assembler a bit to make it easier to write, and I had experience. But coding in assembly was as important a speed factor as any.

My assembler is one-thirteenth as large (program size) as Intel's and it's over ten times as fast. Also, surprisingly, it contains fewer lines of source code, even factoring in the product simplifications. (How? See the theoretical section.)

Text Editors

I have written four text editors in my career. Two were written in high-level language and two in assembly language. The assembly language versions took the least time to develop.

Interestingly, the assembly language versions had more features and more lines of source. When I removed the code for the additional functions, I found that the assembly language programs contained fewer lines of source code.

Theoretical Arguments

Let's perform a point-by-point comparison between C and 8086 assembly. We'll weigh their advantages and disadvantages with respect to speed, reliability, ease of development, and maintainability.

Porting To New Systems

One of C's strengths is software portability. As long as the compilers are compatible, the computers don't even need the same CPUs. This is an extremely powerful advantage, especially for programs intended to be maintained and run on many different computers. In the case of the text editor, however, that advantage is not so clear.

Editor's note: Remember, this is a letter written to a client...

First, my C program contains many calls to the Xenix operating system, as well as direct calls to the serial ports.

Second, you are porting to an IBM-PC, which has a compatible assembly language. The only changes you'd need to make for assembly language are the same ones needed in C. The job may actually be easier in assembly since I isolated the code that was hardware and operating-system dependent.

Structured Languages

C is a block-structured language. As a descendent of Algol, C is optimized for the clear expression of flow control.

Back in the days when Algol was developed, this was a monstrous advantage. Machine instruction sets were very primitive, so assembly language was unbelievably clumsy at flow control (and, for that matter, at almost anything else).

But CPU architecture has made giant strides. The following 8086 features make assembly language competitive with high-level language for clarity of program structure:

- Subroutines: The hardware stack, together with the CALL and RET instructions, make the subroutines and functions as straightforward as in high-level languages. The wealth of machine registers makes it easy to pass parameters and return results.
- Conditional branching: The 8086 flags register, together with an especially rich set of conditional branch instructions, make condition-testing obvious and straightforward. Single instructions test each of the relations =, !=, >, <, >=, and <=, for both signed and unsigned numbers. Old machine languages required long sequences to make the same tests.
- Rich register set: The 8086 has a large set of general-purpose registers, with complete symmetry for the most common arithmetic, logical, and data-movement operations. The skilled 8086 programmer will use the registers as local variables, so the manipulation of those variables is usually as easy to program as it is in high-level language. A single accumulator handled all the arithmetic for the older processors, so most lines of code shuffled values into and out of the accumulator. (And the program's function was hard to follow.)

Language Improvements

In addition to improvements in the machine architecture, there have also been important advances in assembly language itself.

The 8086 assembler has strong-typing of symbols. This means that the kind of machine instruction generated by a line of source code can be determined using the types (byte variable, word register,

I have written four text editors in my career. Two were written in high-level language and two in assembly language. The assembly language versions took the least time to develop.

immediate constant, etc.) of the operands. Old assembly languages required a different mnemonic for each type of operand (e.g., MOV_B for a byte-move, MVI for the move of an immediate constant into a register, etc.). This makes assembly language code more closely resemble high-level language code.

Historical note: The designer of the first 8086 assembly language wanted it to resemble high-level language even more closely than it does. He proposed an in-fix notation, with the equals-sign as a valid synonym for MOV (for example, MOV AL,6 would have been coded as AL = 6). This notation was vetoed by management and marketing.

I wonder if they vetoed the idea to make assembly's similarity to high-level language less obvious. (At the same time, Intel was saying that assembly language would be-

come obsolete within a few years.)

8086 assembly language has adopted one of the most valuable and wonderful of high-level language features: the recognition of long symbol names—up to 32 characters. This really improves program clarity. You can use complete English words for procedure and variable names. Symbols in the old assembly languages were limited to six characters, forcing gross abbreviations (PRTCHK, ENEIOT, etc.).

Ironically, the situation has been reversed: our Xenix-based C compiler limits symbol names to seven characters. (More precisely: longer symbols are allowed, but they must differ in the first seven characters. The limit is seven, not eight, because the compiler prepends an underscore to all user symbols.)

Of course, this limitation of one C compiler extends to all C compilers (if you want portability). As far as I am concerned, this single restriction renders C useless for significant projects.

Local Labels

My assembly language features local labels. These are redefinable labels (L1, L2, R1, etc.), used as place-markers for control-mechanisms (loops, branches, etc.). Without this feature, you must invent a different name for every minor place-holder. The really significant names become lost in the sea of irrelevant place-names. With this feature, the symbol table becomes as clean and uncluttered as it is in high-level language; and the identity of the control mechanisms is obvious.

Library...

C comes with an extensive library. This is C's greatest strength when compared to all languages, not just assembly language. It makes C a good choice for drafting short programs.

This advantage disappears, however, when the program becomes large and complex. The library's usefulness stays fairly constant as the program grows, so it becomes a smaller percentage of the total effort.

I designed my assembly language for easy linking of source modules with library routines. Once you've coded a routine (admittedly nontrivial) you can continue to use it by simply including its source in the list of files to be assembled.

Coding Time

C claims it needs fewer lines of source than assembly to create an equivalent program. C also claims that

development time is relative to the number of lines of code. Thus, theoretically, it takes less time to develop C programs.

While I accept these claims on average, I have frequently found them untrue. For short programs, the claim of fewer lines of C code is certainly true (because of C's library). For longer programs, we must consider the number and power of primitive constructs in each language.

C has 28 built-in keywords and 43 punctuation marks (or sequences of 2 marks).

My 8086 assembly language has 149 built-in keywords (not counting 20 keywords that are synonyms for other keywords) and 15 punctuation marks. Thus, in terms of primitive tokens, my assembly language is richer than C by a count of 164 to 71.

How do we relate these counts to lines of code? We must consider how efficiently each language makes use of its tokens and the kinds of tasks the language is optimized for.

In C you can squeeze a lot of instructions onto a single line. But that can be a false advantage. The more code we stuff into a line, the more time it takes to develop and debug that line.

Token Strengths

C uses its tokens primarily for expression evaluation, flow control, and pointer manipulation (the auto-increment and auto-decrement features). Its primary advantage lies in run-time expression evaluation (understanding formulas expressed in algebraic notation).

8086 assembly language uses its tokens primarily for allocation of scratchpad variables (i.e., the registers) for low-level flow control, low-level manipulation of 8 and 16-bit quantities, and assembly-time evaluation of algebraic expressions. Also, the 8086's string instructions approach (but don't quite match) the efficiency of C's pointer manipulations.

The efficiency of C versus assembly language depends, therefore, on the type of task.

The screen editor spends most of its time manipulating text bytes—buffering them, filtering them, displaying them, etc., and 8086 assembly language is well-suited for this. Since there is almost no run-time evaluation of algebraic expressions, C holds no advantage.

Experience

C has one advantage, however. Most

programmers know a lot more about C than they do about assembly. So they seldom take full advantage of their assembler. Top programmers, such as the authors of VisiCalc, Lotus 1-2-3, and the Macintosh, use all of their assembler's features, and write less code.

Another important factor in speed of program development is not the language, but the abilities of the debugger.

Debuggers

Our C compiler has an effectively useless debugger, so the task is done by inserting print statements into the code and recompiling. This might work for small programs, but it's hopelessly slow and hopelessly difficult for something as complicated as a screen editor.

How does a newcomer, unfamiliar with 5000 lines of C code, find out where to insert print statements? Without some sort of single-stepping facility, there's no chance he or she will understand how the program works, let alone where a problem might be.

My assemblers include a debugger, optimized for stepping through a program while watching registers and memory change through window-type displays of the machine state. It's only slightly more cumbersome to run a program from the debugger than it is to run it directly.

Compilation Vs. Assembly

Compilation is significantly slower than assembly. Slow compilation adds significantly to development time and contributes to the "if-it-works-don't-touch-it" philosophy.

High-level compilers also generate atrociously inefficient code. As memory becomes cheaper, the size of programs becomes less important. But speed of execution remains as important as ever.

There is a harsh, inescapable reality: microcomputers are slow and they are likely to remain slow, because improvements in the hardware will be taken away by multitasking.

I have seen claims that compiled programs are as little as 30% slower than optimized assembly language programs. Such claims are ludicrous. I have never seen a large, high-level language program that I could not make ten times faster by recoding in assembly language. The factor of improvement is often a hundred times, or more.

To appreciate just how bad compiler-generated code is, just grit your teeth and examine it. Let's look closely at the very first lines of program code com-

plied for each of the programs driver, init, options, opts, phantom, sconfig, and uconfig. They are as follows:

```
#define IOPMEM "/dev/iopmems/8a1X0X"

zopen(c)
char c;
{
    char iopmem[30];
    strcpy(iopmem, IOPMEM);
    iopmem[strlen(iopmem)-1] = c;
    if ((z80=open(iopmem, O_RDWR)) ==
        -1) {
        printf("can't open device\
                %s\r\n", iopmem);
        perror("error");
        system("stty -raw echo\n");
        exit(1);
    }
}
```

zopen opens a device that allows access to the Z-80 memory within the 986 system. The name of the device depends on the number of the user (for whom the device is being opened). Thus one must take the user number c and plug it into the device name. I do the same thing in my assembly language screen editor; it is coded as follows:

```
IOP_NAME: DB '/dev/iopmems/8a1X0'
IOP_USER DB '*',0

; (parameter c has already been
placed in the AL register)

        MOV IOP_USER,AL
```

In assembly language the act of placing the user number into the device name takes precisely one machine instruction. In C, things are a bit messier.

C frowns upon the modification of pre-declared arrays, so we declare a local array, and call the function strcpy, which copies the string IOPMEM to the local array. Note that IOPMEM is defined as a macro—every usage of IOPMEM within the program will generate a distinct, identical copy of the 19-byte file name, plus the 1-byte terminator.

We shall generously ignore the (terrible) code generated to allocate the local array and copy to it, and concentrate on the single line of C code:

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

which duplicates a single assembly language instruction.

Here we have a curious situation. 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 this case, the device name). We therefore use the run-time function strlen, which calculates the length of the string. The last element lies at length minus one.

Let's hold our noses and look in Figure 1 to see the code generated by

that single line of C.

If STRLEN had been skillfully coded in assembly language, it would replace the six-instruction loop with a single REP SCASB. The length would have been calculated at the end, by subtracting pointers.

Further, notice the tremendous overhead involved in getting into and out of every procedure call.

My final count of machine instructions executed for the single line of C code is 153, versus 1 for assembly language. The example of this line, while worse than average, is not atypical.

A Lazy Language

All those easily-accessed library functions can be very seductive; the programmer stops bothering to find the

Figure 1 - Code Generated By A Single Line Of C.

```
LEA DI,[BP-34] ; point DI to our local copy of device name
PUSH DI ; in C parameters are pushed onto the stack-you
; can never just leave them in registers, that would
; be too easy
CALL STRLEN ; set AX to the length of the device name
ADD SP,2 ; the 8086 has a special 'RET n' instruction that
; allows procedures to pop their own on-stack
; parameters, but C never generates this instruction;
; instead, it generates this ADD after every CALL to
; a function with parameters!!
MOV DI,AX ; spurious move, why not leave it in AX?
LEA SI,[BP-35] ; here we see that our compiler is actually cleverer
; than many—it has combined the pointer to the
; local array with the coded '-1' at compile time!
ADD DI,SI ; now DI points to the last byte of the array
PUSH DI ; spurious PUSH, why not leave the pointer in DI??
MOV DL,[BP+4] ; fetch the parameter 'c' from the stack
POP BX ; if we were hell-bent on moving DI to BX, why PUSH
; then POP, why not just MOV BX,DI??
MOV [BX],DL ; finally, the byte is placed into the array!!!
STRLEN:
PUSH BP ; always save BP, whether we need to or not
MOV BP,SP ; BP is set up for addressing parameters, whether there
; are any or not
PUSH DI,SI ; save DI and SI, whether or not they are clobbered
SUB AX,AX ; AX=0 for no local storage (coded MOV AX,nn if the
; procedure does have on-stack local storage)
CALL CHKSTK ; insure the stack is NOT running rampant
;-----
SUB SI,SI ; again—we are very clever and have register
; variables, an advanced compiler feature
JMP L2 ; jump into loop, this is also a clever little
; optimization
L1:
INC SI ; increment the return length count
L2:
MOV BX,DI ; spurious move, why not index DI directly??
INC DI ; string pointer increments
MOV DL,[BX] ; spurious load, we could have tested [BX] directly
TEST DL,DL ; semi-clever—I would have expected CMP DL,0 here
JNE L1 ; loop if the terminating zero byte has not been found
MOV AX,SI ; move answer to AX, it could have been there in the
; first place
;-----
JMP CRET ; 3-byte jump instruction to common return-point
CRET:
LEA SP,[BP-4] ; release local storage (not to be confused with the
; release of parameters, the caller must do that)
POP SI,DI,BP ; restore the possibly-clobbered registers
RET
CHKSTK:
MOV BX,SP ; load our stack pointer into BX, to check it
SUB BX,AX ; decrement, to add local storage to the stack
JB L9 ; error if SP has fallen through the floor
CMP BX,LIMIT_VAR ; have we fallen below our limit?
JB L8 ; error if yes
POP AX ; pop our return address into AX
MOV SP,BX ; revise SP, local storage is now on the stack
JMP AX ; this JMP effects a return from CHKSTK
L8:
L9: ; ... code not executed if the program is correct
```

most efficient way to perform a task.

The aura of laziness extends to the design of the C language itself. Many features built into other languages are library functions in C: Figure 1 for example, or "printf". These functions make a bare-bones language usable, but program efficiency suffers. C conceals the price with its terseness.

For instance, Donald Knuth has written probably the definitive books on computer science. Their central theme is the analysis of algorithms, with the goal of finding the most efficient way to perform a given task.

To enable the precise measurement of his algorithms' efficiency, Knuth invented his own language, and presents in his books the coding of all his algorithms in that language. It is, of course, an assembly language, for his mythical MIX computer.

Editor's note: This, of course, makes Knuth the original MIX master.

If assembly is the antithesis of decent, structured code, then why does Knuth force his readers to plow through all that assembler? Why didn't he invent something like Pascal or C to code his algorithms?

The answer is that high-level language removes the programmer from the real machine. It conceals from him the final code (no matter how terse the source.)

In assembly language, however, a one-to-one correspondence exists between source code and machine instructions. Terse code is, almost by definition, efficient code.

Inflexibility Of C

Yet another disadvantage of C, and of other high-level languages, is inflexibility of the compiler:

- Procedures and parameter passing. The rich 8086 register set makes it rarely necessary to pass parameters on the stack. Yet C always does this. It makes assembly language interfacing to C very tedious.
- Memory usage. My C compiler assumes that all data fits into a single 64 Kbyte segment.
- Flow control. Since the assembly language programmer has complete control, he can invent controls that drastically reduce program size and increase speed. For instance, in my assembler I created an error-escape mechanism similar to the EXCEPTION feature of the language Ada. The

mechanism virtually eliminated the messiness of error recovery. It could not have been done in any of the high-level languages available at the time.

Terse

C is an unusually terse language. So terse that cryptic, incomprehensible code becomes easy to write. The Kernighan and Ritchie C "bible" gives some examples (with a tone of approval!) of atrocious C programming technique.

Many high-level languages have an advantage over assembly language, in that bad programmers find it harder to write rock-bottom-bad code. But in C, I would have to call it a standoff.

Summary

C is a good language for short and medium sized programs. Its library functions assist development of such programs. When the program gets large, however, the lack of a debugger, the short symbol names, and the inefficient code make C a poor choice.

I have been tempted to write a C compiler with a built-in "expert system" for generating efficient 8086 code. It would perform global optimizations (instead of "peephole" optimizations); and it would use registers optimally just as a human expert would. But I haven't done it, and I haven't seen another compiler come close.

Meanwhile, assembly language has been subject to undeserved abuse. Sure it takes more skill to write good programs in assembly language. But the best assembly language programs are incomparably better than the best high-level language programs. They match high-level programs in reliability, readability, and speed of development. And they far surpass high-level programs in execution speed and code size.

◆ ◆ ◆

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 Software Publishing Corporation.

Reader Service Number 26

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)

```

TEXT LINE: 15 COL: 16 FILE: PHOTO .203 INSERT ERL
WINDOW 0
/* Main loop - displays the ma
do {
  scrlines = SCRINES;
  scrwidth = SCRWIDTH;
  clrscr(scrlines-20);
  show(main_menu);
  ret_val = getrange(mm_pro
  processf ret_val, (new_ved
  ) while (ret_val != EXIT_OK)
}
if (new_vedit && (table in !=
  printf crt_sel);
  if (yesno(" ")) setcrtf ar
  else outcrlf());
}
-WINDOW $
WINDOW 1
VEDIT PLUS is an advanced editor that
makes your program development and word
processing as efficient and easy as
possible. VEDIT PLUS is simple enough to
learn and use for the novice, yet has the
speed, flexibility and power to satisfy
the most demanding computer professional.
VEDIT PLUS is particularly suited for
writing all types of programs and lengthy
documents such as reports or manuscripts.
.sp 2
This shows how VEDIT PLUS can perform
windowing. One window is used for word
processing, a second for program
development, and the third for commands.
Up to 40 windows are supported and you
determine each window's size and color.
DIRECTORY C:\VEDIT\NEW
COMPARE .VDM CV203 .VDM MAIL .VDM MENU .VDM PRINT .VDM
SORT .VDM STRIPV .VDM Z80-8086.VDM

```



#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	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec

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

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.

CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821
Reader Service Number 7

Turbo Pascal 4.0

Borland's Flagship Comes of Age

This is a special piece. It's special because Borland has added very significant new features to Turbo Pascal, features which make it a viable language for large projects. It's also special because Jim has extensive experience working with Pascal and working on large projects. Here is his reaction to version 4.0.

If you are like me, and millions of other Turbo Pascal users, you have been anxiously waiting for the "new and improved" version. Though the programming environment of earlier versions of Turbo Pascal revolutionized software development, those versions had some significant limitations.

Now Borland has come to our rescue. They began shipping Turbo Pascal 4.0, the IBM PC version, during the first week of November. The upgrade price is \$39.95 plus \$5 shipping and handling. They have even extended their upgrade offer to owners of CP/M versions. Now you can upgrade your old CP/M version to the new PC version. The suggested retail price is \$99.95.

My copy of Turbo 4.0 came on three disks, in contrast to one for Turbo 3.0. "New and improved" always seems to mean larger, doesn't it?

The first disk contains two copies of the compiler; the integrated environment version and a command line only version. It also contains the README file with a few corrections to the 654 page manual.

The second disk contains some utility programs to help you manage large projects and the "upgrade" program to help you convert your Turbo 3.0 programs to Turbo 4.0. Yes, convert. (I'll cover conversion of Turbo 3.0 programs a bit later.) You will also find several coding examples of Turbo

4.0 programs.

The third disk contains source code for MicroCalc, Borland's freebie spreadsheet program that has been completely revised for Turbo 4.0. The Graphics interface routines are also found on this disk.

What's New?

Turbo Pascal 4.0 sports a ton of new whistles. I can't take the time (or the space) to cover all of them in appropriate detail, so I will highlight a few of the important ones. The following is a pretty complete list:

- Integrated development environment
- Project management tools (Build, Make, Pick list)
- Separate compilation of program parts (UNITs)
- Library of standard UNITs
- Symbolic debugger interface
- Faster software-only floating point
- IEEE standard reals (if you have a numeric coprocessor)
- Faster compiler
- Smaller object code (hardly visible)
- Faster object code
- Device-independent graphics
- Additional data types including "LongInt" and "Word"
- Optional range and stack checking, and short-circuit boolean expression evaluation
- Variable and value type casting
- Nested include files—up to eight levels deep
- Interrupt procedure support—allows you to write Interrupt Service Routines
- Assembly language interface (External)

Separate Compilation

Probably the most important new

feature is the compilation of program parts.

Imagine finding one bug in your 9,365 line checkbook-balancing program. Think of the wear and tear on your disk drives, and your patience, just to recompile all 9,365 lines.

Now, imagine making that one-line change to one of your nine source modules and having Turbo compile it and link it in under 13 seconds.

Wow!

Other compilers have supported modular compilation for years, but not to this degree. Let's take a closer look at this "module" concept as used by Turbo Pascal 4.0

UNITs And USES

To understand the Turbo 4.0 version of "modules," we have to look at the UNIT. The UNIT was first implemented by the University of California at San Diego, in their UCSD Pascal.

A Turbo 4.0 UNIT is basically a collection of data definitions and procedures compiled together and stored on disk as a .TPU (Turbo Pascal Unit) file. Each UNIT contains its own code segment that may be up to 64K bytes long.

By combining segments you can create a program which can use up to 640K of memory.

A Turbo Pascal UNIT has two parts, much the same as a Modula 2 module:

(1) The INTERFACE part containing the constants, types, variables, and headers—all of the things other UNITs will need to see.

(2) The IMPLEMENTATION part containing internal declarations as well as the actual procedures and functions. The IMPLEMENTATION part may also contain some UNIT initialization code.

In Figure 1, I've declared a UNIT named MinMax. When compiled, the resulting code will be placed in a file called MINMAX.TPU.

Figure 1 - Sample Turbo 4.0 UNIT

```

Unit MinMax; (Declare Unit)

INTERFACE      {Interface section}

Const
  Min = 10;
  Max = 20;

Var
  Total : integer;
  Error : boolean;

Procedure ChkMin(value : integer);
Procedure ChkMax(value : integer);

IMPLEMENTATION Var
  MaxCnt : Integer;
  MinCnt : Integer;

Function TotCnt : Integer;
begin
  TotCnt := MaxCnt + MinCnt;
end;

Procedure ChkMin;
begin
  MinCnt := MinCnt +1;
  Error := False;
  If value < Min then Error := True;
  Total := TotCnt;
end;

Procedure ChkMax;
begin
  MaxCnt := MaxCnt +1;
  Error := False;
  If value > Max then Error := True;
  Total := TotCnt;
end;

begin {Initialization section}
  Total := 0;
  Error := False;
  MaxCnt := 0;
  MinCnt := 0;
end. { End of Unit MinMax }

```

This UNIT has declared, in the INTERFACE section, the constants Min and Max, the variables Total and Error, and the procedures ChkMin and ChkMax. These are all (globally) available to any program or UNIT using MinMax.

In the IMPLEMENTATION section I declare variables MaxCnt and MinCnt

and the function TotCnt. These are not available outside the UNIT.

The coding at the end of the UNIT, appearing as the main body of the UNIT, is the initialization section. It's optional but if present, it will be executed before the main program. This lets you initialize variables or poke at the hardware before main program execution.

A UNIT is included in a program or another UNIT through the use of the new statement type, USES.

The USES statement is placed at the front of the program or UNIT in the following manner:

```

Program CheckBook;
Uses Crt, MinMax;

```

or

```

Unit Balance;
Uses Crt, MinMax;

```

In the above examples, the program CheckBook and the Unit Balance both use two UNITS. The INTERFACE sections of the Crt and MinMax UNITS will be included in the program CheckBook and the UNIT Balance as if they had been declared within the program or UNIT.

Note that a UNIT is not an include file. A UNIT is separately compiled and stored on disk in binary form, ready for the compiler to reference whenever it sees a USES.

Turbo Pascal 4.0 contains a library of pre-defined UNITS. The UNIT names and their functions are:

- System—This UNIT contains all of the standard UNITS and functions of Turbo Pascal. System gets linked into every program.
- Dos—This UNIT contains all of the most commonly used DOS calls, GetTime, SetTime, Disk-

Size—as well as the low-level routines MsDos and Intr. The low-level routines handle direct calls to MS-DOS or system interrupts.

- Crt—This UNIT provides PC specific I/O routines for screen and keyboard (windowing, direct cursor addressing and the like).
- Printer—This UNIT declares the LST variable and links the Write and Writeln procedures to the printer driver routines.
- Graph—This UNIT provides a set of fast, powerful graphics routines and implements the device-independent Borland graphics handler. You get graphics support of CGA, EGA, VGA, Hercules, AT&T 400, MCGA, and 3270 PC graphics hardware.

The folks at Borland have been kind enough to provide two additional "compatibility" UNITS for those of us who already have drawers full of Turbo Pascal source. They are:

- Turbo3—This UNIT contains two variables and several procedures not supported by Turbo 4.0. These include the file variable Kbd, the boolean variable CBreak, and the integer versions of MemAvail and MaxAvail, which return the number of free paragraphs instead of free bytes.
- Graph3—This UNIT supports all of the Turbo 3.0 graphics functions (including Turtle Graphics).

Managing Libraries

Borland has included a utility program for creating and managing libraries of UNITS. It is called TPUMOVER. This utility helps you create or add to existing .TPL (Turbo Pascal Library) files. TPL files are collections of UNITS. For instance, the standard UNITS reside in TURBO.TPL.

As you can see, the UNITS and USES of version 4.0 are a major departure from the Turbo Pascal that we have come to love (and curse).

Project Management

Now that you have learned how to use UNITS, and you have decided that breaking your large projects into workable pieces is a great idea, you have one more problem: How do you keep track of all those pieces? How do you remember which ones to recompile after a long edit session? Not to worry. Turbo 4.0 will remember for you!

Make and Build help you manage your large programming projects.

The Make option performs three checks when you compile a program.

First, Make compares the date and time on all of the .TPU files to the .PAS files for all of a program's UNITS. If the .PAS file has been modified since it was last compiled into a .TPU, then it's recompiled.

Then Make checks to see if the programmer has modified the INTERFACE portion of the recompiled UNIT. If the INTERFACE has changed then Make recompiles all the other UNITS that use the modified UNIT.

The third check is to see if changes have been made to any include files or any .OBJ files (assembly language routines) used by any UNITS. If so, all UNITS using these files are recompiled.

The Build option is a special case of the Make option. When you use the Build option to compile a program, all its UNITS are also compiled.

Borland also threw in a stand-alone Make utility. You use it when you're doing batch mode project management.

The New Environment

Turbo Pascal 4.0 has a new look which strongly resembles the Turbo C package. Borland calls it the "Integrated Development Environment." I call it super.

The new environment features two main windows—one for the editor and one for capturing program output during a RUN. Either window can zoom to fill the whole screen. The screen is handled in one of three modes: Mono or CGA mode with 25 lines; EGA mode with 43 lines; or VGA mode with 50 lines.

Borland included several pull-down menus for controlling the compiler—such features as: the math method, paths and directories, memory model...

(You set the defaults during installation.)

The editor's default command set remains faithful to the older versions of the compiler (and therefore, to WordStar). In fact, the editor is so compatible with Turbo 3.0 that it still has the old file length limitation! Come on guys! When are you going to finish this fine upgrade?

The editor also maintains a Pick-File list. This is a pop-up menu containing a list of names for the last eight files that you have edited. When you reenter a file contained in this list, the editor remembers where you were and what conditions you had established in that file. In other words, the editor puts you back into the file at the point where you left, with block markers and so on intact.

Notice the "conversion comment" immediately following the old compiler option {\$C-}. UPGRADE changed the directive into a comment by inserting a space between the curly bracket and the dollar sign {\$C-}.

In this case, the "conversion comment" indicates that the "C" directive is obsolete in Turbo 4.0. (The KeyPressed function works all the time now!). This converted program compiles without further editing. You can find the inserted conversion comments by searching for the string {! in the source.

Overlays?

Turbo 4.0 does not support Overlay Procedures but before you panic, remember what I told you about UNITS. UNITS replace overlays.

When you use the /U option with

Figure 2 - Pascal Program Before Conversion By UPGRADE

```
Program Numbers;
{$C-} {Make KeyPressed func work!}
const
  ms = 10; {delay in milliseconds}
var
  x, y, col, att,
  char, color : integer;
begin
  clrscr;
  repeat
  begin
    x := random(80)+1; {col indx}
    y := random(25)+1; {lin indx}
    col := random(16);
    att := random(8);
    char := random(10000);
    if ((y = 25) and (x >= 74))
      then x := 70;
    gotoxy(x,y);
    textcolor(col);
    textbackground(att);
    write(' ',char,' ');
    delay(ms);
  end;
  until KeyPressed;
  clrscr;
end.
```

Conversion Of 3.0 Source Code

The Turbo 4.0 package includes a source code conversion program called UPGRADE. This package is remarkably easy to use and does a very good job. In many cases, the output from the upgrade program will compile without any further changes. In other cases, minor modification is required.

Compare the Turbo 3.0 program listed in Figure 2 to the converted version listed in Figure 3. Notice that six lines of compiler directives were inserted at the beginning of the file. UPGRADE also inserted "Uses Crt;"

UPGRADE, it will break your source code into UNITS. This is not, however, a completely automatic feature. You must first edit the original source code to insert special comments for UPGRADE. These comments are of the form {U unitname}.

When Upgrade encounters one of these special comments, it will break the source code at that point, creating a new UNIT with the name you specified. UPGRADE automatically deletes "overlay" keywords and, if you specified /U, it'll add USES statements.

Figure 3 - Pascal Program After Conversion By UPGRADE

```

{$R-} {Range checking off}
{$B-} {Boolean short circuiting off}
{$S+} {Stack checking on}
{$I+} {I/O checking on}
{$N-} {No numeric coprocessor}
{$M 65500,16384,655360} {Turbo 3 default stack and heap}

Program Numbers;

{ $C-} {Make the KeyPressed function work!}
{!^ 1. Directives A,B,C,D,F,G,P,U,W,X are obsolete or changed in
meaning)

Uses
  Crt;

const
  ms = 10; {delay in milliseconds}

var
  x, y, col, att,          char, color : integer;

begin
  clrscr;
  repeat
    begin
      x := random(80)+1; {col indx}
      y := random(25)+1; {lin indx}
      col := random(16);
      att := random(8);
      char := random(10000);
      if (y = 25) and (x >= 74)
        then x := 70;
      gotoxy(x,y);
      textcolor(col);
      textbackground(att);
      write(' ',char,' ');
      delay(ms);
    end;
  until KeyPressed;
  clrscr;
end.

```

Conclusion

I have been using Turbo 4.0 for about three weeks now. At first I was not impressed (the very first thing I ran into was the old file size limit in the editor). Since then I have had a chance to explore UNITS, work with the new graphics environment, and convert a large program from Microsoft Pascal to Turbo 4.0 (the difference in compile time alone was enough to win me over—Turbo took 22 seconds, Microsoft took 12 minutes!).

I can now say, without any reservations, that I think Turbo Pascal 4.0 is a great product. It's not perfect, but it's pretty close.

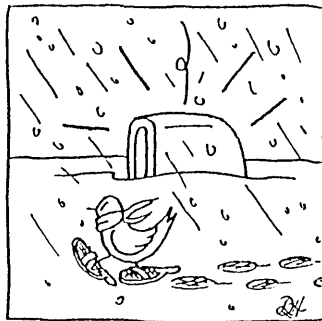
In his last article in *Micro C* #38 (November-December 1987), Ron Miller wrote, "Will C become the BASIC of the late 80's? Golly, I hope so." To that I must reply, "Not while Borland continues to improve Turbo Pascal."

This has been a very quick look at Turbo Pascal 4.0. Many of the features were covered in name only. Sorry. I hope I have at least managed to pique

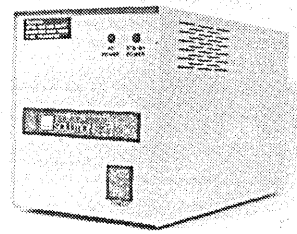
your curiosity. Maybe we'll have another chance to do some in-depth exploring in another article.

Until then, happy coding!

◆ ◆ ◆



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 5¼" 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

SCIENTIFIC SOFTWARE

SCI-GRAF \$99.95 Create huge hi-res plots with log or linear scaling. Screen and printer output. Automatic legends and labels. Flexible ASCII input. Works with CGA, EGA, Hercules, and mono cards.

SCI-GRAF MODULES \$250.00 Create custom hi-res graphs from within your own programs by linking to our object code. Supports all *SCI-GRAF* features plus plotter output, and more! Microsoft C, Turbo C, and Aztec C versions. No royalties.

FONTEEDIT \$49.95 Create custom Greek, math, or other symbols for use with *SCI-GRAF* or *SCI-GRAF MODULES*. Requires IBM compatibility and CGA or EGA.

SCI-DATA \$59.95 Perform least squares fits (linear, parabolic, and exponential) and normal curve approximations. Convert between polar and rectangular coordinates. Also supports a variety of scaling transformations. Great complement to *SCI-GRAF!*

SCI-EVAL \$49.95 Pop-up scientific expression evaluator, more powerful than other pop-up calculators. Complete expression editing facility. Full range of functions: scientific, statistical, logic. Requires IBM compatibility.

Free shipping on prepaid orders. No credit cards.

MSC Microcomputer Systems Consultants
P.O. Box 747, Santa Barbara, CA 93102
(805) 963-3412
Reader Service Number 36
CALL FOR FREE CATALOG

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.

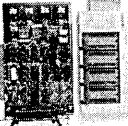


\$69.95
ASSEMBLED & TESTED

NEW! PRICE CUT!

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!

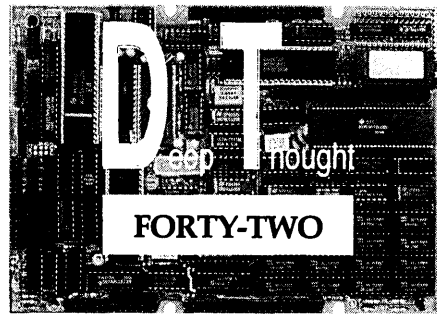
* LATEST DESIGN * PROGRAMS UP TO 4 DEVICES AT ONE TIME * FEATURES EASY TO USE MENU DRIVEN SOFTWARE THAT RUNS UNDER PC OR MS-DOS. * USES AN INTELLIGENT PROGRAMMING ALGORITHM FOR SUPER FAST (8X) EPROM BURNING. * THIS PLUG-IN BOARD ATTACHES TO AN EXTERNAL MINI CHASSIS CONTAINING 4 TEXTOL Z.I.F. SOCKETS. * 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



The "thoughtful" alternative
from *SemiDisk*.

Designed around the 64180 microprocessor, the DT-42 is loaded with *more* of all the features you need: More speed, more memory, more ports and more TPA!

How did we fit all these features on one 5.75" by 8" single-board computer?

- 9.216MHz 64180 Microprocessor (runs Z80 programs)
- 512K DRAM, Zero wait states, fully populated.
- Three RS232C serial ports (Standard baud rates to 38,400)
- One Centronics parallel printer port
- WD2793 disk controller (up to 8 drives, SD, DD or High Density, 3 1/2", 5 1/4", and 8" drives)
- SASI channel for hard disk controller (software provided)
- Many popular disk formats supported
- Requires only +5V @ 1 amp.
- ZRDOS/ZCPR3 with exclusive "Hyperspace" operating system, offering 57.5K TPA (NOT 48K like some others. *No 8 bit is bigger!!*) Richard Conn's ZCPR3, The Manual included free!
- Provisions for real-time clock and on-board terminal options.
- Socket for 28-pin EPROM.

**Compare! You won't settle for less.
Or slower. Or smaller.**

DT-42 Computer	\$ 499
TMP (on-board terminal)	\$ 100
SmartWatch	\$ 50
Z-system software	\$ 50
ZAS & Debuggers	\$ 25
8MB disk emulator w/ SCSI	\$ 2049
Battery backup for above	\$ 150

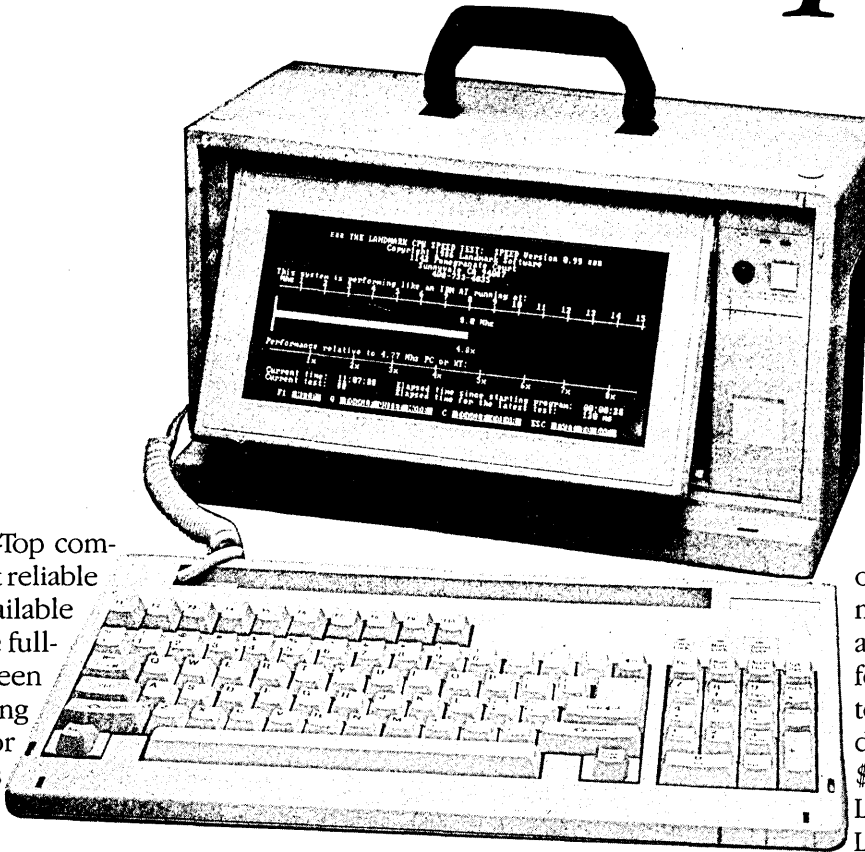
Call or write for more
information or to place an order.

SemiDisk

11080 SW Allen Blvd., Beaverton, OR 97005
(503) 626-3104

Reader Service Number 18

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 Complete 4.77/8MHz
\$659 Superturbo Complete 4.77/10MHz

McTek Systems, Inc. • 2316 4TH Street • Berkeley, CA 94710 • 415-843-0714



DISK DRIVES

Fujitsu 360k\$75
 Fujitsu 1.2MB\$99
 Teac\$79
 Teac 1.2MB\$105
 Toshiba 3½" 720k\$119
 Floppy controller\$22
 20MB Hard Disk Kit....\$289
 30MB Hard Disk Kit....\$319
 ST-225\$219
 ST-238\$249
 ST-4038\$529
 ST-251 40MB.....\$429

PRINTERS

Citizen CD 120\$159
 Citizen CD 180\$189
 HPLASAR Serial2.....\$1799
 Epson LX-800\$219
 Toshiba 321 XL.....\$559
 Call for prices of other brands

MODEMS

Easydata Int. 300/1200 ...\$79
 Taiheo external 3/12...\$105
 Everex 2400 external...\$195

MONITORS

Samsung amber\$79
 TVM EGA color\$399
 TVM RGB color\$289
 NEC Multisync\$559
 Sony Multiscan\$650
 HGC-compat.mono card ..\$55
 Color graphic card\$49
 EGA Paradise\$159

MOUSE

Logimouse C7.....\$75

PC/XT

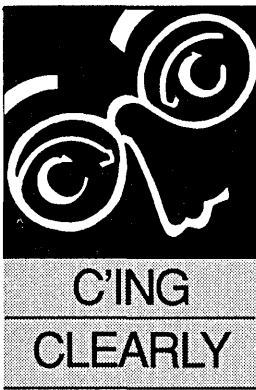
640k TurboMothrbrd....\$85
 10MHz TurboMothrbrd...\$69
 Multi I/O w/disk contrir...\$59
 640k RAM card\$39
 2MB Expansion card...\$115
 RS232 2-port card\$35
 4-serial port card\$95
 Game I/O card\$15
 384k Multifunction card ..\$69
 FCC-app. slide XT case ..\$29
 150W power supply\$55
 XT keyboard\$49

PC/AT

McTek286 6/8/10/12MHV \$289
 Baby McTek 286B-AT
 10/13 O-wait\$409
 McTek 286A O-wait 3MB
 4 ports on board\$449
 3 MB Multifunction card \$125
 2MB Expansion card...\$125
 Multi I/O card\$59
 Locking slide case\$65
 200W power supply\$79
 Enhanced keyboard\$59
 WD HD/floppy controller \$149

MISC.

Kingtech Portable Computer Kits: XT/AT (power supply, case, keyboard, monitor)\$380/\$410
 Eprom burner 4-socket \$139
 Lap-Top Kits\$799
 AC power center\$25
 AC power strips\$15
 Diskette file box\$9
 Printer or serial cable\$8
386 16MHz 2MB ..\$1599



Source Level Debugging In Turbo C

An Easy Way To Catch Creepy Crawlers

Gary L. Mellor
10100 Independence Ave.
Chatsworth, CA 91311
(818) 709-1057

It's easy to tell the difference between an experienced C'er and a beginner. The first knows the value of a source level debugger. The second soon finds out. If the introduction of Turbo C has convinced you to try your hand at this powerful language, you should stop whatever it is you're doing (such as reading this intro) and read on.

Like thousands of others, I purchased version 1.0 of Borland's Turbo C. It's easy to develop programs in the Turbo environment, but once the program is ready to execute you're on your own.

Having been thoroughly spoiled by source level debuggers on other machines, I found adding print statements and recompiling to be very tedious and frustrating.

What I needed was a source level debugger that would work within the Turbo C environment. After giving it some thought, I decided to put together my own debugger. It needed to:

- (1) Display the source listing.
- (2) Display local and global variables.
- (3) Set a breakpoint on a given source line.
- (4) Execute the program a line at a time.
- (5) Require no special code.
- (6) Not change program execution.
- (7) Provide all these functions without leaving the Turbo environment.

Fortunately, the Turbo C manual includes enough information on the load map to make most of these possible.

I could manage goal (1) by opening the source file, counting lines and displaying them. The load map relates source line numbers to their physical location (compiler option `-y`, linker option `/l`).

I could also include the addresses of global symbols in the map file (linker option `/m`), so we've covered goals (3), (4), and part of (2). (More on this later.)

Of course, I still had to take all of this information and create a debugger that met the final three goals.

Small Model

In an effort to keep the debugger as small and simple as possible, I used the "small" memory model. The easiest way to make the debugger run from within the Turbo environment is to make it part of the executed program. In this case I used a single function call to the debug initialization routine.

The call to the debug initializer can be placed virtually anywhere in the program, typically near the beginning, but it could be invoked by a special "hot key" or hidden in the startup code.

The debugger adds about 10K to the compiled program size. And, it calls common library routines like `gets()`, `puts()`, and `printf()`. A sample main program that calls the debugger might look like this:

```
main()
{
    debug_init("HELLO.C");
    printf("Hello, world\n");
}
```

The function of `debug_init` is to read the map file and set up the line number and global symbol tables. In order to do this, `debug_init` must be passed a name ("HELLO.C") so it can find the map file.

Prior to MS-DOS version 3, the name of the executable file was not passed from the command line, so the name must be passed from the main program explicitly (the identifier `__FILE__` will work). If you are using version 3 of MS-DOS, the name of the map file could be constructed from `argv[0]`, as long as it has the same root as the name of the executable file on the command line.

In order to have minimum effect on the data space of the program being debugged, I've dynamically allocated the line number and symbol tables in a far heap (using a simple linked list of far pointers).

Editor's note: Farmers will also give you pointers about far heaps. (They smell better than near heaps.)

This leaves the main program's local heap

and storage area unaffected, with the exception of a few global variables to maintain the table pointers and other necessary information.

Details

In this implementation, the debugger assumes the small model. This means that all data offsets are within the data segment and all code offsets are within the code segment. So the debugger only uses the offset portion of the symbol and line number data to set breakpoints or display variables. If you want to use the debugger with one of the other memory models, you'll have to add the segment information to the tables.

If you have multiple separately-compiled modules, Tlink (the Turbo linker) adds the line number information for each compiled object module separately in the order they appear in the project file. The debugger will only use the line numbers from the first module, and its name is listed in the map file as the source file to open. It ignores all subsequent line number lists.

If you need to debug code in another module, list that module first in the make file.

After initialization, the debugger enters the command mode and displays the prompt "TBUG?" The command line parser will accept the least unique spelling of the command (the first letter in most cases). Typing RETURN on a blank line will execute the previous command as if it had been entered with no arguments. Thus, you can step through a program (after the first step command) or list the next group of lines, without retyping the command.

Most of the commands are straightforward and are listed in Figure 1. However, the "PRINT" command has several variations which bear mentioning.

Figure 1 - TBUG Commands

```

HELP          lists the TBUG commands
MAP           displays the line number map
SYMS         displays the global symbols
LIST <line>  displays 20 source lines from line number or from last list
PRINT variable displays the current value of the variable
BREAK <line> sets a break on a line or displays current breakpoints
NOBREAK line removes a break from a line
STEP         executes the current line and stops on the next line
CONTINUE    continues execution to the next breakpoint
REGISTERS   displays the contents of the processor registers
QUIT        terminates execution of the program

```

ARGUMENT SYNTAX

line

a decimal line number referencing the source file
note: breakpoints may only be set on lines listed in the map file

variable

a variable description in one of the following forms:
<type-specifier> identifier <+/- constant>
<type-specifier> identifier[constant]
<type-specifier> address <+/- constant>
<type-specifier> _BP <+/- constant>

type-specifier

a subset of the valid C types, namely (int,char,float,double)

identifier

a variable identifier which may be optionally preceded by an '*' to indicate a pointer type identifier

constant

any valid decimal, octal or hexadecimal constant expressed in the normal C format

address

an address expressed as a hexadecimal constant

note: arguments contained in <> are optional

The "PRINT" command lets you format and print local and global data. Optional type-specifiers may be used to override the default type "int" and specify floating point or character data formats. The following command lines:

```

print float xxx
print char yyy

```

display xxx as a floating point number and yyy as a single character. The pointer operator "*" may be used to

reference pointer data:

```
print int *zzz
```

which uses the contents of zzz as a pointer to an integer value and displays that value. The command line:

```
print char *yyy
```

will display the null terminated character string pointed to by yyy. You may also specify single-dimensioned arrays by using the format:

```
print char yyy[2]
```

or add and subtract single constant offsets with:

```
print int *zzz + 2
print int *zzz - 2
```

No Locals

Unfortunately, Turbo C does not supply any information on local variables. In fact, if you do not turn register variables off (compiler option -r-), some local variables aren't even stored in local memory and are maintained in SI and DI for the entire function.

The "PRINT" command does have a mode that allows access to local variables or passed parameters using pseudo register _BP, much the same way an assembly language debugger would access them using the BP register. Let's assume the function:

```
int func(p1,p2)
int p1;
char *p2;
{
    float v1;
    int v2;
    .
    .
    .
}
```

The normal C function call method will cause p2 and p1 to be pushed onto the stack followed by the return address. The first line of the function will push BP on the stack, so after that the stack looks like this:

```
SP + 6: p2
SP + 4: p1
SP + 2: return address
SP:   old BP
```

The SP register is moved to BP and SP is decremented by 4 to make space for the locals (assuming register variables are off). The new stack, including locals, shown relative to BP looks like this:

```
BP + 6: p2
BP + 4: p1
BP + 2: return address
BP:   old BP
BP - 2: v1
BP - 4: v2
```

Given this information, locals can be displayed using the pseudo register _BP with commands like:

```
print char *_BP + 6
```

to display the string pointed to by p2, and:

```
print float _BP - 2
```

to display the floating point number in local variable v1.

Note that in the last two examples the constant value is used as a byte count, while in the earlier examples, using global variable identifiers, the constant value is sized according to the normal C conventions for pointer arithmetic.

Static variables present a more difficult problem because they do not appear in the publics list. Instead, they are stored in a common data area and referenced by offset during compilation. For this reason the debugger cannot resolve the locations of static variables, and for that matter doesn't even know they exist. If you have any static variables in your program, you will not be able to print them from the debugger unless you know where they are located in memory.

How The Debugger Works

Now that we have seen what the debugger does, let's take a look at how it works. We initialized the debugger by calling the function debug_init from the main program and passing it the name of the program. The initialization routine then begins reading the map file looking for the publics list.

Once it has found the publics list, the initialization routine passes each line to the function addsyms, which adds the public symbols to the debug symbol table. Addsyms makes sure it's adding only data symbols by checking that

each line's segment is equal to the data segment.

Addsyms allocates the space required for each new symbol entry on the far heap, saving the far pointer for the newly allocated entry in the old last entry structure. This creates a linked list to all of the symbols, starting from the first symbol. While not the most efficient way to build a symbol table, debugger searches are infrequent enough so that linear searching of the table to find a symbol is not a problem.

Once the space has been allocated and the link created, the symbol offset and name are stored in the new entry, along with a zero in the link field to indicate the end of the table. Also, in order to remain consistent with the source listing, the leading underscore is removed from the symbol name and up to 32 characters are saved to remain consistent with Turbo C's default for significant identifier length.

Creating The Line Number List

When the initialization routine encounters the first line number list (for the first module), it saves the name of the source file and calls addlines for each line of line numbers. Addlines works basically the same way as addsyms, saving line numbers and code offsets in the line table. In addition it stores the opcode in the table for later use by the breakpoint handler.

Stepping Through The Code

After passing the source filename to the command_line routine and allowing you to set up your initial breakpoints, the initializer sets up two interrupt vectors. These vectors are the key to how the debugger handles breakpoints and steps through code.

The 8088 has two special interrupt vectors. Vector #1 is the single-step interrupt and vector #3 is the breakpoint interrupt. They were built into the 8088 specifically to help us write debuggers. The breakpoint interrupt is really not any different than any other interrupt, except that it generates different code.

Normally the "INT n" instruction generates two bytes of code, the first byte is the opcode and the next byte is the interrupt number "n." The breakpoint interrupt, "INT 3," is unique because it generates a special single-byte opcode. This allows the breakpoint interrupt instruction to be substituted for any other instruction in a program, regardless of the length of that instruction. (You couldn't replace a single-byte

instruction with a two-byte instruction without also overwriting the next opcode).

In order for control to be returned to the debugger when the program encounters a breakpoint, our fancy vector #3 must be assigned to an interrupt handler. Turbo C makes this very easy by allowing you to declare a function as an interrupt handler and use the setvect function to assign an interrupt handler to a vector.

Single-Stepping

Assume we want to set a breakpoint at some location. After replacing the opcode with a breakpoint (INT 3), we execute the program. When execution reaches the breakpoint, control returns to the breakpoint handler via interrupt #3. At that point we find ourselves in the debugger's command mode.

Continuing, however, creates a slight problem; we must replace the breakpoint with the original opcode.

Back at the beginning, we saved the opcode in our table; but if we simply put back the instruction and return from the interrupt, we'd lose our breakpoint. This isn't acceptable.

You see, if the breakpoint is within a loop or in a function that gets called again, the program will not stop the next time through. We must have some method of restoring the breakpoint after executing the original instruction. This is where we learn the single-step.

The 8088 has a bit in the flag register that, when set, causes the processor to generate a type 1 interrupt (INT 1) after each instruction.

We can set this flag when we return from the breakpoint so that after executing our restored instruction we execute the single-step handler (via vector #1). This, of course, again replaces the instruction with the breakpoint, resets the single-step flag, and off we go.

When we want to step through a C program, we don't need to stop the processor on every instruction (as we would in assembly language) because one line of source may produce a lot of instructions. Instead, the "STEP" function sets a breakpoint at the first instruction from each line of source code.

More Details

In order to avoid conflict with other debuggers, like Microsoft's Debug, the command line routine restores the old interrupt vectors and returns a value of zero when there are no breakpoints set.

This tells the break handler not to set

the step flag when returning from the interrupt. If it did, the step interrupt would be serviced by whoever had that vector previously. This might confuse things a bit or even cause the system to crash.

An interesting problem arose from allocating the symbol table in far memory. I simply could not understand why the "SYMS" command would list the addresses but give blanks for all the symbol names.

It finally occurred to me that if I had a pointer, "table," to a structure allocated in far memory, the address of a character array within that structure, "table-string," was also a far address and could not be passed to the normal str... functions within the small model. (I guess I've been spoiled by Motorola 68000's and other non-segmented machines.)

As a result, I created the routine fstr, which copies a far string to local memory and returns a near pointer to it.

Finally

I hope I have provided enough insight into how debuggers work for you to start developing your own source level debugger. Even if you don't have Turbo C, these techniques, if you have line number and symbol information, can be applied to other compilers and other processors.

The Motorola 68000, for example, has similar breakpoint and step functions. With a little help (like local variable offsets, identifier types, and structure information—it doesn't yet compete with Microsoft's Codeview) this could be a super debugger.

I'm working on a program to preprocess the source files and extract the above information, but it sure would be easier if the necessary information were available directly from the compiler (hint, hint, Borland). Once you have used a source level debugger, you may wonder how you ever developed code without one; and like me, you may never again want to debug high-level languages in assembly.

Editor's note: The debugger and some sample code are available on the Micro C RBBS (503-382-7643, 300-1200-2400, 24 hrs., 8 bits, no parity) and on Micro C's Issue #40 disk for \$6 postpaid for U.S. subscribers and \$8 for non-subscribers and foreign orders. To order, call 1-800-888-8087 or 503-382-5060, or use the prepaid order form bound in this issue.



The C Store™

EVERYTHING FOR THE C PROGRAMMER

**PLUS FREE SHIPPING!
FREE SOFTWARE!**

LATTICE C COMPILER Ver 3.2 The classic DOS development environment.	\$229
MICROSOFT C COMPILER Ver. 5.0 Innovative CodeView™ debugger, "make", more.	\$269
TURBO C COMPILER Ver. 1.5 Fast, full development environment bargain.	\$69
INSTANT C INTERPRETER Ver. 3.0 Instant linking, execution and debugging! Directly link Microsoft, Lattice libraries.	\$379
C-TERP C INTERPRETER Ver. 3.0 Virtual memory support, versions for all compilers.	\$229
PC LINT Ver. 2.10 Shake out C bugs before compiling; neat!	\$99
PANEL PLUS Ver. 1.0 (w/source) Complete screen I/O development, no royalty.	\$379
WINDOWS FOR C Ver. 4.14 WINDOWS FOR DATA Ver. 2.06 Flexible, fast, high quality windowing system.	\$149 \$229
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

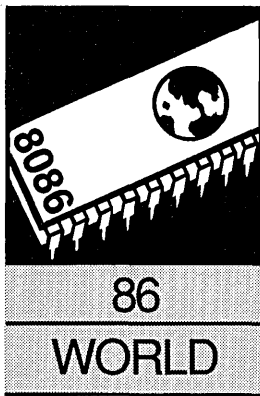
IN NEW YORK CALL:
(800) 341-1950, EXT. 889

- 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



PC Keyboards, The Real Truth

By Laine Stump

Redhouse Press
Merkez PK 142
34432 Sirkeci
Istanbul, Turkey

Laine tangles with keyboard interrupts only to discover that some of his keys can't be trained to roll over.

Every once in awhile, I come up with a few problems that really bother me and I just have to tell someone about them. The kind of problems I'm talking about are 1) software that doesn't work like you think it should, and 2) hardware that doesn't work like you think it should. In the last three months I've bought several of both. I thought I'd take this chance to warn you before you make the same mistake.

Software Incompatibility

First, software. Major complaint: Borland TSRs. Who do these guys think they are? Allah? I shelled out 75 bucks for Superkey because I needed to do some complicated and repetitive format translation with MS-WORD files. From reading the manual and all the "rave reviews," I thought Superkey would be the fastest way to get things rolling.

The package gets here and I suddenly find out that it trashes my Turkish keyboard program. Not only that, but it also trashes the foreign language keyboard programs supplied by Microsoft with MS-DOS.

That means I have to boot up a separate disk for typing and for running Superkey (Superkey can't handle the Turkish characters because it doesn't recognize the difference between "alt+c" and "alt+shift+c"). Besides that, it doesn't even have a "loop" instruction. So much for "eliminating those repetitive keystrokes."

A positive note: WORD 4.0 has built-in macro keys (with looping instructions even) which should solve my problems.

Secondary complaint: I just bought MS-Windows to set up an integrated desktop for PageMaker. The magazines have been talking about Windows 2.0 for months. What did they send? Windows 1.04, of course. And does it have a

printer driver for the printer we're using (Epson LQ-800)? No. Even the LQ1500 driver included in version 1.03 wasn't there anymore. I had to find a friend with 1.03 and copy the driver from his disk.

Hardware Incompatibility

Okay, now for hardware. First is the combination of Zenith 181 and Logitech C7 Serial Mouse. Now, don't get angry yet. ("But that's like badmouthing your own mother!")

Actually I'm quite happy with the Logitech Mouse and the Zenith 181. It's just that I specifically bought the serial version of the mouse so that I could use it on my Zenith.

After I bought the mouse and brought it all the way back to Turkey, I discovered that it worked with every machine in the house *except* the Zenith 181. And phone support is a bit expensive from Istanbul. This is more a complaint aimed at the Zenith. Apparently its serial port isn't quite IBM compatible (although it's advertised as such).

And another hardware gripe. Remember that cute little Diconix printer I talked so lovingly about? Well, as it turns out, the printer can only print the IBM extended character set when it's in IBM emulation mode, and it can only print italics if it's in Epson mode. The mode cannot be switched by software; you must turn the machine off and find a screwdriver small enough to reach in and flip the switch.

Every other "Epson compatible" printer I've seen which can print the IBM extended characters (Epson's own printers included) can print the entire 256 character set while in Epson mode. This would only take a small change in the firmware chip, and it would make the machine much more saleable in Europe.

It's not the only firmware problem with the printer either—I have a list. I thought that braindamaged firmware went out of style with Hazeltine terminals (okay, okay, that's a low blow. But somebody's got to let you know...)

Real Business

Okay, enough complaining. I promised last

issue to talk about a nice little program to allow you to easily input foreign characters from the keyboard of an IBM compatible machine. This came up in my discussion of working overseas and what you might expect to encounter there.

On my way to writing about this program, I kind of got sidetracked into talking about all kinds of keyboard things, as well as digging out another program and doing a marathon rewrite of it in C. I became so sidetracked, in fact, that I couldn't get it all organized in time. And I was nearly suffocated by noxious fumes from my kerosene heater while typing at 4 a.m. to boot. So it looks like this one is going to be a two-parter, folks.

For some of you, IBM keyboards may be ancient history (Keyboard 101). If you already know about interrupts 9 (this time) and 16h (next time), then you should just turn the page, this is all review (probably). If the lowest-level keyboard input routines you know are read() (Pascal) and getch() (C), then you'll learn a few new tricks.

When A Key Is Pressed

Before we get into the usual examples and specifics, let's talk a bit about the route that a pressed key takes through the various BIOS interrupts and DOS services before it is actually received by your program. I'll not bore you with the electrical interface (that's Larry's department—explaining the electrical characteristics I mean, not boring you). Instead, I'll start at the lowest level any POP (Plain Old Programmer) cares about.

A short primer goes something like this: Finger presses key. Keyboard sends scan code to PC. PC receives an interrupt 9 from its keyboard interface. Interrupt 9 tells the PC to read the scan

code, convert to ASCII character, and save it. Sometime later a program (or MS-DOS) calls interrupt 16h to retrieve the character.

This issue we'll take a look at interrupt 9 and see what we can do with it. (In the next thrilling episode we'll interrupt this column with 16h.)

INT 9 - Keypress Servicing

The PC keyboard doesn't work like normal ASCII keyboards. That's because it isn't a normal ASCII keyboard. Matter of fact, it's not an ASCII keyboard at all.

Instead of sending an ASCII character code when it sees a keypress, (or key-release) the PC keyboard sends a "scan code." Since every key on the keyboard has a unique press and release scan code (even the control and

board. The ESC key is 1, for instance, then comes the "1" key, which is 2, and so on. The "release" code for each key is just the "press" code plus 80h.

When a key is pressed or released, the hardware triggers an INT 9 which, under normal operation, points to the keyboard interrupt service routine in the ROM BIOS.

The ROM BIOS reads the scan code sent by the keyboard from the keyboard data port (60h) and converts it into the appropriate ASCII character, taking into account which of the shift and control keys are currently pressed (remember, I said you could keep track of the up/down state of any keys you liked).

The new character is then placed in the keyboard typeahead buffer for later use by the keyboard software interrupt, 16h (more on that later). Finally, an ac-

Figure 1 - BIOS keyboard data locations

```

BIOSDATA      SEGMENT AT 40h
ORG 17h
SHIFT_ST      DB ?
RIGHT_BIT     EQU 001H
LEFT_BIT      EQU 002H
CNTRL_BIT     EQU 004H
ALT_BIT       EQU 008H
SCROLL_BIT    EQU 010H
NUM_BIT       EQU 020H
CAPS_BIT      EQU 040H
INS_BIT       EQU 080H

ORG 1Ah
BUFFER_OUT    DW ?      ;addr of nxt empty pos
BUFFER_IN     DW ?      ;addr of nxt char removed
KEY_BUFFER    DW 16 DUP (?) ;circ buf for 16 chars

```

shift keys), a properly written program can know at any instant which keys are down.

Scan codes don't correspond to ASCII character codes at all. The keys are simply assigned an arbitrary number according to their position on the

knowledge is sent to the keyboard by flipping the highest bit of port 61h.

Keystrokes and shift key states are stored in the locations shown in Figure 1. I am showing this list for completeness of information only! Never directly use these variables unless you are com-

pletely replacing either INT 9 or INT 16.

Creating Problems

It really upsets my stomach when I see yet another example program which stuffs characters directly into the typeahead. What happens if someone decides to put the typeahead in a different place for some stupid reason (like to make it bigger)??? These guys should get a job at Borland writing great new programs like BlooperKey and Drop-Kick for every Tom, Dick, and Jerry to stuff into their Big Kat. Then we'd have that many more programs to be incompatible with.

(Laine's note: maybe I should write a TSR program called SideTrack, which would jolt the keys with 110 volts whenever it detected my writing digressing into another bitch session... Back to the subject below.)

Custom Interrupt Services

If we want, we can easily replace the standard scan code conversion and storage with whatever action we like. All we need to do is write a new interrupt service and point to it with INT 9. Figure 2 is a C program which does just that. The new interrupt 9 simply keeps track graphically of which keys are currently pressed and which are not. I originally wrote the program to test the "n-key rollover" of various brands of keyboards.

Installing interrupt vectors in Turbo C programs has been beat to death the last couple issues (and grateful I am that it was; saved me a lot of guesswork), so I won't get into those details either. I will say, however, that the C version of this program is much cleaner than the original, which was written in Turbo Pascal (when I heard that this issue was going to be a "C Issue," I couldn't resist rewriting it).

Figure 3 is VIDEO.H, the header file for my own private set of video output functions for use with Turbo C. Figure 4 is VIDEO.C, the actual code for the video routines, and Figure 5 is KBDTEST.PRJ, the project file.

C Video Package

(Speaking of video: what does everybody think about putting together a "standard video package" to be used in all examples of C code in *Micro C*? It would save a lot of pages of define_window() and clearscreen() function definitions and make it easier to follow new programs. Let's think up what routines we want and put it

Figure 2 - KBDTEST.C

```
/* KBDTEST.C - a program to demonstrate interrupt 9 (the keyboard hardware
interrupt) and test for proper operation of all keys on an
IBM compatible keyboard.

METHOD: The keyboard interrupt service vector is saved and replaced
with a pointer to our own routine which simply prints a
little message somewhere on the screen according to which
key is pressed. By doing this, the screen can give a graphic
display of which keys the computer thinks are pressed at any
given time.
If there is a difference between what is shown on the screen
and what is pushed down on the keyboard, then the keyboard
very likely has some design flaws, either in hardware or in
the firmware on the keyboard MPU.

COMPILING: Use the file kbdtest.prj as your project file, and MAKE
SURE YOU USE A LARGE DATA MODEL!! The program works under
compact and large, but it doesn't work under tiny, small,
or medium. I haven't tried huge. Be sure to compile video.c
with the same memory model.
Required Files: KBDTEST.C, KBDTEST.PRJ, VIDEO.C, VIDEO.H,
MCMVSMEM.OBJ, normal C libraries, etc.

NOTES: Feel free to use any part of this program anywhere you like,
with NO restrictions. Just remember me kindly.
originally written in Pascal, Feb. 16, 1987
translated from original into C, Dec 13, 1987
The version printed in the magazine uses normal ASCII
characters for drawing the keyboard. The version on the
Micro C. bulletin board uses IBM graphics box characters
which looks much nicer. Laine Stump, Dec 13, 1987
*/

#include <stdio.h>
#include <arg.h> /* needed for va_arg in video.h */
#include "video.h"

#define ESCLIMIT 25 /* # of consec. ESC presses to mean "exit" */
#define KEYLIMIT 83 /* higher for models w/F11 & F12 keys */

struct
{
    char message[6];
    unsigned char col, row;
} keydisp[KEYLIMIT] = {
    {"Es",18,9}, {"I!",21,9}, {"2@",24,9}, {"3#",27,9}, {"4$",30,9},
    {"5%",33,9}, {"6^",36,9}, {"7&",39,9}, {"8*",42,9}, {"9(",45,9},
    {"0)",48,9}, {"-_",51,9}, {"+=",54,9}, {"<---",57,9}, {"Tab",18,11},
    {"Qq",22,11}, {"Ww",25,11}, {"Ee",28,11}, {"Rr",31,11}, {"Tt",34,11},
    {"Yy",37,11}, {"Uu",40,11}, {"Ii",43,11}, {"Oo",46,11}, {"Pp",49,11},
    {"[",52,11}, {""]",55,11}, {"<-",59,12}, {"Ctrl",18,13}, {"Aa",23,13},
    {"Ss",26,13}, {"Dd",29,13}, {"Ff",32,13}, {"Gg",35,13}, {"Hh",38,13},
    {"Jj",41,13}, {"Kk",44,13}, {"Ll",47,13}, {";:",50,13}, {"\"",53,13},
    {"~",56,13}, {"LS",18,15}, {"|\\",21,15}, {"Zz",24,15}, {"Xx",27,15},
    {"Cc",30,15}, {"Vv",33,15}, {"Bb",36,15}, {"Nn",39,15}, {"Mm",42,15},
    {"<",45,15}, {">.",48,15}, {"?/",51,15}, {"RS",55,15}, {"PS",59,15},
    {"Alt",19,17}, {"Space",34,17}, {"Caps",54,17}, {"F1",9,9}, {"F2",12,9},
    {"F3",9,11}, {"F4",12,11}, {"F5",9,13}, {"F6",12,13}, {"F7",9,15},
    {"F8",12,15}, {"F9",9,17}, {"F0",12,17}, {"Num",63,9}, {"Scr1",69,9},
    {"Hm",62,11}, {"^",65,11}, {"Pu",68,11}, {"-",71,11}, {"<",62,13},
    {" ",65,13}, {">",68,13}, {"+",71,15}, {"En",62,15}, {"v",65,15},
    {"Pd",68,15}, {"Ins",60,17}, {"Del",66,17}
};

/* static globals */
int escctr;
void interrupt (*int9save)(); /* a place to save address of old */
/* interrupt handler */

/* the routine that will be executed for every keypress */
void interrupt int9service()
/* a basic outline: read kbd data port
act on the key
stroke acknowledge of kbd control port
signal end of interrupt to int controller
return
*/
{
    unsigned char scancode;

    /* should enable interrupts here, but I don't feel like
```

DEBUGGING SWAT TEAM

Order Eco-C88 Rel. 4.0 New Modeling Compiler
and get C-more at no extra charge!

Seek and Correct

You already know that fast compilation does not mean fast program development. Backing up for bogus error messages and removing the bugs takes time. Eco-C88's "Seek and Correct" three-way error checking finds even the most elusive bugs, clearing the path for swift program development.

Double Barrel Error Checking

Eco-C88 nails syntax errors cold and tells you about the error in plain English. And there's no avalanche of false error messages, either. Other compilers can generate up to four times the number of error messages actually present; they leave it up to you to guess which ones are real. You'll be more productive with Eco-C88 because there is no guess work.

Eco-C88 provides ten levels of semantic error checking. You can select from almost no checking to the fussiest you've ever seen. Eco-C88's "picky flag" finds subtle errors that slip by other compilers.

Eco-C88 also features:

- All data types, plus ANSI Enhancements
- Robust library, including many new ANSI functions
- CED editor with online function help, split windows, compile-edit-link capability
- New, expanded manual with sample programs for the library functions

C-more Source Code Debugger

Finally, if a really nasty bug persists, put C-more, our source code debugger, to work. With C-more you can watch your program as it executes, single-step it, set simple or conditional breakpoints, test complex expressions, use variables as indexes into other variables, initialize and trace variables, examine CPU registers, display results with printf()-type options and much more. C-more can help you track down bugs in minutes rather than days.

The price for Eco-C88 is \$99.95. And, for a limited time, we'll give you our C-more debugger at no extra charge.

Ecosoft Inc.

6413 N. College Ave.
Indianapolis, IN 46220

(317) 255-6476 (Tech Info)
(800) 952-0472 (Orders)

Memory Options

Model	Code Size	Data Size
Small	64K	64K
Compact	64K	1 Meg
Medium	1 Meg	64K
Large	1 Meg	1 Meg

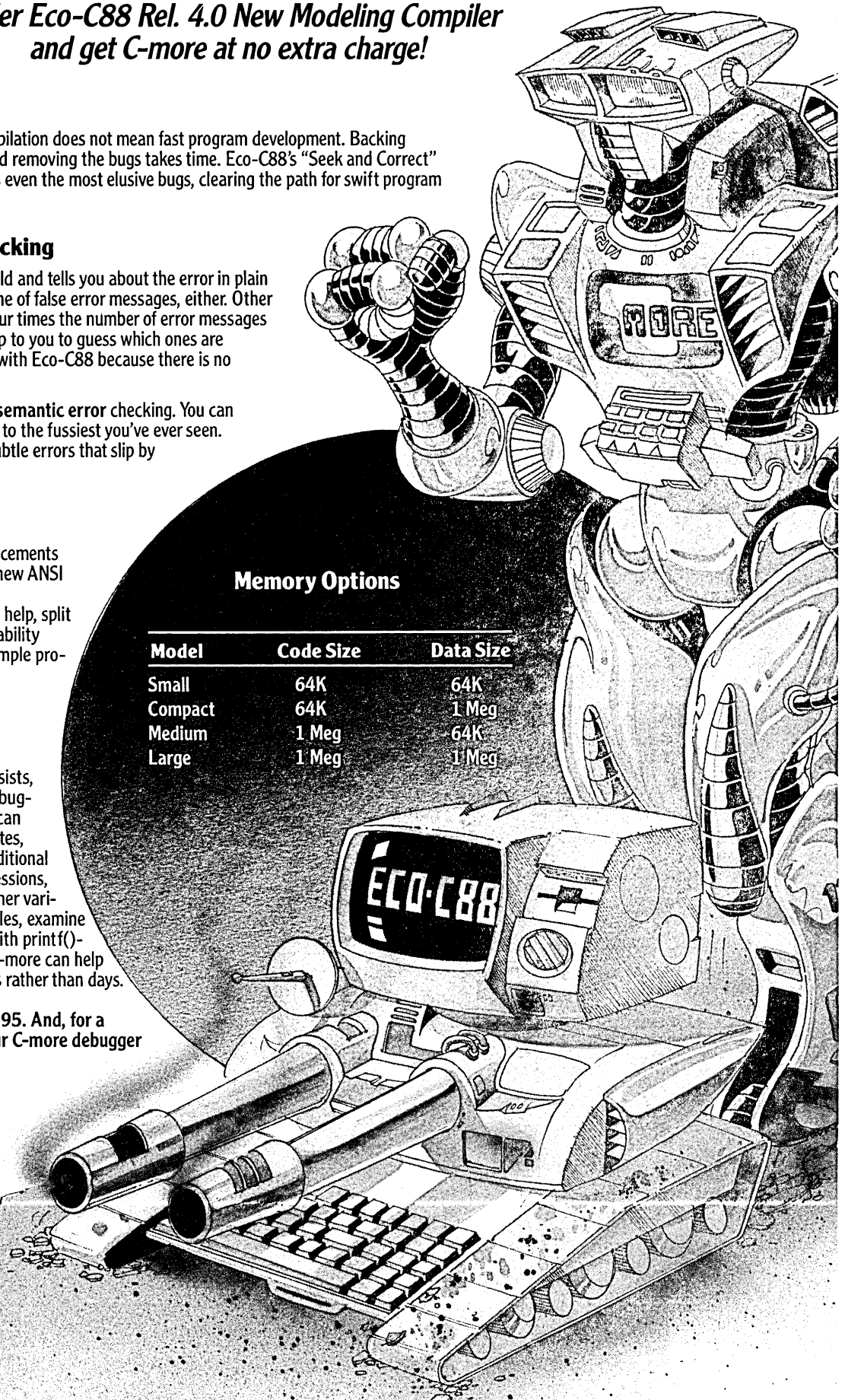


Figure 2 continued from page 60.

```

getting out the command line version of the compiler
for just one line of assembly (maybe it's already done??) */

scancode = inportb(0x60); /* get input from keyboard */
if ((scancode & 0x7F) == 1) /* check for ESC (scan code 1) */
{
    if (scancode > 0x7F)
        escctr = 0;
    else
        escctr++;
}
/* display or undisplay a key */
if (scancode > 0x7F)
    in(NORMAL);
else
    in(REVERSE);
scancode &= 0x7F;
if ((scancode > 0) && (scancode <= KEYLIMIT))
{
    scancode--; /* convert position to offset */
    at(keydisp[scancode].row, keydisp[scancode].col);
    cprintf("%s", keydisp[scancode].message);
}
else
{
    at(20, 39);
    cprintf("%2X", (int) scancode);
}
/* signal EOI to kbd & controller */
outportb(0x61, inportb(0x61) | 0x80);
outportb(0x61, inportb(0x61) & 0x7F);
outportb(0x20, 0x20);
} /* int9service */

void printkbd(void)
/* put up a graphical representation of an IBM XT keyboard */
{
    at(1, 32); in(REVERSE); cprintf(" KBDTEST v1.1 ");
    in(NORMAL);
    at(8, 8);
    cprintf("+-+--+ +-----+");
    at(9, 8);
    cprintf("|F1|F2| |Es|1!|2@|3#|4$|5%|6^|7&|8*|9(|0| |_-|=+|<---| Num | Scrl|");
    at(10, 8);
    cprintf("+-+--+ +-----+");
    at(11, 8);
    cprintf("|F3|F4| |Tab|Qq|Ww|Ee|Rr|Tt|Yy|Uu|Ii|Oo|Pp|{|}| | |Hm| ^|Pu| -|");
    at(12, 8);
    cprintf("+-+--+ +-----+");
    at(13, 8);
    cprintf("|F5|F6| |Ctrl|Aa|Ss|Dd|Ff|Gg|Hh|Jj|Kk|Ll|:|\"'|~|^| |< | 5 | >|");
    at(14, 8);
    cprintf("+-+--+ +-----+ |");
    at(15, 8);
    cprintf("|F7|F8| |LS| |\\|Zz|Xx|Cc|Vv|Bb|Nn|Mm|<, |>. |?/| RS |PS|En| v|Pd| +|");
    at(16, 8);
    cprintf("+-+--+ +-----+ |");
    at(17, 8);
    cprintf("|F9|F0| | Alt|          Space          | Caps| Ins | Del | |");
    at(18, 8);
    cprintf("+-+--+ +-----+");

    at(22, 23); in(REVERSE); cprintf(" hold down ESC for 3 sec. to Exit ");
} /* printkbd */

/*-----*/
main()
{
    initvideo();
    printkbd();

    escctr = 0;
    int9save = getvect(9);
    setvect(9, int9service);
    while (escctr < ESCLIMIT) /* wait until the signal to end */
    ;
    setvect(9, int9save); /* restore old service so we don't have */
    in(NORMAL); clearscreen(); /* to reach for the BRS (Big Red Switch) */
} /* main */

```

Figure 3 - VIDEO.H

```

/* VIDEO.H - constants and
prototypes for video
output routines
in VIDEO.C */

#define BLACK 0
#define BLUE 1
#define GREEN 2
#define CYAN 3
#define RED 4
#define MAGENTA 5
#define BROWN 6
#define LIGHTGRAY 7
#define DARKGRAY 8
#define LIGHTBLUE 9
#define LIGHTGREEN 10
#define LIGHTCYAN 11
#define LIGHTRED 12
#define LIGHTMAGENTA 13
#define YELLOW 14
#define WHITE 15
#define BLINK 128 /* add to
backcolor */
#define REVERSE BLACK, LIGHTGRAY
#define NORMAL LIGHTGRAY, BLACK

void scroll(int lines, int x1,
           int y1, int x2, int y2);
void cursorsize(int startline,
                int endline);
void clearscreen(void);
void at(int row, int col);
void in(char forecolor,
        char backcolor);
void cprintf(va_list arg_list, ...);
void initvideo(void);
void far movescreenmem(
    char far *source, char far *dest,
    unsigned len, int snowcheck);
/* from file MCMVSMEM.C (.OBJ) on
Turbo C dist. disks */

```

together, huh?)

The results of testing keyboards, by the way, were appalling. I didn't find a single Taiwanese manufactured keyboard which couldn't be made to produce alias (incorrect) keypresses when more than two keys were down at a time. If you're a fast touch typist like I am, that can be very frustrating.

For those of you interested in checking out your own keyboard, try the combinations of (left shift)+R+E as well as ctrl+W+U, W+E+(space), and A+S+F. Different brands puke on different combinations. There are other bad combinations, but these come to mind immediately.

The only keyboards in our entire office which performed acceptably were mine: the keyboard of my Toshiba T1100, the keyboard on my Zenith Z181, and my faithful old Cherry. I've given up forever on \$30 keyboards.

Are We Having Fun Yet?

Getting back to the main subject

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

Figure 4 - VIDEO.C

```
/* VIDEO.C - a few video subroutines to make writing C programs
   a bit simpler
```

a line of code for printing something will look like this:

```
at(2,20); in(RED,WHITE); cprintf("Just Like English!");
```

Of course, you can omit the at() or in() (or both) if you like.

NOTE: In order for this package to work, you must link in the file MCMVSMEM.OBJ from the Turbo C distribution disk.

Also remember that you must somehow force VIDEO.C to recompile whenever you switch memory models. This is because I haven't specifically declared all the routines and variables as 'far' (I hate using extra space and code unless I have to). If you like, you can do so yourself, then you can link the same OBJ with any memory model.

To use the following routines in your own program, just put the files video.c, video.h, and mcmvsmem.obj in your working directory, add the line '#include "video.h"' to the top of your program, call initvideo() at the beginning of main(), and make a project file that makes reference to all three files. See kbdtest.c and kbdtest.prj for examples

As usual: permission granted to use this for whatever you want to, commercial, private, obscene, or otherwise.

Laine Stump, 12/13/87

*/

```
#include <stdarg.h>
#include <stdio.h>
#include <dos.h>
#include <mem.h>
#include <string.h>

#include "video.h"

char snow, curcolor;
int currow, curcol;
char far *screen;

struct character
{
    char ch;
    char color;
};

void scroll(int lines, int x1, int y1, int x2, int y2)
/* Scrolls the screen between x1,y1 and x2,y2.
   Scrolling is up if lines is positive, down if
   it is negative */
{
    union REGS reg;

    if (lines > 0)
    {
        reg.h.ah = 7;
        lines = (-lines);
    }
    else
        reg.h.ah = 6;
    reg.h.al = lines;
    reg.h.bh = curcolor;
    reg.x.cx = (y1 << 8) + x1;
    reg.x.dx = (y2 << 8) + x2;
    int86(0x10, &reg, &reg);
} /* scroll */

int egainstalled(void)
/* returns true if EGA is installed */
{
    union REGS reg;

    reg.x.ax = 0x1200;
    reg.x.bx = 0x0010;
    reg.x.cx = 0xFFFF;
```

again (ZZAAAAPPP! OUCH!), is this where we want to put in our recognition of keystrokes meant to be international characters? Well, if you wanted to spend the time to write an interrupt handler all the way from scratch, constructing all the tables and everything, you could. It may be the best choice even. My final solution for inputting Turkish characters was to modify INT 9 (actually, I completely replaced INT 9).

I would love to show you this version of my program. Unfortunately, the sample code would be too long to put in the magazine, it wouldn't show anything that hasn't been demonstrated by the program in Figure 2, and besides I would be violating copyright laws if I showed it to you (or used it on any machine other than an X16). I admit, I used precooked code and just hacked it up a bit (thanks, Earl).

Figure 5 - KBDTEST.PRJ

```
kbdtest.c (video.h)
video.c (video.h)
mcmvsmem.obj
```

Really, though, the only reason I finally settled on INT 9 was that I thought that some incompatibility problems I was having with certain application software were caused by the application bypassing INT 16h. It turned out later that the problem was entirely different, but by that time I had already finished the new version of the program. The original, written to modify INT 16h, was much shorter and easier to understand. And I wrote it all by myself!

Tune in for the next episode when I'll give all the gory details of INT 16h, as well as some ideas on how to write programs to take the place of Superkey. Until then, please, please, PLEASE try to sleep more regular hours than I do. And remember to adjust the air intake of your kerosene heater. And don't wear wet sneakers on an empty stomach; you'll get a head cold and die of nose fungus.

◆ ◆ ◆

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
Greenleaf Data Windows (windows, menus, data entry, interactive form design)	\$315
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	\$300
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	\$275
Vitamin C (MacWindows)	\$200
resident C (TSRify C programs)	\$165
Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	\$160
Greenleaf Functions (296 useful C functions, all DOS services)	\$160
Essential C Utility Library (400 useful C functions)	\$160
Essential Communications Library (C functions for RS-232-based communication systems)	\$160
PC/IP (CMU/MIT TCP/IP implementation for PCs)	\$100
B-Tree Library & ISAM Driver (file system utilities by Sofffocus)	\$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
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	\$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	\$80
ME (programmer's editor with C-like macro language by Magma Software)	\$75
Wendin PCNX Operating System Shell	\$75
Wendin PCVMS Operating System Shell	\$75
Wendin Operating System Construction Kit	\$75
EZ_ASM (assembly language macros bridging C and MASM)	\$60
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm 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
PC/MPX (light-weight process manager; includes preemption and coroutine packages)	\$45
Biggerstaff's System Tools (multi-tasking window manager kit)	\$40
TELE Kernel (Ken Berry's multi-tasking kernel)	\$30
TELE Windows (Ken Berry's 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
C Compiler Torture Test (checks a C compiler against K & R)	\$20
Benchmark Package (C compiler, PC hardware, and Unix system)	\$20
PKG (task-to-task protocol package)	\$20
A68 (68000 cross-assembler)	\$20
Small-C (C subset compiler for 8080 and 8088)	\$20
tiny-c (C subset interpreter including the tiny-c shell)	\$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	\$20
List-Pac (C functions for lists, stacks, and queues)	\$20
XLT Macro Processor (general purpose text translator)	\$20
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	\$15

Data

DNA Sequences (GenBank 48.0 of 10,913 sequences with 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 T _E X 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 USA 78750-3409

Free surface shipping on prepaid orders

Reader Service Number 4

Voice: (512) 258-0785
BBS: (512) 258-8831
Email: FidoNet 1:382/12
MasterCard/VISA

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.

```
int86(0x10, &reg, &reg);
return((reg.x.cx == 0xFFFF) ? 0 : 1);
} /* againstalled */

void cursorsize(int startline, int endline)
/* Sets the size of the cursor */
{
    union REGS reg;

    reg.h.ah = 1;
    reg.x.cx = (startline << 8) + endline;
    int86(0x10, &reg, &reg);
} /* cursorsize */

void clearscreen(void)
/*clears screen to curcolor & puts cursor at(0,0)*/
{
    scroll(0, 0, 0, 79, 24);
    at(0,0);
} /* clearscreen */

void at(int row, int col)
/* places cursor at a specific row and column */
{
    union REGS reg;

    currow = row; /* for use by cprintf */
    curcol = col;
    reg.h.ah = 2;
    reg.h.bh = 0;
    reg.x.dx = (row << 8) + col;
    int86(0x10, &reg, &reg);
} /* at */

void in(char forecolor, char backcolor)
/* sets current color to 'color' */
{
    curcolor = forecolor | (backcolor << 4);
} /* in */

void cprintf(va_list arg_list, ...)
/* Prints a string in video memory at current loc.
in current color, then advances cursor to end
of string. This cprintf replaces the standard
cprintf in UNIX c. Is functionally identical,
except it will use color set function in(). */
{
    va_list arg_ptr;
    char *format;
    char output[81];
    struct character buffer[80];
    int counter, len;
    unsigned size;

    va_start(arg_ptr, arg_list);
    format = arg_list;
    vsprintf(output, format, arg_ptr);
    len = strlen(output);
    size = len << 1;
    setmem(buffer, size, curcolor);
    for (counter = 0; counter < len; counter++)
        buffer[counter].ch = output[counter];
    movescreenmem((char far *) (buffer), screen +
        (currow * 160) + (curcol << 1), size, snow);
    at(currow, curcol+len);
} /* cprintf */

void initvideo(void)
/* determine type of video & set screen pointer, set color to
lightgray on black & clearscreen Variable snow is needed by
movescreenmem routine contained in file MCMVSMEM.C (.OBJ)
that comes with Turbo C.
*/
{
    union REGS reg;

    reg.h.ah = 15;
    int86(0x10, &reg, &reg);
    snow = (!againstalled() && reg.h.al != 7);
    screen = (char far *) ((reg.h.al != 7) ? 0xB8000000L : 0xB0000000L);
    in(LIGHTGRAY, BLACK); clearscreen();
} /* initvideo */
```

EMERALD MICROWARE – Your CP/M and MS-DOS Connection

MicroSolutions - Software and hardware to link CP/M and MS-DOS

UniForm-PC by MicroSolutions \$ 64.95
 This program allows you to read, write, copy, and format diskettes for over a hundred CP/M and MSDOS computers on your PC, XT, or AT, including 8", 96 TPI, high density, and 3 1/2" formats (with optional hardware). Once installed, UniForm stays memory resident so you can use your standard DOS commands and other programs directly on your original diskettes.
 Uniform for Kaypro and other machines \$ 64.95

★★★ NEW ★★★ **MatchMaker by MicroSolutions** \$139.95
 MicroSolutions newest board gives you complete access to Macintosh 3 1/2" diskettes on your PC/XT/AT. Just plug your external Mac drive into the MatchMaker, and format, read, or write single or double sided Mac diskettes.
 MatchMaker w/Mac External Drive \$325.00

UniDOS Z80 CoProcessor Card \$169.95
 This 8Mhz. Z80H half-card runs 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!

The CompatiCard by MicroSolutions \$169.95
 This half-card floppy controller allows you to run up to four 8", 5 1/4"(standard, 96 TPI, or high density), or 3 1/2" disk drives on your PC/XT. With the Compaticard and the UniForm-PC program you can format, read, and write almost all CP/M and MSDOS disk formats.

Compaticard with UniFORM-PC ★★★ SPECIAL ★★★ \$225.00

MatchPoint-PC by MicroSolutions \$169.95
 This half-card allows you to read and write to NorthStar hard sector, Apple DOS, PRODOS, and Apple CP/M diskettes on your PC. INCLUDES a copy of the UniForm-PC program, as well as utilities to format disks, copy, delete, and view files.

Hard Disks for your Z80 CP/M computer from Emerald Microware and MICROCode Consulting

No other upgrade improves your computer's productivity like a hard disk. We have all the hardware and software to install a hard drive on your Xerox 820, Kaypro, Zorba, or almost any Z80 CP/M 2.2 system.

HDS Host Board with Winchester Connection software ... \$ 89.00
 HDS Board with software and WD1002-05 board \$250.00
 WD1002-05 Hard Drive Controller Board \$185.00
 Rodime, LaPine, & Miniscribe hard drives, and XT controller cards – call for prices

The KayPLUS ROM Package by MICROCode Consulting

Get the performance of a Kaypro 10 and more, even on your Kaypro 2. Lets you install up to four floppies and two hard drives, with no software assembly required. Adds features such as automatic screen blanking, type-ahead buffer, boot from hard drive, and quad density support. Includes manual, standard utilities, AND hard disk utilities

KayPLUS ROM Set \$ 69.95
 KayPLUS ROM Set with QP/M ★★ SPECIAL ★★ \$115.00

QP/M by MICROCode Consulting, CP/M 2.2 compatibility with outstanding performance.

QP/M adds features such as automatic disk relogging, drive/user selection from colon, 31 user areas, drive search path, and transparent time/date stamping; all in the same space as CP/M. Installs from a convenient customization menu, no software assembly required. Bootable disks available with CBIOS for Kaypro, Xerox, & BBI.

QP/M Operating System, complete ready to boot \$ 80.00
 QP/M without CBIOS (installs on any Z80 system) \$ 60.00

Call or write for our complete catalog of parts and accessories for the Kaypro, Xerox 820, and PC/AT's. Full repair services available for Kaypro, Morrow, Xerox, disk drives, and most clones.

★★★★★★ SUPER SPECIALS!! ★★★★★★★★

PC-Mastercard by Magnum Computer

Up to 1.5 Megabytes of RAMDISK and PRINT SPOOLER (or boost your system to 640k), with serial, parallel, game ports, and real time clock! This is one of the BEST multi-function cards on the market. Can use mixed banks of 64K and 256K chips. Comes with memory manager, ramdisk, spooler, and diagnostic software.

PC-MASTERCARD (0k installed) \$ 69.95
 call for pricing on 384k and 1.5M boards

Turbo Editor Toolbox by Borland \$ 19.95

Ever wanted to add text editing to your Turbo Pascal application, or write a word processor that does things the way that YOU want? Comes with source for two sample editors, modules for windowing, multi-tasking, and many other options. Requires PC with Turbo Pascal 3.0.

COPY II PC by Central Point Software \$ 19.95

Stop worrying about your copy protected disks. COPY II PC lets you back them up, so you can keep going when your master disk can't.

PC Tools Deluxe by Central Point Software \$ 69.95

Fantastic! Norton, Mace, Fast Back all in one.

InfoStar Manual Set \$ 18.00

Genuine DataStar and ReportStar manuals in MicroPro three vol. set

8" Generic Diskettes - Ten Pack \$ 7.50

8" Generic Diskettes - Hundred Pack \$ 65.00
 Single sided, SD or DD, with Tyvec sleeves

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

Four Device Printer/Data Switch \$ 49.95

Quality with economy. These four port boxes can be used with either RS232, or IBM parallel (DB25) printer cables.

IBM style Parallel Printer Cable \$ 12.00

300/1200 Baud Internal Modem \$117.00

Half-card, Hayes compatible, auto-answer, auto dial

CT-6260 MULTI I/O - FLOPPY DISK CONTROLLER \$ 69.95

Half-card parallel, COM1, optional COM2, game port, floppy disk controller, real time clock, with manual and cables.

TWO DRIVE FLOPPY DISK CONTROLLER \$ 29.95

Half-card floppy controller with cable.

Double Density for the Xerox 820-1 by Emerald Microware and MICROCode Consulting

Run up to four 5 1/4" (48 or 96 TPI) and 8" drives at once. Get support for all standard printers, mini-monitor functions, autoboot capability, 19 built in disk formats, and banked ROM-BIOS for more TPA. Software compatible with Kaypro and Xerox 820.

Plus2 ROM Set and X120 Board A&T \$135.00

Plus2 ROM Set and X120 Bare Board \$ 62.00

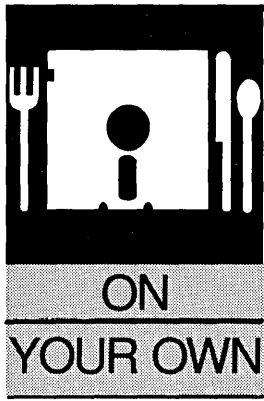
Plus2 ROM Set only \$ 49.95

X120 Bare Board only ★★★ CLOSE-OUT SPECIAL ★★★ \$15.00
 or two for \$ 25.00, five for \$ 50.00



P.O. Box 1726, Beaverton, OR 97075
 (503) 641-0347

Prices subject to change without notice. Include \$4.00 shipping and handling, \$7.00 for COD, call for Blue Label charges. VISA and Mastercard accepted. 30 day money back guarantee on all products.



Magazine Advertising, A Little Background

By David Thompson

I'm probably the best guy to write this article because I see the selling side of advertising (with Micro C) and I see the buying side of advertising (we advertise Micro C in magazines, a card deck, and direct mail).

Of course, having a magazine also makes me the worst guy to write this article. I'm biased.

OK, you have the best new product ever. Its impact on the computer world will make Intel look like just another parts house. You've sent copies to all conceivable reviewers (and more than a few inconceivable reviewers). Now you're ready to advertise. In a magazine.

Before you pick up the phone and start calling the advertising departments of the computer magazines you probably should know something about the business.

In The Past

If you'd placed your call a few years ago the conversation would have gone:

"Generic Micro advertising sales department. At the tone please leave your name, address, size of ad, and Visa Card number. Deadline is April 1. (No fooling.) Don't be late and don't forget our blue light special on quarter page verticals. Beep!"

At least that's how the conversation felt. The only additional information they'd give you was circulation. At that point publications competed on the basis of "cost per thousand." (A full page in most magazines will cost you between \$50 and \$100 per thousand readers.)

Anyway, selling cost per thousand was really selling price. (Not bad if they could adjust (double or triple) their circulations simply by making up numbers.)

The only thing that kept publications from being completely creative with their circulation figures was the once-a-year audit. (No major advertising agency would consider dealing with an unaudited publication.) Most 100,000+

circulation publications are audited and the auditors simply provide an independent verification of a publication's numbers. However, there's still room for creativity.

Back when computer magazine circulations were doubling every year they'd list next year's audit numbers on this year's rate card. When audit time came around, sure enough, the numbers matched.

The only person who was hurt was the advertiser who paid for 50,000 circulation and got only 25,000. If orders were good, he wasn't too concerned. If they were bad, he figured there was something wrong with his product or his ad.

Spring Of '84

Of course circulations can't go on doubling forever and by the Spring of 1984, they stopped growing. In fact, many declined.

By fall it was clear that the audit numbers were going to be dismal. For instance, several well known computer magazines which had advertised 75,000 circulation had declined to 25,000. That meant if they had the audit they'd be forced to refund two-thirds of their ad revenue.

Over the next six months lots of magazines folded, right before audit time. Many of the ones which didn't fold were "combined" with vaguely similar magazines so that one magazine reached guaranteed circulation levels.

That was a trying time for the body counters and it started a shift away from simple circulation numbers. And magazines quit peddling next year's circulation.

Lately, however I've been hearing a variation on the old refrain:

"We're doing a big promotion, our readership will be growing rapidly so get in now before the rates go up."

More Than Just Numbers

During all this trial and tribulation, magazines realized they'd developed unique styles and unique followings. If the style and following were in vogue then the magazine

prospered.

A few years ago *Byte* was in vogue. If you were anyone, or even thought about becoming someone, you had to advertise in *Byte*. And you couldn't stop with just one. If you dropped out for a single issue, prospective customers (and competitors) usually assumed you'd gone into something else.

If you had a classy full-color full-page ad then you were successful, solid, a giant in the industry. If you ran a quarter page black and white they assumed your management team worked nights at McDonald's and your designers were just finishing high school.

The Class Mags Now

Now things are less clear. *PC Magazine* is king, if there is one. But advertisers are a lot more practical about what they're buying.

They're looking at sales per dollar. Where those sales per dollar are best is a pretty well-kept secret. Ask *Byte* or *PC* or whomever, and they'll tell you they're the best. Ask us, and we'll tell you we're the best.

So you don't ask the magazines, you ask other advertisers. (It's OK, really. Network with other companies and you'll all benefit.)

But even with networking there'll be times when you'll be flying on hunches. The following might help your hunches:

- There is nothing more effective than the combination of a really zingy ad and a really good review.
- There is nothing less effective than a really zingy ad and a really bad review. (Should you advertise when you're not sure how the review will come out? Probably. The upside potential is huge and reviews don't happen all that often.)

Run a quarter page black and white ad and they assume your management team works nights at McDonald's and your designers are just finishing high school.

- Any editorial mention at all will generally make an ad more effective. (But beware if you can "buy" editorial mention by placing an ad. Magazines making those offers cheat their readers by selling you editorial space. They're probably not above cheating you too.)
- There aren't a lot of good independent reader surveys but you can get a pretty good feel for a magazine's audience by its content. If your product is aimed at a person like yourself, then advertise in whatever you enjoy reading. (Within reason, of course. I'm not sure *Playboy* is the place for a disassembler, if you follow my drift.)
- Ad placement is very important in magazines which aren't

thoroughly read (particularly the large ones). The major players, for instance, are very position conscious: They're either up front, on a right hand page, or on the covers. In a smaller, more intense magazine like *Micro C*, *Dr Dobbs*, *Programmers Journal*, or *Computer Language*, you're not so likely to get lost. (But you'll still pay extra for a special spot, assuming it's available.)

- Look at your ad. The clarity, the feeling, the content, and the offer are so important that there are specialists who work full-time in just these areas. However, there is one big short cut. Find an ad that's really attractive and modify it to fit your product. (Borland's ads are probably the most watched and most copied in the industry.) Think twice before dropping something from the ad (like screens or color) or the offer (like a guarantee or toll-free number). They're using those things because they work.
- You don't have to participate in the reader service game. If you're selling something that's expensive and needs lots of explanation, use reader service. Otherwise, it may not be worth contacting all those people. (I've seen small outfits go out of business because of the cost of replying to lots of uninterested people.) If they're really excited about your product, they'll contact you directly. On average, only 3 out of every 100 folks who check off the reader service form will purchase.

Get Media Kits

After you pick out a magazine you'll need to get a media kit.

Pick up the phone, dial the number for their advertising department (it's usually in the masthead):

"Hello I'd like a media kit goodbye."

The key is to hang up before they can get out their order pad. Of course, if you were too quick (not unusual on your first time) you'll have to call them back:

"Hello my address is..."

It's best to give them the number of a small post office box. (One they can't stick a salesman into.) But you forgot to mention that so you call again:

"Don't bother to send a salesman, the box is too small."

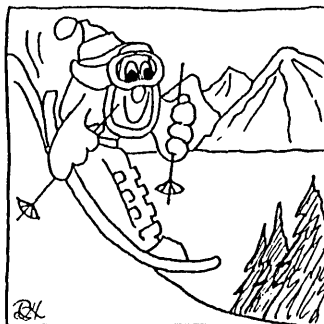
After the media kits arrive you can compare them in the privacy of your own home.

Calculate the cost per thousand. Then estimate the percentage of readers who could use your product. That gives you the true cost per thousand. The point is to make that cost as low as possible.

Of course it still comes down to sales—return for the dollar spent. If something makes money, continue it. And, hopefully you'll find it before your money runs out.

And finally, good luck. There are lots of people doing very, very well in this industry. They are the people who find their P.O. box full of checks every day. That's magic. Real, honest, modern day magic.

◆ ◆ ◆



Come to
SOG VII!
July 14 - July 16
Bend, Oregon
Rafting
Barbeque
Saturday Banquet
Registration
&
Reservations
begin
May 1st!

1- (503)-382-5060
See you in July!!

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

68000 NOW!

TinyGiant 68000 Single Board Computer

The HT68K TinyGiant is a great little 68000 single board computer. It uses only +5 and +12 volts and has all of these features: Two Serial Ports, 1 Parallel Port, Expansion Bus, 128K RAM - Expandable on Board to 512K. Floppy Disk Controller, uses MS-DOS Disk Format. 5.75" x 8.0", Fits on a 5 1/4" Drive. K-OS ONE Operating System and Software.

K-OS ONE 68000 Operating System Package

Get the K-OS ONE operating system for your 68000 hardware. With it you can read and write MS-DOS format diskettes on your 68000 system. Included in the package are: K-OS ONE Operating System Source Code, Editor, Assembler, HTPL Compiler, Sample BIOS Code.

HT68K TinyGiant with K-OS ONE . . . \$395.00
K-OS ONE Operating System Package \$50.00
HT - Forth Language \$100.00
Edit Toolkit - HTPL Source & Manual
Line Editor, Screen Editor
and Text Formatter \$50.00
Lizard Land - An HTPL Adventure Program
with Source Code \$15.00

Write or call today for more information and a free 68-KNEWS letter.

Order Now: **Hawthorne Technology**
VISA/ MC /COD 8836 Southeast Stark
(503) 254-2005 Portland, OR 97216

Reader Service Number 34

CP/M™ Software: Saving The Best for Last

Plus: The Greatest ConIX™ Giveaway

CP/M: Some people love it, others love to hate it, but most still use it. Its users complain that most software companies have abandoned it. Very true, yet we haven't! We've been selling the ConIX software line for many years; we developed it, we market it, and we support it - completely! *What?! You haven't tried it?* Saving the best for last, eh? *Don't wait!* Support your CP/M software company - try ConIX for as low as \$10! What's more, you could even get lucky and receive your entire order **FREE!** See details below.

ConIX™ Operating System

An extensive upgrade for 48K+ CP/M 2.2/3.0 and equivalent systems. Provides professional capabilities with blinding speed, as often found on high-end UNIX™ machines. Installs easily in just minutes to add over 100 new built-in commands and features while maintaining 100% compatibility with all your existing software! Includes I/O redirection, aliases, improved user area access, auto-searching, PF Keys, Screen Paging, Print Spooler, Archiver, New SysCalls, ... Eliminates many points of user frustration with CP/M. Uses only 1/2K TPA, 0-27K disk minimum.

Included **FREE** with commented source is the Pull-Down Menu System, a user-friendly interface to ConIX. Loads with a single keystroke!

ConIX is the greatest, most powerful 8-bit upgrade, with speed and capabilities that are so incredible it's bringing users *back* to CP/M!

ConIX™ Programming System

A structured programming language for ConIX extends CP/M SUBMIT capability. Adds conditionals, loops, subroutines, labels, nesting, interrupt processing, error traps, and debugging facilities. Design intricate menu systems and command-automation shells. Also includes a special source-code "compiler" that provides string and numeric variables. An absolute *must* for CP/M power-users and developers!

ConIX™ Library Vol. I XCC Utilities

Over 25 utilities for ConIX written in the shell language, including hierarchical directories with overlay - adds pathname capability to existing software, interactive debugger, move/copy/link multiple files, print files with pagination, review disk files for deletion, unerase disk with stats, full-screen TYPE, and more. Source code included!

ConIX™ Shareware Version

A new Shareware version of the ConIX O.S. includes our regular distribution software less the Archiver, On-Line Manual, Menu source code, and some satellite utilities. ConIX Shareware is available through CHI for just the cost of the diskette and shipping, or on-line via many popular bulletin board systems. Register by purchasing regular ConIX.

ConIX™ Disk Manual Version

To reduce the cost for those who want to purchase only the ConIX O.S., we are offering the complete software package with documentation provided on disk. The disk manual has each chapter stored in individual files, excluding the Chapter Summary, Chapter Reference, and Index sections that come standard in our regular typeset manuals.

The Great Giveaway: Every 100th ConIX Order **FREE!**

That's right! Every 100th order processed by our computer will be shipped with a Credit Certificate for the total purchase price or \$100, whichever is lower. This credit may be used toward a future purchase from CHI, or may be redeemed for cash within ninety (90) days of receipt. Your odds are an incredible *1 in 100!*

Offer applies only to private individuals and non-profit institutions ordering directly from CHI. Orders placed by PO or purchased for commercial use are not eligible. To enter, certify eligibility by signing order form.

Product Trademarks - CP/M: Digital Research Inc., ConIX: Computer Helper Industries Inc., UNIX: AT&T Bell Labs.

Reader Service Number 6

Name: _____
Company: _____
Address: _____

Tel. #: _____ Hours: _____

End-User Software Licenses:

ConIX Operating System	\$ 29.95	\$ _____
Disk Manual Version	\$ 19.95	\$ _____
Limited Shareware Version	\$ 0.00	\$ _____
Printed Manual Only	\$ 9.95	\$ _____
ConIX Programming System	\$ 29.95	\$ _____
Printed Manual Only	\$ 9.95	\$ _____
ConIX Library I XCC Utilities	\$ 24.95	\$ _____
Printed Manual Only	\$ 9.95	\$ _____
All ConIX Packages Above	\$ 69.95	\$ _____

Computer Brand:

Software Distribution Disk Format: *

8" SSSD Standard	\$ 5.00	\$ _____
5-1/4" DSDD 48 TPI Soft Sector	\$ 5.00	\$ _____
5-1/4" SSDD 48 TPI Soft Sector	\$ 6.00	\$ _____
5-1/4" S_D_TPI Sector	\$ 10.00	\$ _____

* Add only one format charge per order.

Format Brand:

Shipping Information:

UPS Ground USA	\$ 4.50	\$ _____
US Mail USA (For P.O. Boxes)	\$ 6.50	\$ _____
Air Mail Canada	\$ 9.50	\$ _____
Air Mail Foreign	\$ 12.50	\$ _____

Subtotal: \$ _____
Sales Tax (N.Y. Residents Only): x _____% \$ _____
Total (Thank You!): \$ _____

I Certify ConIX Giveaway Eligibility:

X _____

POs and UPS CODs accepted by phone only. Sorry, credit card payment unavailable. Personal checks require 10 days to clear. Non-USA orders must be prepaid by bank draft in US \$. Delivery in 2-4 weeks.

Computer Helper Industries Inc.
Post Office Box 680
Parkchester Station, N.Y. 10462
(212) 652-1786 9AM-5PM M-F



Small-C Projects For CP/M

Public Domain Still Lives

Stephen S. Mitchell
320 King St., Suite 506
Alexandria, VA 22314
(703) 360-4659

This is the Micro C "C Issue," so it's as good a time as any to start Stephen's two-part series on Small-C. In this issue he includes a quick (and easy) file display program.

With the current upsurge of interest in C, it's perhaps time to revisit Small-C. *Micro C* readers with long memories will undoubtedly recall Tony Ozrelic's caustic assessment of Micro C's Small-C (User disks K7 and K8).

You might also remember Fred Scacchitti's defense (Issue #23, April-May, 1985) of an upgraded version (available on disks K35 and K36). The one drawback of the new, enhanced version is that it requires Microsoft's M80/L80 package to assemble and link the compiler output. And, unfortunately, that means that running Small-C is no longer free.

The earlier version of the compiler produces output that assembles under ASM (the 8080 assembler that comes free with CP/M). And with the money you save by not having to purchase a relocating assembler and linker, you can easily afford to pick up James Hendrix's *Small-C Handbook* and Brian Kernighan and Dennis Ritchie's *The C Programming Language*. Both are essential.

A Real Tool

Despite its price, Small-C is no toy. It may lack a few features, and it won't win any awards for ease or speed of compilation, but you can write real programs with it and the resulting code will be reasonably compact and efficient. Along the way, you will learn a lot about both the C language and (if you have Hendrix's book) the operation of a C compiler.

What follows are a few (modest) examples of Small-C. These are working programs aimed at the weekend programmer (myself included). That means, barring serious blunders (and football), each can easily be

coded and debugged in a Sunday afternoon.

I assume you have compiled HELLO.C. It's covered in the file SAMPLE.DOC on disk K7. True, the program only prints "Hello, world" but don't skip it.

As Kernighan and Ritchie point out, when you try out a new language, compiler, or system, the last thing you need is a complex program. Figure out the compiler first, then tackle substantial programs.

For instance, the C function library, LIBASM.C cannot be included in HELLO.C using the #include command; LIBASM.C also contains #include statements, and Small-C does not support "nested" #includes.

SAMPLE.DOC recommends appending LIBASM.C to the source file using PIP, but there are easier ways. If your editor supports block reads (^K^R in WordStar, for example), you can simply insert LIBASM.C. Alternatively, you can let the compiler do the appending. That is:

```
A>smc b:hello.c libasm.c >b:hello.asm
```

will tell the compiler to create HELLO.ASM from HELLO.C and LIBASM.C. Most of the time you can (and probably should) avoid appending LIBASM.C.

Okay, you've compiled and run HELLO. Now you want something a bit more useful. VIEW (Figure 1, page 74) displays a text file on the screen, pausing every 22 lines. It also strips the high bits from WordStar files.

Though it's simple, the program illustrates how to open a file (unique in Small-C), how to use command line arguments, and how to declare functions. To look at textfile.txt enter:

```
A>view textfile.txt
```

If you omit the "textfile.txt," VIEW will ask you for it.

Runtime Library

The runtime library is contained in two files, CALL.ASM (containing the math and logic

routines) and IOLIB.ASM (which contains the input/output routines).

Files must be included in the correct order: STDIOA.H must be first, then IOLIB.ASM and finally CALL.ASM. STDIOA.H, contains definitions but no executable instructions. IOLIB contains an ORG pseudo-op as well as command line parsing.

If IOLIB.ASM doesn't precede CALL.ASM, neither the compiler nor the assembler will complain, but you will get an "INVERTED LOAD ADDRESS" error message when LOAD tries to create a .COM file.

File Declaration

Files in full C are normally declared in terms of a pointer to a structure, something like this:

```
FILE *infile;
```

where FILE is a typedef defined in the standard header file. In Small-C, files are referenced by a file descriptor, which is declared as an integer:

```
int infile;
```

infile contains the value returned by fopen(). fclose(), getc(), and putc() then use this value to specify the file.

First, VIEW checks the command line. "argc" tells VIEW how many commands the user entered and "argv" contains the actual entries. In full C, the syntax for declaring argc and argv is:

```
int argc; char *argv[];  
or  
int argc; char **argv;
```

This version of Small-C doesn't allow an array of pointers or pointers to pointers, so argv is declared as an integer array, like this:

```
int argc, argv[];
```

Although argv[] is declared as an integer array, you can use each value of argv[] as though it were a string. For example fopen(argv[1],"r") opens (for reading) the file whose name is the first command line argument, just as fopen("textfile.txt","r") would open a file named TEXTFILE.TXT.

Note that argc contains a number that's one greater than the number of command line arguments. As far as argc is concerned, the program name (VIEW) is 1. Add the text file to display and the argc count goes to 2.

argv[], however, counts from 0. So the program name (VIEW) is normally argv[0], and the text file to display is argv[1].

(CP/M doesn't really tell you the name of the current program, so Small-C returns an arbitrary string—in this case "PGMNAME"—for argv[0]. argv[1] will, of course, contain "textfile.txt.")

I/O, I/O

Input and output are simple. Once VIEW opens textfile.txt using fopen(), it uses getc() to read one character at a time.

After masking the high bit, putchar() outputs each character to the screen. Each carriage return in the stream increments the line count, and when the count reaches 22 the program pauses.

pause() uses BDOS function 6 (direct console i/o) rather than getchar() to read the keyboard. getchar() echoes characters to the screen while direct console i/o does not.

Since function 6 does not watch for a control-C (abort), pause() must check the entry. Otherwise, there would be no way (short of using the reset button) to abort the program.

You might wonder about the #define

NOCCARGC. This statement tells Small-C that called functions don't need an argument count. (In this implementation, only printf() can accept a variable number of arguments.)

If you don't use printf(), and don't otherwise need to pass an argument count to your functions, defining NOCCARGC will speed things up. (VIEW makes many calls to getc() and putchar().)

Also, I'm not appending LIBASM.C because VIEW doesn't need any of its functions. While appending LIBASM.C would not affect the way the program ran, it would significantly increase both the compilation time and the size of the output file.

Without LIBASM.C, VIEW.C compiles and assembles to a modest 4K .COM file. This is larger than the equivalent in assembly language, but not at all bad for a compiler.

Slimming Down The Runtime Support

The compiler can be greatly speeded up, and the size of the resulting .ASM output file greatly reduced, if you clean up the runtime support files, CALL.ASM and IOLIB.ASM.

Comments make up approximately half of each file. In addition, CALL.ASM uses spaces rather than tabs to separate fields. By eliminating the comments in both files and by replacing the spaces in CALL.ASM with tabs, you can reduce the files from 26K (total) to 17K.

You should, of course keep original copies of both files. You never know when you may need the comments.

Next Issue

Next time Stephen walks us through more Small-C utilities. Don't miss it.

◆ ◆ ◆

ICs

PROMPT DELIVERY!!!

SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN for DEC. 27, 1987

OUTSIDE OKLAHOMA: NO SALES TAX

640K MOTHERBD UPGRADE: Zenith 150,
IBM PC/XT, Compaq Portable & Plus; hp Vectra

DYNAMIC RAM			
1Mbit	1048Kx1	100 ns	\$32.00
1Mbit	256Kx4	100 ns	28.00
51258	* 256Kx1	100 ns	6.75
4464	64Kx4	150 ns	4.15
41256	256Kx1	80 ns	5.65
41256	256Kx1	100 ns	5.15
41256	256Kx1	120 ns	4.25
41256	256Kx1	150 ns	3.95
41264	+ 64Kx4	120 ns	5.25
EPROM			
27C1000	128Kx8	150 ns	\$37.95
27C512	64Kx8	200 ns	15.50
27256	32Kx8	250 ns	6.50
27128	16Kx8	250 ns	6.25
STATIC RAM			
43256L-12	32Kx8	120 ns	\$11.70
5565PL-15	8Kx8	150 ns	3.30

*STATIC COLUMN
DRAM FOR 386
COMPAQ
+2-PORT
VIDEO RAM
80387-16 80387-20
80287-8 80287-16 80387-8 80387-20
8087-2 8087-16 80387-8 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

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

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.

Reader Service Number 33

Figure 1 - VIEW, Displays CP/M text file to the screen

```

/* Written for Small-C compiler vers 2.03 (ASM)
   Pauses every 22 lines; strips high bit so that
   WordStar files will display properly on terminals
   with graphics displays.
*/

#include <stdio.h>
#include "iolib.asm"
#include "call.asm"

#define MASK 127      /* high bit mask = 0111111b */
#define NOCCARGC

int infile;
char fname[15];

main(argc,argv) int argc, argv[]; {

    int line, ch;

    if (argc == 2)
        infile = fopen(argv[1],"r");
    else {
        fputs("Name of text file to view?",stdout);
        gets(fname);
        infile = fopen(fname,"r");
    }

    if (infile == NULL) {
        fputs("File not found.",stdout);
        exit();
    }

    line = 1;
    while ((ch = getc(infile)) != EOF) {
        ch &= MASK;      /* strip high bit */
        putchar(ch);
        if (ch == CR) {
            line++;
            if ((line % 22) == 0)
                pause();
        }
    }

    fclose(infile);
}

/*
 * Pause and wait for user to hit any key.
 * Key pressed does not echo to the console.
 * Control-C will exit program.
*/

pause() {
    int c;
    while ((c = cpm(6,255)) == 0);
    if (c == 3) exit();
}

```

Letters

(Continued from page 6)

simple to set up the Priam as three logical drives with a total of 90 MB.

I have owned or installed a total of 14 Seagate ST 225s. All of them failed within one year of installation. None were bought from vendors who provide the one year warranty. Bummer. I refuse to buy or specify Seagate drives because of the 100% failure rate and because the company doesn't uniformly support a one year warranty.

I now recommend Priams because of my success and the way they are built—solid. With an RLL or Konan controller you get a lot of storage at a cheap price.

Gene Dasheill

220 S. King St. Suite 220
Honolulu, HI 96813

MicroSphere RAMdisk On A Clone?

Is it possible to install a MicroSphere Kaypro II RAMdisk on an IBM XT parallel port? And, more to the point, has anyone written the software to drive it?

An external RAMdisk with an independent power supply would be just as useful on a clone as it has proved to be on the Kaypro—especially for floppy based systems. There are probably a lot of MicroSphere RAMdisks out there gathering dust that could be put to use on clones if the software became available.

Cameron Hall

PO Box 221543
Carmel, CA 93922

Editor's note: With a little hardware tweaking you could make the parallel port listen as well as talk—an important feature unless you really want write-only memory. Take a look at Bruce's article on p.28 of Micro C Issue #38 for loads of info on the parallel port.

We haven't talked to anyone who's done this mod. If you have, let us know and we'll pass along the particulars.

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 \$1095.00

EGA UPGRADE FOR ABOVE \$ 75.00
512K UPGRADE (1024 INSTALLED) \$ 80.00
5339 KEYBOARD UPGRADE \$ 30.00

HARD DRIVES FOR XT AND AT
ST-225 KIT FOR XT (20 MEG) \$ 279.00
ST-238 KIT FOR XT (RLL 30 MEG) \$ 299.00
ST-251 FOR AT (40 MEG) \$ 425.00

MONITORS

Color Monitor RGB (CGA) \$ 275.00
Color Monitor RGB (EGA) \$ 375.00
Monochrome TTL (Green) \$ 95.00
Monochrome TTL (Amber) \$ 95.00
EGA Color Video Card \$ 129.00

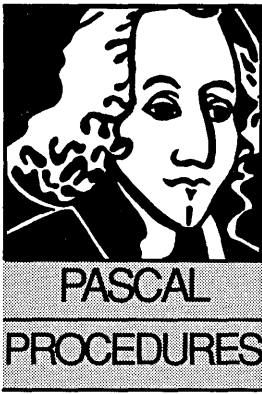
CITIZEN PRINTERS

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



Turbo Pascal 4.0 - Modula-2 Features In A Pascal Compiler

John Paul Jones
6245 Columbia Ave.
St. Louis, MO 63139

John compares Turbo Pascal with Modula-2 and finds that Pascal has matured well. Then continue on as he delves into graphics with his \$6.00 scanner.

Borland is finally shipping Turbo Pascal version 4—I got my copy just over a week ago. If you have version 3, you should have received an upgrade offer from Borland—for \$40 + shipping you can get the latest and greatest.

Version 4 is a MAJOR upgrade. Although syntactically it looks and feels like Turbo, it's entirely new.

Borland has managed substantial, though not complete, compatibility with their earlier releases. Translation from version 3 is aided by a translator program provided with the package; not perfect, but very good.

The High Points

Release 4.0 retains the integrated editor-compiler environment, but with major improvements.

Functions and options are normally accessed through pull-down menu windows, "hot" keys or function keys.

The editor retains the WordStar compatible command structure and, as with earlier versions, can be customized for both commands and screen appearance. You are still limited to about 60K of text for the source file. Since the editor can understand tabs rather than spaces, things aren't too tight.

Other than frizzled up cosmetics and conveniences in the integrated environment, the changes took place inside the compiler.

First, the compiler does not produce executable code directly, but generates linkable object code. This is then linked (automatically) with other object modules by the built-in linker to produce the executable program. The compile and link process is *fast*, in most cases you probably won't even notice the link.

The final program can be directed either to

memory for immediate testing, or to an .EXE file on disk. Compile errors invoke the editor at the point of the error.

The separation of the compiler and linker has some interesting benefits. For instance, you can maintain libraries of precompiled routines, which then need only the link step for inclusion in new programs. This is contained in the new UNIT.

To compare it to Modula-2, a UNIT looks very much like a module with both DEFINITION and IMPLEMENTATION portions in the same source file.

In a UNIT, the INTERFACE section comes first. It defines all the identifiers visible outside the unit.

The IMPLEMENTATION part follows, it contains the executable portion of the unit and can also contain "private" definitions. No EXPORT statement is needed, all identifiers in the interface section are exported. The equivalent to Modula's IMPORT statement is the USES statement.

If you change a module's interface section you'll have to recompile all the modules which call it. So, the Turbo 4 system has "make" and "build." Build recompiles all units used by your program, make recompiles only those which are affected by changes in units they import.

The built-in linker is a "smart" linker. It discards all portions of a unit not used by the importing program. Since all of Turbo's extensions, and some of the fundamental language features are contained in units, minimum program size can be very small indeed.

Going to the other extreme, since each unit can have a full 64K of code, a Turbo program can have a full 640K of code. Since the 64K limit on code was removed, Borland also removed the overlay facilities. This may or may not be the best idea; I can envision situations in which you would want to be able to have overlay capabilities even in a relatively small program.

Since the linker can also handle standard .OBJ files produced by other languages (C, assembler, Modula?), it is easy to include

modules developed outside of Turbo. Unfortunately, the compiler will *not* produce .OBJ files for use by other languages. Perhaps Borland plans for Turbo Pascal to be the primary language, with the other languages providing an assist now and then.

Graphics

I'll get into the specific language features in future columns, but I do want to say a few words about the Graph unit. The graphics support has been greatly expanded over version 3 and now includes many of the facilities which were in the Graphix Toolbox. With only a little effort, your program can run in graphics mode on any of the popular graphics cards without recompilation!

The Procedure InitGraph can automatically detect the type of graphics hardware and load the proper hardware drivers from disk. The program's output can be scaled relative to the maximum X and Y resolution (from functions GetMaxX and GetMaxY) to produce undistorted images.

First Impressions

My first impressions are very good. Turbo 4 provides the speed of both development and execution we've come to expect from Borland—the ease of learning and use of Pascal, and the ability to develop large applications in a modular fashion. I've just completed a large application which I wrote in Modula-2 (several thousand lines of source); if I'd had Turbo 4 when the project was started, I probably would have used it.

On the other hand, the lack of a source level debugger (which I've come to depend on in Logitech's Modula-2) is a significant omission. The system has a command line version of the compiler, a

separate make program which can help in updates even in mixed language environments and other utilities. Pretty impressive for \$100 retail.

Scanner Project

I'll finish up the scanner project using Modula, but you should not have too much trouble translating to Pascal if you want. I'll probably do it myself

“real soon now.”

I mentioned last time that to improve the resolution of the scanner we would probably have to aperture the sensor. Doing this is, unfortunately, not as simple as it sounds.

Since both the emitter and detector are recessed in the sensor, and also angled relative to the front face, you can't just cover the end and put a pin-

Figure 1 — Printer Driver

```

PROCEDURE StartPrinter(npoints:CARDINAL);
CONST
  (* Change constants and add or delete DOSCALLs to match printer *)
  ESC = 33C;
  L = 'L';
VAR
  I, J : CARDINAL;
BEGIN
  DOSCALL(5H,ESC); (* output graphics prefix *)
  DOSCALL(5H, L);
  DOSCALL(5H, npoints MOD 256); (* Low order byte of npoints *)
  DOSCALL(5H, npoints DIV 256); (* high order of npoints *)
  FOR I := 1 TO npoints DO
    DOSCALL(5H,0);
  END;

  FOR J := 0 TO 1 DO
    FOR I := 0 TO 10000 DO END; (* Delay for printhead to start *)
  END;
END StartPrinter;

PROCEDURE Scan (VAR R : Buffer);
VAR
  A : ADDRESS;
BEGIN
  StartPrinter(1); (* print one column, forces head to home *)
  StartPrinter(Xsize);
  A := ADR(R); (* address of where Modula needs the data *)
  SETREG(AX, 2);
  SETREG(BX, A.OFFSET);
  SETREG(DX, A.SEGMENT);
  SETREG(CX, Xsize);
  CODE (PUSHBP);
  SWI(60H);
  CODE (POPBP);

  WHILE Scanning^ DO END; (* This is a quick and dirty method.
  More elegant would be to have the resident scan
  software act as a M2 coroutine. *)
  StepPrinter;
END Scan;

```


hole in the cover. In order to eliminate the lens effect of a pinhole, any mask has to be right on the surface of the photo device.

I got best results with a small (.02") aperture on the surface of the emitter. This is most easily done with a small black donut like the ones used for printed circuit board layout. They are available many places—Radio Shack used to carry kits of them, Bishop Graphics and Datak are other sources. I found Datak .08" O.D. X .02" I.D. donuts just fit inside the front of the sensor and can be positioned with a toothpick. The package I bought for \$2 contained 800, so if you have trouble finding an equivalent send me a SASE and I'll share them.

There is one final problem with this preliminary hardware-software system, horizontal jitter. Because software timing is used to wait for the print head's return trip, even minor mechanical variations can shift the start of each scan line.

Figure 1 shows modified StartPrinter and Scan routines from the implementation module of ScrnStuff. This minimizes the jitter, but it is still present. This should probably be combined with a routine that waits for the end of a reference black area put on the left of the picture before starting the scan.

One of the features of this scanner that was obvious even before the project began is that it's *slowwwwwww*. Once you've got the adjustments made to give full scale readings (0..15 from black to white), it's most economical time-wise to just scan an image once, save the data to a disk file, and then do any image processing on the file.

There is a disadvantage to this: at 4 bits input per pixel, a full Hercules screen of 720 X 348 pixels (packed 2 per byte) is 125,280 bytes.

Figure 2 is last column's basic capture program modified slightly. It prompts for a picture file name, opens it, then as each scan line of data is captured, it gets packed into a temporary array and written to disk. Very few changes to this code are needed to read from the file and display the data on the screen. Also, the "processed" data could be read directly from the screen memory and written to disk. For that, it's not necessary to worry about the screen interleave, just blast it out to a file.

Processing

I promised I'd get into image

Figure 2 - Capture and display an image.

```
(* Capture an image to a file as well as displaying it *)

FROM ScrnStuff IMPORT Screen, ClrScr, GraphMode, TextMode, Scan,
                    PixAddress, Buffer, SetBit, SetClock, ClrBit;
FROM Terminal IMPORT KeyPressed, ReadString, WriteString;
FROM Config IMPORT Xsize, Ysize;
FROM FileSystem IMPORT File, Lookup, WriteNBytes, Close;
FROM SYSTEM IMPORT SIZE, ADR;

CONST
    TickSize = 1536;          (* real time clock chip divisor, this value gave
                              reasonable results. Subject to change. *)
    packsize = Xsize DIV 2 - 1;

VAR
    S [0b000h:0] : Screen; (* use appropriate constants for your adapter *)
    I, J, K, L : CARDINAL;
    B : Buffer;
    A : POINTER TO CHAR;
    BP : CARDINAL;           (* not used except as throwaway parameter *)
    ch : CHAR;
    byteArray : ARRAY [0..packsize] OF CHAR;
    byteidx, w : CARDINAL;
    f : File;
    fname : ARRAY [0..40] OF CHAR;

BEGIN
    ClrScr(S);                (* clear the screen *)
    WriteString('Name of picture data file: ');
    ReadString(fname);
    GraphMode;                (* put it in graphics mode *)
    Lookup(f, fname, TRUE);   (* open/create file function *)
    SetClock(TickSize);
    FOR J := 0 TO Ysize-1 DO (* just try for same res as screen *)
        byteidx := 0;
        Scan(B);              (* capture a line of data *)
        FOR K := 0 TO Xsize-1 BY 8 DO (* Xsize is bits, do byte *)
            A := PixAddress(K, J, BP); (* calculate byte address *)
            (*==>*) ch := CHR(255); (* clear assembly variable *)
            (*ch := CHR(0); (* to get white on black *)*)
            FOR L := 0 TO 7 DO (* then do each bit in the byte *)
                IF ODD(K+L) THEN
                    byteArray[byteidx] := CHR(ORD(byteArray[byteidx]) +
                                                ORD(B[K+L]));
                    INC(byteidx);
                ELSE
                    byteArray[byteidx] := CHR(ORD(B[K+L]) * 16);
                END;
            (*==>*) IF B[K+L] 17C THEN
                ch := ClrBit(ch, 7-L);
                (*ch := SetBit(ch, 7-L); (* to get white on black *)*)
            END;
            A^ := ch;          (* actual screen byte update here *)
        END;
        WriteNBytes(f, ADR(byteArray), SIZE(byteArray), w);
    END;
    Close(f);
    WHILE NOT(KeyPressed()) DO END; (* admire the pic for a bit *)
    ClrScr(S);                    (* then do orderly exit *)
    TextMode;                      (* should also SlowClock *)
END Pic2File.
```

processing this time, but there won't be room for much more than a few concepts. The simplest processing you can do with the data is to alter the contrast based on the input values.

At the places in Figure 2 marked with (*==>*), I'm doing just that. The first place initializes the assembly character for either black CHR(0) background or white CHR(255). At the other mark, the black/white status of each captured pixel is set based on its value.

Experiment, you will get significantly different displays for different values.

Dithering

No, this is not what you do when you're trying to get a better deal on a car. Dithering is a technique to get apparent gray scale from black and white dots. The idea is to work with blocks of pixels, the number of lit pixels being proportionate to the brightness of the block.

If we decide on a 2 X 2 pixel block, five brightness levels can be represented by 0, 1, 2, 3 or 4 lit pixels. (What a coincidence! The scanner provides us with five brightness values.) Of course, the image will be of lower resolution, but visually it will be much more pleasing and natural.

Now, some serious decisions. We can either reduce the number of captured data points by a factor of 4 and just randomly light the proper number of pixels, or process the larger database to provide the proper values. Although it's more work, the second alternative works better.

Image Enhancement

Another relatively simple technique

for improving a scanned image is to do a nearest neighbor analysis. (*Editor's note: This is something you should also do before moving into a new neighborhood.*)

Except for the border pixels, each dot in the display has eight neighbors. By looking closely at the neighbors (working in different directions) we can enhance vertical or horizontal edges.

I had hoped to have more done by now, but the time to do it just wasn't there. If you want to continue the project on your own, there's a good series of articles on the basics of image processing in the March, 1987, issue of *Byte* magazine.



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

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 \$99.00 - A MONEY-SAVING PRICE!

TO ORDER OR FOR ADDITIONAL INFORMATION

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

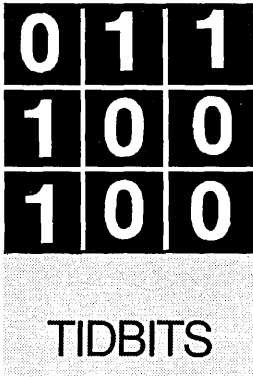
OR WRITE



PEACOCK SYSTEMS, INC.

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

Reader Service Number 20



Benchmarking Sequel And A Toolbox For TSRs

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Gary surprises us with some floating point benchmarks and reviews some very, very interesting TSR code.

Back in Issue #38 I wrote some simple statistical functions which I used to benchmark the Turbo C, Pascal, and Prolog compilers. Many of you "took the bait" and sent me your comments, compile times, and other versions of the standard deviation function. Thanks a bunch for the feedback.

From your responses, I conclude that you were genuinely amazed at the results. For those of you who didn't attend a reading of *Micro C* #38—those results are in Figure 1.

On Your Mark...

In short, Turbo Prolog (1.1) was twice as fast as Turbo C (1.0) and half as fast as Turbo Pascal (3.x).

One responder, J. Wilson (from Rawland Heights, California), noted that, "I forced the C compiler into large model by declaring the list as a stack variable." He suggested using register variables in order to move the list out of the stack (and into small model). J. was able to tweak his code down to 3.2 seconds using the Mark Williams C compiler and an AT clone (80286 CPU) running at 10 MHz.

The register variable suggestion is a good one. In Turbo C, we can specify a register variable in the code (see Figure 2) or in the compiler (by turning the register variable optimization on). Both produced identical benchmark results (2.5 seconds) on an AT clone running at 10 MHz. (Note: I ran the original benchmarks on an 80186 XT clone running at 8 MHz. The 10 MHz AT clones are much faster!)

Note also that the Turbo Prolog code runs in 1.2 seconds on a 10 MHz AT clone. So the results are still, relatively speaking, the same—Turbo Prolog is about twice as fast as Turbo C, which is a little faster than Mark Williams C. But Turbo Pascal is still faster than all the Cs

reported and tested. And version 4.0 (which is half again as fast as 3.x) is roughly twice as fast as Turbo Prolog.

But the mystery deepens even as I write. Dr. Purdum (president of Ecosoft in Indianapolis, Indiana) recently sent me the latest release of Eco-C (4.05), so I compiled my original program

In short, Turbo Prolog (1.1) was twice as fast as Turbo C (1.0) and half as fast as Turbo Pascal (3.x).

and ran it on the original (80186) XT clone. Result—12.2 seconds, or 6 seconds faster than Turbo C on that machine. And the code was only 12K (8K smaller than Turbo C's).

Jim Palmer (Optical Sciences Center in Tucson, Arizona) sent in his Turbo BASIC and Quick BASIC results, which were little short of amazing: both BASICs were better than twice as fast as Turbo Pascal (3.x).

See Figure 3 for a summary of these results.

Dr. Purdum suggested that internal representation of floating point might be part of the explanation.

Turbo Pascal (all versions) represents floating point numbers in 6 bytes (1 bit for sign, 39 bits of number, and 8 bits of exponent). Turbo C, Turbo Prolog, Turbo BASIC, and many other C's (like Eco-C) use the 8 byte IEEE standard (1 bit for sign, 52 bits of number, and 11 bits of exponent) to represent floating point numbers.

Assuming something like 2 bytes a move, Turbo Pascal saves a move every number (or a fourth the time). The rest of Turbo Pascal's secret is still open to speculation, but Turbo

Figure 1 -- Results from Micro C, #38

Compiler	Size (COM/EXE)	80186	80286	V-30 (with 8087)
Turbo C (1.0)	20K	18.2	2.5	n/a
Turbo Pascal (3.x)	12K	4.8	n/a	n/a
Turbo Prolog (1.1)	41K	9.1	1.2	n/a

Figure 2 - Two Pass Standard Deviation. Declares Reg. Vars.

```
#include <stdio.h>

void stat(double *list);

main()
{
    register short i,j;
    double list[7000];

    for(i = 0; i < 6999; i++){
        list[i]=i + 1;
        stat(list);
    }
}

void stat(double *list)
{
    register i,c;
    double x,z,dev,square,squares,var,sd;
    double sqrt(double sd);
    x = squares = 0;
    puts("start");
    for(i = 0; i < 6999; i++){
        x = list[i] + x;
    }
    z = x/i;
    printf("%f\n",z);
    for(i = 0; i < 7000; i++){
        dev = z - list[i];
        square = dev * dev;
        squares = squares + square;
    }
    var = squares/(i-1);
    printf("%f\n",var);
    sd = sqrt(var);
    printf("%f\n",sd);
}
```

Prolog's and Turbo BASIC's speed remains a mystery, since they use the same 8-byte IEEE standard as the C compilers we've tested.

Editor's note: Most people aren't aware

that some compilers use spring-loaded floats (as do some carburetors.)

And finally, both Dr. Purdum and Geoffrey Chase (at Portsmouth Abby School in Rhode Island) suggested a

faster algorithm for the standard deviation. Their method (the lemma) makes a single pass (instead of two) through the list, reducing execution time by almost half. Figure 4 contains one version of the algorithm.

Thanks guys.

I'm partial to a new series of Turbo C toolboxes released recently by Zortech.

Hotkey

I like programs that tell me how they work. The details are, well, educational, and particularly so when they include source. So I'm partial to a new series of Turbo C toolboxes released recently by Zortech.

I've seen three of their packages—a screen generator, a communications program, and a terminate and stay resident toolbox. All three come with source and with terse, technical manuals. And, they're only \$49.95 each.

I was able to—

- read most of the TSR toolbox manual
- compile & link a C program which calls assembly language code to handle the TSR details

in about an hour.

The manual begins with a short history of TSR software and then briefly ex-

plains the TSR process.

In a nutshell, programs are loaded at LOAD POINTS. Normally when a program exits it returns control to DOS, which loads the next program at the LOADPOINT.

A TSR "hotkey" program does two things:

- it changes the LOAD POINT so DOS can't overwrite it
- it filters keyboard input.

If the input is its hotkey, then it runs itself; if not, it passes the input on to DOS.

This is essentially how Sidekick and numerous other TSR programs work.

The TSR toolbox includes well-commented assembly language modules for creating TSR programs, a library of functions you need to use with TSRs, and several examples showing how to call these modules from C.

Although I haven't thoroughly tested these toolboxes, I'm impressed with what I've seen so far. The code I used worked with Turbo C and Tlink (the Turbo linker), although the manual (and README file) was a little confused about how Turbo C worked.

Zortech is both an American and European company which markets its own C compiler (in Europe) as well as these toolboxes. The toolbox manuals currently refer to the Zortech compiler, which is a little confusing since the products are intended as add ons for Turbo C.

So, if you're beginner, keep this in mind or ask Zortech for a clarification. Meanwhile, I think intermediate hackers and developers will benefit from studying Zortech code.

For more info:

Zortech
361 Massachusetts Ave.
Arlington, MA 02174
(617) 646-6703

And that, folks, is Tidbits.



Figure 3 -- Reader Results

Compiler	Size (COM/EXE)	80186	80286	V-30 (with 8087)
Turbo C (1.0)	20K	18.2	2.5	n/a
Turbo Pascal (3.x)	12K	4.8	n/a	3.9
Turbo Pascal (4.0)	6K	2.9	n/a	n/a
Turbo Prolog (1.1)	41K	9.1	1.2	n/a
Eco-C (4.05)	12K	12.2	n/a	n/a
Mark Williams (3.1)	12K	n/a	3.2	n/a
Turbo BASIC	29K	n/a	n/a	1.4
Quick BASIC	29K	n/a	n/a	2.2

Notes: 80186 computer is running at 8 MHz;
80286 computer is running at 10 MHz;
V-30 computer is running at 8 MHz;
Times for 80286, V-30, Mark Williams C, and BASIC compilers were contributed by readers.

Figure 4 -- Lemma (one pass) algorithm

```
#include <stdio.h>

void stat(double *list);

main()
{
    double list[7000];
    int i;

    for(i = 0; i < 6999; i++)
        list[i]=i + 1;
    stat(list);
}

void stat(double *list)
{
    int i,c;
    double x,z,dev,square,squares,var,sd;
    double sqrt(double sd);
    x = squares = 0;
    puts("start");
    for(i = 0; i < 6999; i++){
        x += list[i];
        squares += list[i] * list[i];
    }
    z = x/i;
    printf("%f\n",z);

    var = (squares - (x * x)/i)/( i - 1);
    printf("%f\n",var);
    sd = sqrt(var);
    printf("%f\n",sd);
}
```

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

VISA/MC/check Shipping/Handling \$ 1.50
card# _____ Tax \$ _____
expires ___/___/___ Total \$ _____



"The Code Busters!"

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'

**ANNOUNCING A
CALCULATED
BREAKTHROUGH
IN COMPUTING**

If you use a
SCIENTIFIC OR FINANCIAL
CALCULATOR and a
PERSONAL COMPUTER,
you need the
PC HYPERCALCULATOR.

YOU NEED: A pop-up scientific/financial calculator for the IBM PC integrated with every program you use.

YOU NEED: A programmable calculator with 100 registers, 1000 program steps, and alphanumeric prompts.

YOU NEED: A faithful emulation of the Hewlett-Packard HP-11C and HP-12C that runs 20-40 times faster.

YOU NEED: Only \$49.95 plus \$3 for shipping (includes free 8087 version and utility programs).

Stop copying from calculator to computer now!

Call toll-free:

(800) 628-2828, ext. 502

Sunderland Software Associates
Post Office Box 7000-64
Redondo Beach, CA 90277

HP-11C, HP-12C and IBM PC are trademarks of Hewlett-Packard Co. and International Business Machines Corp., respectively.

Reader Service Number 41

**New Desktop
Services From
Micro C**

We'll design a style sheet for you, then take your (ASCII, WordStar, WordPerfect, or other) text, illustrations, and listings, from disk, paper, or our RBBS and put them together. What you get in return are pages ready for the print shop. And you'll get them for less: usually 1/5th the price of the old-fashioned graphics services. (Of course when it's time to revise the piece, the old-fashioned methods aren't even worth considering.)

Services Include:

- Graphic Design
- 300 dpi Proofing
- 300, 1270, 2540 dpi Final Art
- Illustrating
- Scanning
- Schematic Drafting
- Technical Editing
- Proofreading

For more information, contact

Micro Cornucopia
PO Box 223
Bend Oregon 97709
503-382-8048



Why are serious PC software developers demanding ...

Show Me!

Because *Show Me!* is the NEW "must have" tool for today's PC programmer. With this memory-resident, file-windowing utility, see your productivity soar as you view as many as four files at once instantly (in ASCII, EBCDIC or hexadecimal) at the touch of a key.

- escape that caged-in feeling while programming in Turbo Pascal, Turbo Prolog, Turbo C, Turbo BASIC, QuickBASIC 4.0, dBASE, BASICA, and similar programming environments, and work with up to four additional program files at once -- complete with copy & paste, print, and search capabilities
- view multiple source and listing files in up to four windows while using CodeView, DEBUG, SYMDEB, and other debuggers -- great for assembly programming!
- peek at any ASCII file including Wordstar document files and easily paste all or part of a file into virtually any program that accepts keyboard input
- visually compare files side-by-side in simultaneously scrolling windows
- find your files in the built-in directory window

Order *Show Me!* today for only \$39 (+\$5 S/H)!

To order call toll-free 800-634-3122

Visa and MasterCard accepted • 30-day satisfaction guarantee • Not copy protected



Ask about *So Help Me!*, a flexible, context sensitive help screen driver. Add full-color help screens to all your programs — royalty free — for only \$79!

Serengeti Software • P.O. Box 27254 • Austin, Texas 78755 • 512-345-2211

Show Me! & *So Help Me!* trademarks Serengeti Software; Borland Int'l, MicroPro, Ashton-Tate & Microsoft trademarks acknowledged.

Reader Service Number 27

**Full Featured AT Motherboard
fits XT or AT case! \$399**

(6/8MHz, \$489 for 6/10)

Upgrade your XT to a real AT for about
the price of an "accelerator" card

OR

build a space-saving AT from scratch.

Features:

Phoenix BIOS, 1MB memory (Ø K installed), VLSI technology, 8 expansion slots, Clock/Calendar.

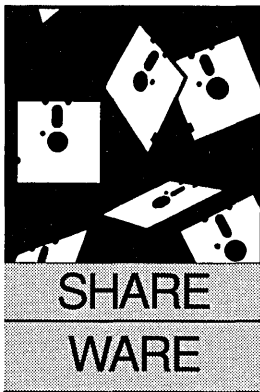
Add \$95 for 1MB memory installed and tested.

Other selected components available include

- | | |
|------------------------------------|-------|
| • Toshiba 1.2MB floppy drive | \$118 |
| • Floppy/Hard drive controller | \$194 |
| • XT size case w/LEDS, lock, reset | \$ 55 |
| • Everex Magic I/O Par/Ser card | \$ 69 |
| • Everex EGA card | \$195 |
| • Mono/Graph/Printer card | \$ 69 |
| • Famous Datadesk Keyboard | \$119 |
| • High density floppies (10) | \$ 10 |

Call SoftSide Systems at (503) 591-0870

Reader Service Number 28



Graphic Prose And Graphic Pros

Anthony Barcellos

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

Tony takes on graphics and CAD as he looks closely at PC-Key-Draw, a powerful graphics and CAD package that's available as shareware.

What are the gaps in shareware's coverage of the major applications? PC-Write, PC-Type+, and a host of others cover word processing. ProComm and Qmodem dial up communications. PC-Calc and ExpressCalc calculate spreadsheets. Databases are addressed by PC-File+ or FileExpress. And for computer-aided design—

Was that a gap we just detected? I thought so, until I came across the latest release of OEDWARE's PC-Key-Draw. While PC-Key-Draw may not yet be a true CAD package, it's a graphics program with high-level drawing functions and entry-level design features.

Author Edward Kidera says that a subsequent release of PC-Key-Draw should offer 3D drawing, autodimensioning, and other advanced CAD features, but let's see what resides in the current version (3.2).

Get the Picture

PC-Key-Draw was designed for mechanical engineers, Kidera's profession. In the documentation, he notes that PC-Key-Draw helps in four areas:

- First, it creates presentation slides.
- Second, it helps design and layout new systems. It is very easy to try a geometry and test it for a new motion compensator or similar device.
- Third, it's great for schematics as well as mechanical drafting of machine parts or complete systems.
- Fourth, it helps create drawings for technical papers.

Now that we've frightened away the non-technical readers, I might mention that PC-Key-Draw will even compute the areas and centers of mass for objects drawn on-screen. Of course, you have to sit and wait a while as the

program grinds out the numbers, but would you rather dust off your old calculus book and hand-crank a few complex integrals?

Beauties In The Beast

You might be surprised to learn that this heavy-tech program also has a more gentle side. Kidera credits the influence of three women for PC-Key-Draw's graphics flexibility. His mother and wife are artists, his sister an architect.

Hence PC-Key-Draw provides nice graphic design functions that Kidera's wife takes full advantage of: "She has made considerable use of the program to design logos and letterheads," says Kidera.

The design features include font design and modification (shading, filling, rotating, scaling, and stretching), the ability to track text along strange curves or mirror-reflect it, and pixel-level editing. Text can be mixed freely with graphics and can be positioned very flexibly—flush left, flush right, centered, or hand-placed.

Seeing Is Believing

I like demos. Kidera provides one of the best on-disk demonstrations I've ever seen. Enter "demo" at the DOS prompt and PC-Key-Draw takes you on a tour of its innumerable features.

Text dances across the screen. Fill patterns spruce up letters. Polygons fly. Smear patterns replicate an object as it's dragged about (in several different ways). Objects are cloned and reflected, colors changed, pixels edited. Aspect ratios are modified and the screen scale zooms in and out. On an 8088-based machine, you're in for several minutes of remarkable illustration.

3-D

Although Kidera lists 3-D graphics as a future enhancement, judicious use of smearing and spray-painting does a nice job of emulating it.

The demo grabs and pulls a cross-sectional drawing to give the appearance of perspective; it spray-paints a disk to give it the shading of a

sphere. Converging lines add the illusion of distance.

The current release of PC-Key-Draw only works with the color/graphics (CGA) adapter and cannot take advantage of higher-resolution video cards. (EGA and Hercules support are slated for version 4.0.)

The PC-Key-Draw images can be saved in a BASIC-compatible form—permitting further manipulation outside the program. Edward supports several printers with plotter support due in version 4.0.

True to its name, PC-Key-Draw works completely from the keyboard. While the state-of-the-art seems to demand a mouse and pull-down menus, Kidera's program was designed for people who feel comfortable at the keyboard. The major modules of PC-Key-Draw are invoked through the function keys. Here's a brief run-down:

- F1: Drawing modifications (edit)
- F2: Boxes and figures
- F3: Curves
- F4: Spraypainting and shading
- F5: Paint
- F6: Draw & text
- F7: Copy, erase, & move
- F8: Files/Zoom printing
- F9: Cursor speed
- F10: Color

Key macros, library files, and object definitions all permit the PC-Key-Draw user to save his work or procedures as building blocks that can be used later in assembling larger graphics projects.

The Bottom Line

Ed Kidera's program is distributed on two disks, complete with sample files, the demo program, and abridged documentation. The registration fee for PC-Key-Draw is \$100, at the high end for shareware programs. On the other

hand, PC-Key-Draw is decidedly a high-end shareware program. (And at 27 times the price, does AutoCAD provide 27 times the power?) Payments should be sent to:

Edward H. Kidera IV
OEDWARE
P.O. Box 595
Columbia, MD 21045-0595
(301) 997-9333

My Wish List

As we've seen, shareware is very strong in the traditional PC applications. I hope to see user-supported entries in a couple of other areas.

While high-end word processing already encroaches on desktop publishing's territory (and PC-Write's laser support makes it a clear high-end contender), no true desktop publishing program has yet appeared on the shareware lists. Is WYSIWYG document processing (with text and graphics combined) too complicated for shareware to attempt?

I doubt it.

Shareware does not compete on price alone. In functionality and sophistication, it has matched or exceeded many programs from the standard commercial sources. Since shareware *is* commercial software (with a non-traditional distribution system), authors aren't stinting on power or quality. Now that desktop publishing is a hot field, we should expect a shareware entry before too long. Does anyone have a lead for me?

I also wish that soft fonts were easier to use with my laser printer. Lasers aren't as expensive as they used to be, but soft fonts continue to be expensive. I've now seen inexpensive commercial software that does a nice job of configuring fonts for use with Microsoft Word and WordPerfect (see the review of *The Soft Font Manager* in the Decem-

ber 8, 1987, issue of *PC Magazine*), so all I need is a nice shareware font generator.

Perhaps I should also mention that the font field is fraught with peril. You can violate a copyright as quickly as you can say "Times Roman."

Of course, cloning fonts has become extremely popular, because not everyone wants to pay the licensing fees for Times Roman and Helvetica. That's why Hewlett-Packard, for example, offers "Tms Rmn" and "Helv" on their LaserJet font cartridges. So far there have been no "look and feel" suits that I'm aware of. Apparently all you have to do is choose slightly different names and everything will be all right.

ASP Update

The Association of Shareware Professionals has sent out a press kit describing "the longest meeting on record." While ASP has never physically "met," their special conference on CompuServe is a non-stop meeting place. Seventy-two active members are now on the ASP roster and prospective new members are invited to contact the organization at:

Association of Shareware Professionals
325 - 118th Avenue SE, Suite 200
Bellevue, WA 98005
(206) 454-0479

Cards and Letters

The people at PC-SIG have all they can do to keep up with the flood of shareware, so you can imagine the task facing a lone columnist. However, if you'll all give me a hand, this will be much more manageable. I welcome suggestions about shareware of special merit. Please tell me about it—even if it's something you wrote yourself.



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 Floppy Drive; MaxiSwitch AT-Keyboard
- FCC Class B approved
- 48 hour factory burn-in and testing
- 12" Monochrome Monitor (720x350) with Tilt/Swivel
- Hercules Compatible 132-column 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	EGA Monitor/Card	Add \$399
20MB XT Kit, ST-225	\$299	Taxan 770 Multisync	\$499
30MB XT Kit, ST-238	\$329	2MB EMS Card	\$129
20MB 65ms Half Height	\$249	Everex 1200B modem	\$89
42MB 38ms Half Height	\$449	Everex 2400B modem	\$179
44MB 28ms Full Height	\$575	Software	Call

Call for information & other options

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). Prices and availability subject to change without notice. Texas residents add 7% Tax.

Reader Service Number 44

AUTOTIME CORPORATION

183 OSWEGO SUMMIT
LAKE OSWEGO, OREGON 97034
(503) 635-8938

WIRELESS MODEMS 9600 BAUD
20 MILE RANGE \$1500.00

FAX BOARDS FOR PC GROUP 3
CALL FOR FAX SAMPLE \$450.00

10 MEGABYTE INTERDYNE TAPE BACKUP . . . \$135.00
20 MEGABYTE XT DRIVE KITS . . . \$225.00
30 MEGABYTE XT DRIVE KITS . . . \$255.00

I.C. CHIPS 1 year warranty on chips

EPROMS: LET US BURN YOUR ROMS
ROMS BURNED AND ERASED USING INTEL SPEC.

2716---250/450	2K*8	\$2.75/2.25
2732A---200/300	4K*8	\$2.95/2.75
2764---200	8K*8	\$3.00
27C64---200	8K*8 Low Power	\$3.85
27128A---200	16K*8	\$3.85
27256---200	32K*8	\$4.25

MATH COPROCESSORS:

8087---3	for IBM PC and XT	\$ 89.00
8087---2	for Turbo 8 Mhz XT	\$130.00
8087---1	for 10 Mhz Turbo XT	\$150.00
80287---3	for IBM AT & 8 Mhz Clones	\$125.00
80287---F	for 10/12 Mhz Clones	\$150.00

MEMORY D-RAMS:

4164---150	64K*1	\$1.15
41256---150256K*1	\$3.00
4464---150	64K*4 for New Clones	\$3.00

Reader Service Number 49

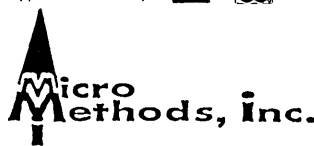
ZRP/M™ creates

Z280®

CP/M® 2.2 compatible

IBM PC

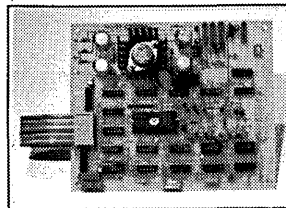
ZRP/M is an operating system combined with a Z280 emulator. Either standalone or with DOS present, ZRP/M provides the solid base of a genuine operating system reliably distinct from the facade created by an MSDOS interface. All 2.2 system and CBIOS calls are supported, 56.5k TPA, file date and time stamping, fast virtual disk, iobyte 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

MS-DOS, CP/M E/EEPROM PROGRAMMING SYSTEM

2708
2758
2716
2516
2532*
2564*
68764*
2816A
2732
2732A



2764
2764A
27128
27128A
27256
27512
27CXXX
2864A
8751*

*SOCKET ADAPTER REQUIRED - DIAGRAMS INCLUDED

A FULL FEATURED HARDWARE/SOFTWARE PACKAGE

- FAST PROGRAMMING ALGORITHM
- NO PERSONALITY MODULES REQUIRED
- INSTALL PROGRAM FOR SOFTWARE
- ALL SUPPLIES ON BOARD
- PROGRAMS 26, 25, 21 & 12.5V E/EEPROMS
- LARGE COMPREHENSIVE MANUAL
- USES NO SYSTEM POWER OR CHASSIS SLOT
- STAND-ALONE BOARD
- HIGH SPEED PARALLEL OPERATION
- FIVE LED STATUS/ACTIVITY INDICATORS
- HIGH QUALITY "TEXTTOOL" ZIF SOCKET
- REQUIRES 24 OR 25 VOLT XFMR FOR POWER

PARALLEL PRINTER INTERFACE

CONNECTS TO ANY PARALLEL PRINTER INTERFACE
USES 8 OUTPUT DATA BITS AND THE PRINTER BUSY LINE FOR DATA INPUT

CONTROL PROGRAM COMMANDS

- PROGRAM EPROM(S) FROM DISK FILE
- READ DISK FILE INTO BUFFER
- READ EPROM(S) INTO BUFFER
- VERIFY EPROM IS ERASED
- CHANGE EPROM TYPE
- SAVE EPROM(S)/BUFFER TO DISK
- PROGRAM EPROM(S) FROM BUFFER
- COMPARE EPROM(S) WITH BUFFER
- COPY EPROM(S)
- BUFFER MONITOR MODE (SEE BELOW)

THE BUFFER MONITOR MODE HAS 17 SUB-COMMANDS FOR DETAILED OPERATIONS. THESE INCLUDE: FILL, DUMP, TRANSFER, PROGRAM READ, VERIFY, EXAMINE, MODIFY, CHECKSUM, BIAS, INSPECT, SINGLE BYTE BURN, LOGICAL OPERATIONS(AND/OR/XOR), SET BUFFER BIAS, HEX ARITHMETIC, ETC.

ASSEMBLED AND TESTED UNIT WITH COMPLETE DOCUMENTATION AND SOFTWARE ON DISKETTE

\$199

PARTS KIT WITH SOFTWARE AND DOC. \$179 BARE BOARD, SOFTWARE & DOC. \$69
SOFTWARE AVAILABLE ON 5 1/4" OR 8" DISK FOR IBM, KAYPRO, & OTHER FORMATS

TO ORDER SEND CHECK, MONEY ORDER, WRITE OR CALL:



ANDRATECH
P.O. BOX 222
MILFORD, OHIO 45150
(513) 752-7218



CALL OR WRITE FOR MORE INFORMATION - ADD \$4.00 FOR SHIPPING - \$3.00 COD

Reader Service Number 30

go, they really go.)

I grabbed another Verbatim disk. This time I left the disk in its jacket and dumped it, jacket and all, in boiling water.

Within minutes the jacket fell apart, but when I lifted out the disk it was dry, shiny, and unscathed (except for a slight curl). I popped the disk into a clean jacket and put it into the machine. The first time through CRC couldn't read one of the files. On the second time through, however, it read all of them correctly.

Then I grabbed the cheapest, generic-est, oldest beater from my disk stack, filled it with files and CRC'd the lot. Then I gave the disk the same boiling water treatment, cut open its jacket, and dried out its flimsier middle (drying took two hours). The surface of the disk looked awful, but none of its data had evaporated.

Conclusion: Common disks stand up amazingly well to water torture. But, I'm surprised how well these teflon disks repel other things, especially fingerprints. (It was hard to get fingerprints to even show up, the surface was just too slick.)

After the scalding water bath, the second Verbatim was as shiny and clean as new. I think the peanut butter was a little too much for the first disk (it stuck to the roof of the drive).

Also, the coating does more than just protect against grime. The teflon coating should protect the drive's head from those hard little oxide particles. And the coated disk surface should wear longer. They've run a disk 3,000,000 revolutions (almost 21 days of head-down reading and writing) and found no phonographing (circular tracks) or gouging.

Prices are supposed to be 15% higher than standard Verbatim media. The premium price might easily disappear, however. I noticed there was no burnishing on their disks (no need to, the head rides on the teflon, not on the oxide). That would account for their variation in coating thickness, and it means that they have saved themselves a very big step. They can punch the cookies out of the tape, stick them in jackets, and test them. No more polishing. If the coating hasn't reduced yield significantly, they've got a winner.

However, the thickness of the coating is definitely a concern. They say they have no plans for higher density 5-1/4" or for 3-1/2." That tells me that coated standard double density may already be pushing the limits.

Too bad. Their liquid resistant, wear proof media inside those fold-proof 3-1/2" packages would make for an almost indestructible disk.

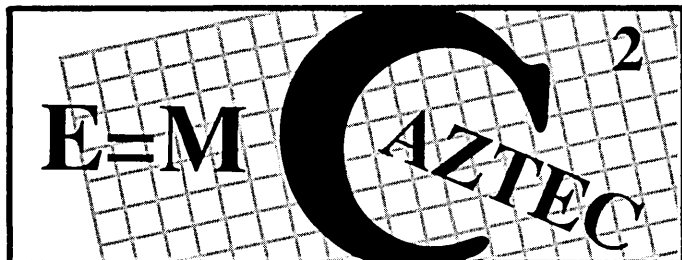
Other Disk Manufacturers

I asked other disk manufacturers what they thought about Verbatim's new product. Each time they responded with statistics. They insisted that only a tiny percentage of data loss results from physical damage to the disk. Operator error, magnetic fields, and misplacement were all more significant.

But I also picked up an underlying note of terror. Verbatim's ads showing a disk covered with everything you'd find in an office (except peanut butter, of course) have really spooked the competition. Ah, the joys of being a marketing type and blowing away the competition with a sexy new product. (And if Verbatim is manufacturing them for less...)

Taiwan, Hong Kong, Korea...

In Comdex's press room I found myself, by chance, seated next to an editor from the Far East (his accent was midwest).



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.

Aztec C86-d.....\$299
• includes all of Aztec C86-p • Unix utilities make, diff, grep • vi editor • 6+ memory models • Profiler.

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-c.....\$499
• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

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 • PforCa • Pre-C • Windows for C • Windows for Data • C terp • db_Vista • Phact • Plink86Plus • C-tree.

CP/M • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c CP/M & ROM....\$349
Aztec C II-d CP/M.....\$199

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.

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

MANX 1-800-221-0440
Manx Software Systems
One Industrial Way
Eatontown, NJ 07724
To order or for information call today.
In NJ or international call (201) 542-2121
TELEX 4995812

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 \$29!

Our FULL COMMERCIAL VERSION of CBTREE in object library format is being offered for the amazingly low price of \$29.

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 \$70 difference. Example source programs and utilities are included FREE.

CBTREE source library \$99
Object library only \$29

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

PEACOCK SYSTEMS, INC.
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

**DISK FORMAT
CONVERSION**

XenoCopy-PC

PC-DOS program
lets your PC

\$79.95 + \$5.00 S/H
Sales Tax if CA.
VISA MasterCard 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™

1454 Sixth Street, Berkeley, CA 94710

(415) 525-3113

this was on purpose.

good software is it's own reward.

Reader Service Number 39

After I assured him that he could remain nameless, he told me some interesting tales.

He says we'll soon be seeing single-board XT clones. They are now starting to manufacture single-board XTs with floppy controllers, graphics controllers, serial, parallel, game port, real-time clock, the works, everything. (Are you Big Board owners getting a little nostalgic?) He said the main board contains two ASICs, a V20, and RAM.

That's it.

(I know, there isn't room for all that stuff in three chips and memory. You know there isn't room. Fortunately, they don't know there isn't room.)

The board should sell for \$117 in onesies. In the U.S.

He mentioned they might send over complete computers based on this board for \$230 each. No other details on the system, but the price is a bit low if you figure drives, cabinet, power supply, monitor, and keyboard. (But then they think they can cram all that stuff into three chips...)

80386

He added that there are 58 companies in Taiwan building 386 boards. They are paying \$400 each for parts from Intel's stash of defective 80386 processors. (Those are the only chips Intel will sell them.) Intel says they will replace any chip that a user discovers is defective, but I understand the problem only shows up when the 386 is doing a multiply instruction in native mode.

Maybe Intel is betting there are no natives in the software business.

PS/2

He also mentioned that Taiwan will definitely support PS/2. IBM says it will be licensing most of the 30,000 patents for the technology for about 1% of the FOB price of a system or 3% of any value added. (I wonder if that means dealers who assemble systems based on licensed PS/2 boards will have to pay 3% of their markup.)

Several manufacturers are ready to sign papers with IBM, but IBM has been dragging its feet. Apparently, sales of the PS/2s have not gone as well as expected.

He says that Taiwan is also looking at licensing TI and AST buses.

Saving Your Seagate

Dusty Johnson at Rotating Memory Systems suggested a way to keep your Seagate drives (the half-heights) running. Install an autopark routine (after 30 seconds without a drive access, it parks the heads on the innermost track). If something happens to the power and the heads are on the inside track, there's no chance something will happen to the drive's guard band.

If something does happen to the guard band, the drive will step back to track 0 and then try to keep going. Usually it's because the heads were already at track 0 when the drive was started up.

When the head tries to back out past track 0, it hits the stop. (And, if the stop has shifted, the head won't find anything.) So the head assembly starts banging repeatedly against the stop (making a terrible ratcheting noise until the stepper motor dies).

If you hear the clatter, turn off your system instantly (if not sooner). Take out the drive and underneath, near one corner, you'll see the stepper motor. You'll also see the end of the

stepper motor shaft pointing at you.

The shaft should be fully clockwise. You want to turn it fully the other way (counterclockwise) to move the heads far away from track 0. (The shaft doesn't stick out so you may need to be ingenious.)

Now fire up the system. Hopefully you won't hear that awful clatter. If things are okay, copy everything off the drive. (If you're using MS-DOS's BACKUP, make five copies, at least.)

Thanks Dusty.

SOG VII, The Systems Design Conference

Again, SOG will be held in Bend, July 14 - 16. Sessions will be held at Central Oregon College, but we won't have the dorm this year (remodeling and all).

There are plenty of places to stay in Bend, trailer parks, campgrounds, motels, resorts, etc. So sleeping shouldn't be a problem.

The only reason we've encouraged early reservations at previous SOGs is because of the limited dorm space. Since we don't have the dorm, we're moving SOG registrations back to May 1 (by request of the SOG registration department).

Despite waiting until May, you'll have plenty of time to sign up for the Thursday BBQ and Saturday evening banquet, the all-day and half-day white water rafting, and T-shirts (don't miss a year in your collection).

Also, this year's technical events will include hardware and software design roundtables. They'll feature great designers and, as usual, audience participation is mandatory.

Want to carpool? Want more information on places to stay? Check into the SOG forum on the Micro C RBBS (even before May 1) (503) 382-7643, 300-1200-2400, 8 bits, 1 stop, no parity, 24 hrs. Leave a message if you have traveling room to spare or need to find someone coming out from your area. Any questions about what's happening? Leave a message for the Sysop.

Or, second best, we can send out brochures about the area and about lodging.

Either way, registration begins on May 1, and we'll look forward to seeing you in Bend, July 14 - 16 for the latest, greatest, designers forum you could imagine.

Back-It Version 3.0

I've been watching for a good backup utility for a long time. I even purchased one of those \$600 tape backups that hooks to a floppy controller. It didn't work. I went back to MS-DOS's BACKUP and RESTORE. Free, but not worth it.

Back-It is a backup utility that corrects its own data and works incredibly fast. However, it has a menu system whose organization takes a little getting used to. (This appears to be a classic case where the programmer is too close to the package to see a new user's problems.) The disassembler I covered in the Comdex report was much easier to use than this backup program.

The program shouldn't require a lot of thought. You should be able to point to some files, specify a destination drive and turn the program loose. The less fuss the better.

Requirements

Apparently Back-It expects a 765 floppy controller, usually not a problem, and, apparently, it expects XTs to be running 4.77 MHz. (If you can't verify a COPY because you speeded up your system and are no longer meeting setup and hold time on your floppy controller, then Back-It gets strange.)

HANDS-ON PC REPAIR TRAINING

GO TO PAGE 1 PART 1

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.



17985-F Sky Park Circle
Irvine, CA 92714
(714) 250-0834

Computer Hardware Training Specialists

Reader Service Number 59

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 Price **\$79**

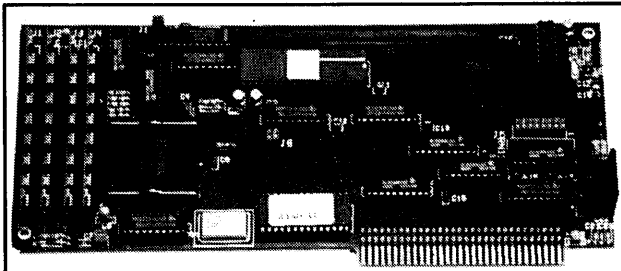
System Requirements: IBM-PC, XT, AT or Compatible running DOS 2.0 or higher. Screen graphs require graphics card. Printer Graphs require Epson EX, FX, JX, RX, HS; Star Gemini, Radix, SD, SG, SR; IBM Graphics; or compatibility with one of the above. Plotter graphs require HP-GL compatibility.



P.O. Box 956, Dept. M, Valley Forge, PA 19482
For Technical Information: (215) 269-0198

Reader Service Number 60

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

Features

Feed it unformatted disks and it will format them after asking: "Do you really want me to format this?" Feed it disks with sector errors and it spits them back (hooray!). It has no trouble with large quantities of data covering many, many disks. It can recover selected files from floppies, create its own subdirectories during recovery, and do it all very quickly. Once you and the program finally get together, the results are great.

With a new user interface and a better manual this package would be dynamite. Even without that, the data reliability makes it very worthwhile.

Back-It Vs 3.0 \$129.95

Gazelle Systems

42 North University Ave, Suite 10

Provo Utah, 84601

(800) 233-0383

YAP (Yet Another Plug)

I was cleaning up my office (opening a path to the desk) today when I came upon the empty remnants of the Mavis Beacon box. (Let's see, Larry had the master last... Or was it Laura?) Since cleaning and I don't get along, I thought I'd fill you in about Mavis. (At least until lunch.)

Mavis Beacon is a typing tutorial. A fun typing tutorial. Seriously, it's the best, most innovative typing trainer I've seen.

This package lets you play games, type mysteries, compete with the program, etc. Its graphics and sound are as good as the best games. And while you're enjoying the lessons, it's keeping statistics on your progress.

If you're a beginner (or an intermediate), it'll put a keyboard on the screen showing the proper fingers hitting the keys as you type. You'll never need to look at the keyboard again.

It even understands the sounder part of typing. Make a mistake and it trills. Do something well and you get a musical pat on the back. Enter the road race and the faster you type, the faster your motor runs. (And you thought your motor ran for other reasons.)

Mavis will move you quickly to harder tasks if your speed and accuracy are high. Or, it will ask you if you're getting tired when your error rate rises.

You may not want to do too well when you start out. Be too successful and you'll soon be over your head, but that's a small knock on a very smart and entertaining program.

Anyway, if you're hooked on hunt and peck (or worried someone you love might fall into this horrid trap) you'll wait a long time to find a better teacher.

Mavis Beacon

The Software Toolworks

One Toolworks Plaza

13557 Ventura Boulevard

Sherman Oaks, CA 91423

(818) 907-6789

And that's all from greater Bend.

David Thompson

Text Editor & File Copier

ORDERFORM

POSTAGE-PAID SELF-MAILER

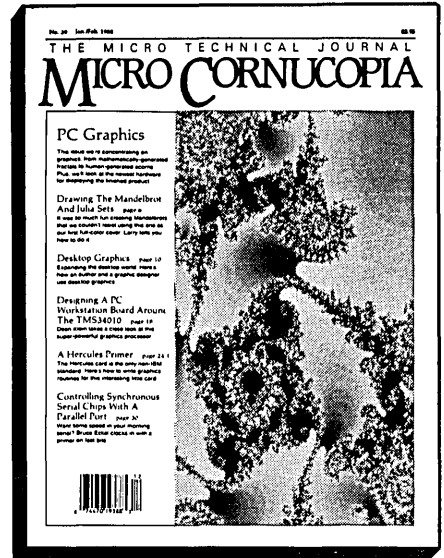
Tear out, fold, and staple both ends if check is enclosed.

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 5¼" MS DOS 3½"
 Other
 Specify Disk # and size _____

DISK TOTAL

OTHER PRODUCTS
 Back Issues, T-shirts... specify size _____

OTHER TOTALS

Save
24% Off
 the
 newstand
 price

GRAND TOTAL

CHECK ENCLOSED
 U.S. funds drawn on a U.S. bank, please

VISA MASTERCARD

- - -

Are you a current Micro C subscriber? Yes No

Expires

To Place Your Order Immediately
CALL: 1-800-888-8087
 9-5, M-F, Pacific Time

NAME

COMPANY

ADDRESS

CITY STATE ZIP

40

TEAR HERE

MICRO CORNUCOPIA

READER SERVICE CARD MAR./APR. 1988 ISSUE NO. 40

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

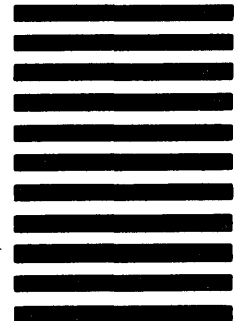
FIRST CLASS PERMIT NO. 19 BEND, OR

POSTAGE WILL BE PAID BY ADDRESSEE

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

P.O. Box 223
Bend, OR 97709-0223



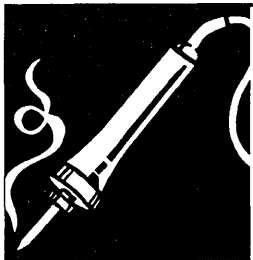
FOLD HERE

**NEXT
ISSUE**

IN ISSUE #41

- Neural Networks — do they really think
- Ray-Tracing Algorithms on the PC Tech 34010
- Parser Generation and Turbo Prolog and C
- Information Theory and Communications
- Fractals In the Real World

STAPLE TO CLOSE



TECHTIPS

Technical Tips

Turbo C Fixes

I agree with the comments made in reviews of Borland's Turbo C in the not-August issue (#37). But I wish to pass on the details of a problem I had with the compiler in the hope of saving others from the same trap.

The atan2(y, x) function must be of much more use to engineers than programmers. It seems to be poorly coded, and untested, in several compilers: Turbo C is one of them. I got Turbo C early in June and found that this function failed for zero values of x.

Microsoft C, Desmet without 8087 support, and Greenhills C for the DSI-32 work well, while Turbo C, Desmet for the 8087 (at least my version), and the UNIX version 3 C compiler do not.

I sent a note with the registration card describing how the function would hang the system and got a prompt acknowledgement from Borland. A few days later a complete set of replacement disks came, labelled version 1.0 (same as the first), with the problem fixed.

Couldn't ask for better support. The code in Figure 1 has been useful for exercising the atan2 function.

John S. Innes
120 Macpherson St.
Cremorne NSW 2090
Australia

The Woes Of Computer Neck

So you've got Mr. Computer all assembled and have been using him full tilt since sometime last week? You've been working 10 - 14 hours a day on the new job, and your head aches? Shoulders tight? Eyes blurry, won't focus? You can't get rid of the pain between your shoulder blades? Lower back stiff and sore?

Welcome to the wonderful world of "computer neck." As a chiropractic physician specializing in spinal biomechanics, I see a lot of "computer necks." You can have your own in as little as two hours.

Here are some simple steps you can

take to help reduce this misery. Although there are conflicts between bits, bytes, and bones, you *can* take advantage of the inherent design strengths of your musculoskeletal system.

There is a basic relationship which balances your bones and muscles with the effects of gravity. If let your bones support you, your muscles won't fatigue so quickly.

From the side, while standing, a line drawn from the lobe of the ear should go through the point of the shoulder, the point of the hip, and just behind the bump of the ankle. Sitting, your upper body balance should not change. Sit upright with your chin in a neutral position and your shoulders down and back.

Chair height should let your legs to reach the floor without dangling. Thighs and forearms should be nearly level with the floor. If you support either your wrists or elbows (both is even better) you're removing load from your shoulders.

The keyboard *must* be directly in front of and below the monitor. I've found that raising the middle or bottom of the monitor to eye level works best for most of my patients. You simply cannot work effectively with the keyboard on your knees and the monitor on the desk, or with the two at a 20 - 60 degree offset.

A short break every 20 - 30 minutes will also help—do head and shoulder rolls or other stretching. Anyone too busy to keep themselves comfortable while working will see their efficiency and accuracy drop. If these basic tips don't relieve the problem, check with your local chiropractor.

Jack Pedersen, D.C.
PO Box 65
Sweet Home, OR 97386



Figure 1 - atan2 Test Function.

```
TSTATAN2.C - tests operation of atan2 function

#include <math.h>
#include <stdio.h>

main ()
{
    double atan2 (), x, y;
    int i;
    static double v[8] = {0, 1.0, 1.0, 1.0, 0, -1.0, -1.0, -1.0};

    for (i = 0; i < 8; ++i)
    {
        x = v [(i + 2) % 8];
        y = v [i];
        printf ("%d times PI/4 = %.3f\n", i, atan2 (y, x));
    }
}
```

MICRO ADS

A Micro Ad is the inexpensive way to reach over 22,000 technical folks like yourself. To place a Micro Ad, just print out your message (make it short and sweet) and mail it to Micro C. We'll typeset your ad (no charge) and run it in the next available issue. You can also send camera-ready copy. Rates: \$99 for 1 time, \$267 for three times, \$474 for 6 times (a best buy at only \$79 per insertion). Full payment must accompany ad. Each ad space is 2 1/4 inches by 1 3/4 inches.

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

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

 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

FREE BUYER'S GUIDE

Call or write for our FREE comprehensive "Buyer's Guide" containing hundreds of languages, utilities and books specially for IBM personal computers and compatibles. We're the world's leading independent dealer of programmer's development tools because we provide sound advice, low discount prices, fast delivery, FREE domestic shipping and no hidden charges.

Programmer's Connection
800-336-1166 USA
7249 Whipple Ave. NW
North Canton, OH 44720
800-225-1166 Canada
9102408879 Telex
216-494-3781 International

Reader Service Number 51

QL Computer \$99.95 pp

128K Ram, expandable to 896K, 68008 Chip, 32 Bit processor, 2 Built-in microdrives. Write for catalog and information.



Sharp's Inc.

Rt. 10 Box 45
Mechanicsville, VA 23111
(804) 746-1664

Advanced C Source Code Analysis With C:LINES/C: TREE

- Outlined listings reflect flow of control *within* procedures.
 - Tree diagrammer shows flow of control *between* procedures.
 - Source code reformatter.
- \$59.95, shipping & handling included. Maryland residents add 5% sales tax. Requires PC Compatible, MS-DOS 256K

SOFTREX • 4807 Bethesda Av., Suite 287
Bethesda, Md 20814 • 301-881-8274

Reader Service Number 56

For PCs \$1250

(212) 580-0257

Parallon™ parallel processors

Human Devices, Inc. 322 W 71 St NY NY 10023

Reader Service Number 45

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

GA/3 VIDEO CARD

Use Up to 3 Monitors From One PC
Use All 3 Monitors Up to 1000 Feet
From Your PC

- Supports CGA/HGC/MDA in any combination
- Works with standard PC software also
- XT/AT example software included
- Long Distance Option \$39.95 per monitor
- Quantity/OEM discounts

GA/3 Video Card \$699



Advanced Products Division
6280 East Progress Lane, East Wing
Parker, Colorado 80134

303/841-4772 (Colorado) 800/237-2842 (Outside CO)

Reader Service Number 81

HOW COMPLEX IS YOUR SOFTWARE? FIND OUT WITH PC-METRIC™!

- computes popular complexity metrics
- checks program against standards
- predicts bugs and effort
- versions for many popular languages
- includes 100 page tutorial/manual

SET Laboratories, Inc.

Dept. MC, Box 03627, Portland, OR 97203

Reader Service Number 58

Programmer's Paradise

The microcomputer software source that caters to your programming needs.

**Gives You Superb Selection,
Personal Service, Unbeatable Prices!**

CALL OR WRITE FOR THE LATEST CATALOG

1-800-445-7899

In NY: 914-332-4548

Programmer's

Paradise™



A Division of Hudson Technologies, Inc. 42 River Street, Tarrytown, NY 10591

Reader Service Number 68

IBM Style Monochrome Card
with parallel port. Made in Japan.
\$35.00

XT Style Keyboard without case
mfg. by Cherry \$25.00

**The dBase Book of Business
Applications**, 335 p. \$3.95

Koala Computer, Inc.

4306 Torrance Blvd.
Torrance, CA 90503
(213) 316-5866

Reader Service Number 88

Macro Language/I (ML/I)

The fast, language independent macro processor

- Extend your current programming language or define your own.
- Many useful utility ML/I definitions included.
- Speed source program conversions or level upgrades.
- Unlimited macro definition passes. Supports 30K macro definitions per pass.
- Define 4GL extension to your production language/environment.
- Define new levels of programming abstraction and reduce software development time, effort, and tedium.
- Define object-oriented language extensions.
- Define complex text search and replacement macros.

Requires: IBM PC, XT or AT and 100% compatibles, PC-DOS (MS-DOS) 3.1 or later, 256K. Now only \$199.95, residents of New Jersey add 6% sales tax. (609) 448-3876 **CACS Software, 34 Brooklawn Drive, East Windsor, New Jersey 08520.**

ADVERTISERS INDEX

Issue 40

RS#	PG#	7 CompuView47	12 Logitech Inc.31	29 Pecan Software25
72 Acquired Intelligence25		8 Datadesk Int'l1	17 Manx Software Systems...87	68 Programmer's Paradise.....7
44 Amtech Computers86		32 Digital Research Comp...52	42 McTek Systems.....53	60 Scientific Software89
38 Analogic Company79		16 Dreamtech41	35 Merlyn Productions70	18 Semidisk Systems52
30 Andratech86		33 E2I Computer74	** Micro Cornucopia30	27 Serengeti Software.....83
48 ASCII15		9 Ecosoft61	** Micro Methods86	19 Slicer Controls/Compute .90
4 Austin Codeworks65		10 Emerald Microware67	36 Microcomputer Systems...52	28 Softside Systems83
49 Autotime Corp86		** Gimpel Software66	37 Microprocessors Unlmted .74	40 Star-K Software Systems ..24
74 BAS/C Software23		55 Guidelines Software25	24 microSOLUTIONS.....51	41 Sunderland Software83
5 Blaise Computing5			2 Microsphere.....2	62 V Communications41
1 Borland Int'lBack Cover			57 MMC AD Systems.....24	
53 C Store57		11 Halted Specialties.....40	59 National Advancement Co.89	14 WindowDOS Assoc.....63
15 Cascade Electronics75		26 Harger I.N.T.46	3 PC Tech ..Inside Front Cover	39 Xenosoft88
31 CC Software82		34 Hawthorne Technology ...70	20 Peacock Systems79	70 Zortech, Inc. .Inside Back Cover
71 Complete Logic Systems ..29		22 Integrand30	20 Peacock Systems88	
6 Computer Helper Ind.71				

*Advertiser wishes to be contacted directly.

FastCopy 3.0

Turns your PC into a diskette duplicating machine!

for ONLY \$69!

SYSTEMS, SOFTWARE, SUPPORT
PO Box 751022
Houston, Tx 77275-1022
713-941-3100

Desktop Publishing, Graphics, Laser Printer Fonts, & More!

Our multiuser BBS has 3 lines on rotary and 260+ Mb of files for MS-DOS, CP/M, and soon Macintosh users. One small \$40.00 per year fee gets you access to our huge online library. Many disks available by mail for \$5.00 each, FREE catalog if you send disk, mailer, & return postage to:

DataCOM Systems, 2643 Cedarview Court,
Clearwater, Florida 34621-3710
(813) 796-5627 modem 1200 or 2400 baud.

Reader Service Number 76

PEN PLOTS

WE SUPPORT:
AUTOCAD. ORCAD. GENERIC
LASER CAD. AND MORE

electronic design services
1743 SE GAIL CT.
HILLSBORO, OR 97123
(503) 648-2310

Reader Service Number 69

Computer Industry Almanac

Fact-filled guide about the computer industry. Includes people, companies, products, trends, natl users groups, associations, headlines of the year, book clubs, periodicals, forecasts, financial facts, benchmarks, fun trivia. 780 pgs. \$29.95 + \$2 shipping.

Computer Industry Almanac, Inc., 8111 LBJ Frwy, 13th Floor MC, Dallas, TX 75251-1313. Telephone (214)231-8735.

Reader Service Number 46

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

Azatar DOS Toolkit

Turbo Pascal Tools for DOS 3.XX

NEW! Featuring:

- 80 Pascal procedures and functions
- Professionally bound manual
- Example programs
- No licensing for compiled code

Only \$95!!

Azatar MicroSystems Inc. 3300 Monroe Ave.
Rochester, NY 14618 (716)385-9780 Hrs: 9-5 Eastern

Reader Service Number 47

OPT-TECH SORT/MERGE


Extremely fast Sort/Merge/Select utility. Run as an MS-DOS command or CALL as a subroutine. Supports most languages and filetypes including Btrieve and dBase. Unlimited filesizes, multiple keys and much more! MS-DOS \$149. XENIX \$249.

(702) 588-3737

Opt-Tech Data Processing
P.O. Box 678 - Zephyr Cove, NV 89448

Reader Service Number 64

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!

Mandrill Corp.
PO Box 33848, San Antonio, TX 78265
800-531-5314 Dealers inquiries welcome. \$185.

TurboCAD™

The New Generation CAD

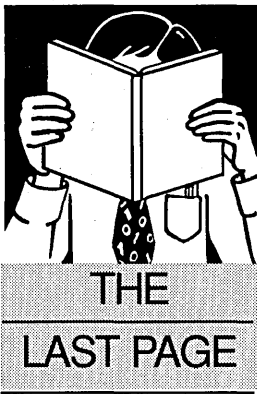
LIST PRICE \$395

"EASILY OUTCLASSES OTHER SIMILARLY PRICED CAD SOFTWARE"

ONLINE

NOW ONLY \$99

Milan Systems America Inc.
8351 Roswell Road
Dunwoody, GA 30350 (404) 642-4131



The Other Side Of TSRs - DOS Shells

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

C shells by the C shore? Prolog shells by the Prolog shore? Here Gary looks at the backside of TSRs—SHELLs.

If you want to have access to more than one program at a time, you can try either of two approaches—terminate and stay resident (or TSR, see Tidbits this issue) or simply stay resident (or SSR).

Another name for SSRs is DOS SHELLs, and here's how they work.

You run the SHELL which executes other programs. When the "other" program finishes executing, the SHELL retakes control and idles until you tell it which program to run next.

Both Turbo Prolog and Turbo C make the writing of this kind of SHELL fun by providing a SYSTEM predicate (in Prolog) and a SYSTEM function (in C). I use one of these SHELLs all the time, and assure you that from simple beginnings (see Figure 1) and the addition of a little imagination, you can create a sophisticated (i.e., Mac-like) SHELL easily.

This issue, I'll show you how to get started, and as time goes by, we'll add user-friendlier features.

The code in Figure 1 (in Turbo Prolog) creates a window on five functions—

- a directory program
- an editor
- a memory status function
- an exit to DOS
- and an exit from the SHELL

After creating the window, it waits for a key and then tries to match the key to instructions in a table. If it finds a match, it executes the function (or runs a program). When the function or program is finished, the window is recreated (via FAIL and BACKTRACKING), and the process recycles until the user enters a 'Q' for quit.



Figure 1 -- SHELL (version 0.1)

```
DOMAINS file = save_file
PREDICATES main table(CHAR) repeat
GOAL main
CLAUSES
main:-
  repeat,
  makewindow(1,7,0,"G's Shell",0,0,7,14),
  makewindow(2,7,0,"",0,0,25,80),
  shiftwindow(1),
  write("Dir"),nl, /* Show choices. */
  write("Edit"),nl,
  write("Memory"),nl,
  write("eXit"),nl,
  write("Quit"), cursor(0,0),
  readchar(C), /* Get key (or command).*/
  table(C), /* Look up key in table.*/
  shiftwindow(1), removewindow,
  shiftwindow(2), removewindow,
  fail. /* force backtracking & repeat.*/

table('D'):-
  makewindow(3,7,0,"",5,10,10,60),
  dir("", "*. *", "_"), /* Show files. */
  removewindow.
table('E'):-
  makewindow(3,7,0,"",5,10,10,60),
  dir("//", "*. *", Knowledge_file),
  /* Show files. */
  removewindow, /* And get choice. */
  makewindow(8,7,7,"Edit",8,15,16,50),
  file_str(Knowledge_file, Knowledge),
  edit(Knowledge, NewKnowledge),
  openwrite(save_file, Knowledge_file),
  /* Save edited file. */
  writedevic(save_file),
  write(NewKnowledge), closefile(save_file).
table('M'):-
  makewindow(3,7,0,"",5,10,5,17),
  storage(S,H,T), /* Get memory stats. */
  write("Stack = ",S),nl,
  write("Heap = ",H),nl,
  write("Trail = ",T),
  readchar(_), removewindow.
table('X'):-
  shiftwindow(2),
  system(""). /* Give DOS cntl temp. */
table('Q'):-
  shiftwindow(2),
  exit(0). /* Give DOS control perm. */

repeat.
repeat:- repeat.
```

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 220 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!

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

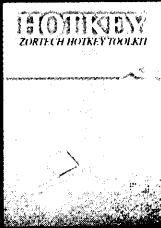
ZORTECH Inc. 361 Massachusetts Ave, Arlington, MA 02174

Orders & Enquiries Tel: (617) 646 6703

ORDER HOTLINE (617) 646 6703

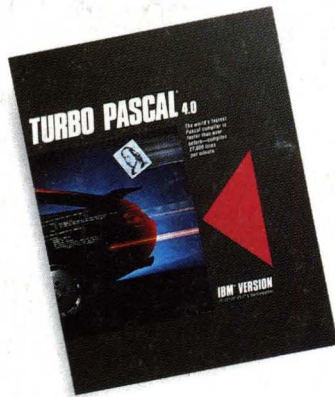
Reader Service Number 70

Turbo C is the trademark of Borland Inc. Quick C is the trademark of Microsoft Inc.



Program in the fast lane with Borland's new Turbo Pascal 4.0!

Our new Turbo Pascal® 4.0 is so fast, it's almost reckless. How fast? Better than 27,000 lines of code per minute.* That's more than twice as fast as Turbo Pascal 3.0.



4.0 Technical Highlights:

- Compiles 27,000 lines per minute
- Includes automatic project Make
- Supports > 64K programs
- Uses units for separate compilation
- Integrated development environment
- Interactive error detection/location
- Includes a command line version of the compiler
- Highly compatible with 3.0

4.0 breaks the code barrier

No more swapping code in and out to beat the 64K code barrier. Designed for large programs, Turbo Pascal 4.0 lets you use all 640K of memory in your computer.

4.0 uses logical units for separate compilation

Pascal 4.0 lets you break up the code gang into "units," or "chunks." These logical modules can be worked with swiftly and separately. 4.0 also includes an automatic project Make.

4.0's cursor automatically lands on any trouble spot

4.0's interactive error detection and location means that the cursor automatically lands where the error is. While you're compiling or running a program, you get an error message *and* the cursor flags the error's location for you.

Only \$99.95

60-Day Money-back Guarantee**

For the dealer nearest you, or to order now,
Call (800) 543-7543

For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles

Sieve (25 iterations)

	Turbo Pascal 4.0	Turbo Pascal 3.0
Size of Executable File	2224 bytes	11682 bytes
Execution speed	9.3 seconds	9.7 seconds

Sieve of Eratosthenes, run on an 8MHz IBM AT

Since the source file above is too small to indicate a difference in compilation speed we compiled our CHESS program from Turbo Gameworks to give you a true sense of how much faster 4.0 really is!

Compilation of CHESS.PAS (5469 lines)

	Turbo Pascal 4.0	Turbo Pascal 3.0
Compilation speed	12.1 seconds	35.5 seconds
Lines per minute	27,119	9,243

CHESS.PAS compiled on an 8 MHz IBM AT

*Run on an 8MHz IBM AT.

**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. Copyright ©1987 Borland International, Inc. BI 1166A

BORLAND

YES! I want to upgrade to Turbo Pascal 4.0 and the 4.0 Toolboxes

If you are a registered Turbo Pascal user and have not been notified of Version 4.0 by mail, please call us at (800) 543-7543. To upgrade if you have not registered your product, just send the original registration form from your manual and payment with this completed coupon to:

**Turbo Pascal 4.0 Upgrade Dept., Borland International
4585 Scotts Valley Drive, Scotts Valley, CA 95066**

Name _____
 Ship Address _____
 City _____ State _____
 Zip _____ Telephone (____) _____

This offer is limited to one upgrade per valid registered product. It is good until June 30, 1988. Not good with any other offer from Borland. Outside U.S. make payments by bank draft payable in U.S. dollars drawn on a U.S. bank. CODs and purchase orders will not be accepted by Borland.

For the IBM PS/2™ and the IBM® and Compaq® families of personal computers and all 100% compatibles

†To qualify for the upgrade price you must give the serial number of the equivalent product you are upgrading.

Please check box(es)

- Turbo Pascal 4.0 Compiler
- Turbo Pascal Tutor
- Turbo Pascal Database Toolbox
- Turbo Pascal Graphix Toolbox
- Turbo Pascal Editor Toolbox
- Turbo Pascal Numerical Methods Toolbox
- Turbo Pascal Gameworks

**Suggested
Retail**

**\$ 99.95
69.95
99.95
99.95
99.95
99.95
99.95**

**Upgrade
Price†**

**\$ 39.95
19.95
29.95
29.95
29.95
29.95
29.95**

Serial No.

Total product amount \$ _____
 CA and MA residents add sales tax \$ _____
 In US please add \$5 shipping and handling for each product
 Outside US please add \$10 shipping & handling for each product \$ _____

Total amount enclosed \$ _____
 Please specify diskette size 5¼" 3½"

Payment: VISA MC Check Bank Draft

Credit card expiration date: _____/_____/_____

Card # _____