No. 47 May-June 1989 $3.95

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Robotics

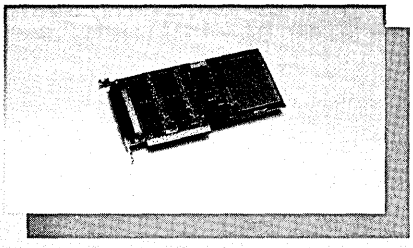
Here we go folks, rug rats that scatter spare (and not so spare) parts on the floor.

And More . . .

04

0 74470 19388 3

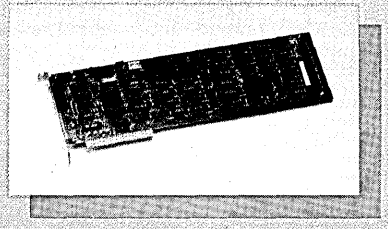
Turn your PC/XT/AT into a powerful Laboratory and Engineering tool...



Digital I/O and Counter Card

- **32 Digital Input Channels**
  - TTL compatible
  - Low loading: 0.2 mA at 0.4V input
- **32 Digital Output Channels**
  - TTL compatible
  - Driving capacity: Sink 24 mA, source 15 mA
- **Intel 8253 Timer/Counter**
  - 3 channels of timer/counter
  - Breadboard area for flexible user configuration

PCL-720.....$160.00



12 Bit Multi-Lab A/D + D/A + DIO + Counter

- **Analog Input (A/D converter)**
  - 16 single-ended channels, 12 bit
  - Input range: +5V to -5V, +1V to -1V
- **Analog Output (D/A converter)**
  - 2 channel, 12 bit
  - 0 ~ +5V full range
- **Digital I/O**
  - 16 channels each, TTL compatible
- **Counter**
  - 1 channel of timer/counter
  - Programmable pacer function

PCL-712.....$295.00

Introducing...

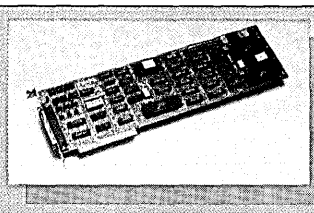
The PC-LabCard Family

Only from HSC Electronic Supply

IBM PC/XT/AT and it's compatible models are moving into Industrial/Laboratory applications at an increasing rate. The reasons for this include their price/performance ratio and short user learning curve. PC-based data acquisition boards are now taking the place of the traditional data loggers or recorders which cost several times more.

"PC-LabCard" is a family of add-on cards to turn your PC into a high performance data acquisition/ testing system at an attractive price.

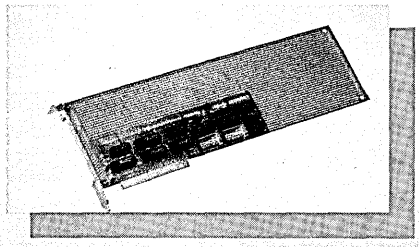
It includes not only the hardware cards, but also the software, accessories and application support packages which come together to make a thoughtful solution to your PC-based automation needs.



Stepping Motor Control Card

- Independent, simultaneous operation of up to 3 motors
- Programmable speed from 3.3 to 3410 PPS
- Built-in acceleration control
- One clock (Pulse, Direction) or two clock (CW, CCW Pulses) output mode
- Opto-isolated pulse-train outputs
- Read-back step position
- Crystal-based timing
- 8 bit digital Input and Output

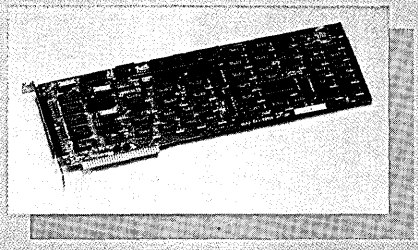
PCL-738.....$395.00



Prototype Development Card

- Large breadboard area (3290 holes)
- Independent memory and I/O address decoders built-in
- Memory and I/O ports are jumper selectable
- All bus signals are buffered, marked, and ready for use

PCL-750.....$74.00



14 Bit Super-Lab A/D + D/A + DIO + Counter

- **Analog Input (A/D converter)**
  - 16 different channels
  - 14 bit, 25K/sec sample rate
  - Input range: +5V to -5V, +1V to -1V
- **Analog Output (D/A converter)**
  - D/A Channels: 1 standard, 1 optional
  - 14 bit, +/- 5V full rage
- **Digital I/O**
  - 16 channels each, TTL compatible
- **Counter**
  - 1 channel of timer/counter
  - Programmable pacer function

PCL-714.....$495.00

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

Electronics - Prototyping Supplies
Computers - Lasers

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

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

VISA MasterCard American Express

We ship UPS-COD
We ship to APO/FPO

Call the ***Halted Electronic Resource*** BBS!
(408) 732-2814

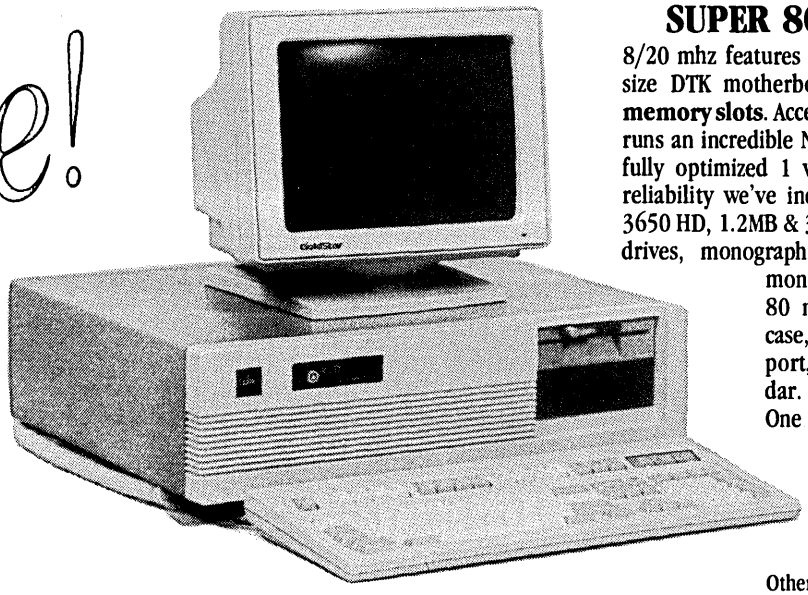
This partial listing of PC-LabCard represents only one of the many products we carry. Send $1.00 for our Giant Flyer!

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

Terms: Minimum order $10. California residents add 7% sales tax. Prepaid orders sent freight C.O.D. or call for charges. Shipping will be added to credit card and C.O.D. orders. $2 handling charge on orders less than $25. Send money order or certified check. Please do not send cash. Some items limited to stock on hand. Prices subject to change without notice. Foreign orders use credit card only.

Quality & Price You Can't Pass Up!

Sale!



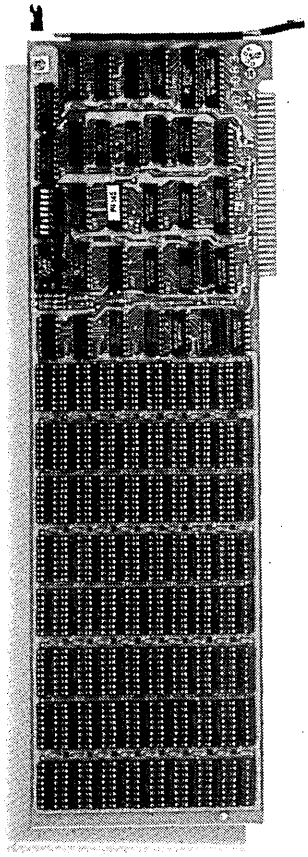
SUPER 80386 SYSTEM

8/20 mhz features an 80386-20 CPU on a full size DTK motherboard with **2 8MB 32 bit memory slots.** Accepts 80387-20. This machine runs an incredible Norton SI test of **24!** Runs a fully optimized 1 wait state. For quality and reliability we've included a 40 MB Miniscribe 3650 HD, 1.2MB & 360K Toshiba or Teac floppy drives, monographics with green or amber monitor, 101 keyboard, 1 MB of 80 ns DRAM, DTK Bios, slide case, 2 serial ports, 1 parallel port, power supply, clock, calendar. FREE assembly and testing. One year warranty.

Now!
$2395!

Other configurations:
tower case, color, etc. ..... $Call
DTK 8 MB RAM Card 99
(32 BIT, 0K)

2 MB EMS MEMORY BOARDS



XT SYSTEMS

Include: 640K RAM, serial/parallel/game ports, clock/calendar, 101 key keyboard, turbo switchable, slide cabinet, power supply, mono graphics with amber/green monitor. 1 year warranty. FREE assembly and testing.

4.77/10 with 2 Floppy 790
1 FD and 1 Miniscribe HD:
4.77/10 with 20 MB HD 995
4.77/10 with 30 MB HD 1010

AT SYSTEMS

Includes: 640K RAM, 1.2 MBFD, 1 360K FD, 40 MB Miniscribe 3650HD serial/parallel/game ports, clock/calendar, 101 key turbo switchable keyboard, slide cabinet, power supply, monographics with amber or green monitor. Full one year warranty. **FREE** assembly and testing.

6/10 mhz 1450
12 mhz 1495
Color options for any kit (includes video card and monitor)
CGA Color 175
CGA/EGA Color 380
VGA (analog) with Mitsubishi monitor 650
CGA/EGA/EGA 480 (Multisync) 450

PC XT & AT

Clock 19
Game 14
Parallel (LPT 1, 2 or 3) 18
Serial Port Card - 1 installed Switchable Com 1, 2, 3 or 4 18
Kit for 2nd SerialPort 18
Multi I/O
Serial/Par/Game 32
2nd Serial Kit 20
Multi Drive Controller 39
Supports 1.44, 720K, 1.2, 360K drives

PC/XT

Floppy Controller 19
Multi-function-1 ser/par/clk/game/2 floppy 47
640K RAM (0K) 25
150 Watt Power Supply 50
Slide case lock, LED 38

AT

200 Watt Power Supply 75
AT/386, Lock, LED 65
Tower AT/386, Lock, LED & 200 Watt ps 239

MOTHERBOARDS

XT/Turbo 4.77/10 79
AT 6/10 Award/Phoenix/DTK Bios 249
AT 6/12 Award/Phoenix/DTK Bios 299
Baby AT 6/12 AMI/DTK 279
80386 8/20 DTK Bios 937
XT/AT Memory ..... $CALL

SOFTWARE

MS DOS 3.21 w/GW Basic 49
DR DOS 3.3 w/GEM 49
MS DOS 3.3 w/GW Basic 95

DISK DRIVES

Teac/Toshiba 360K 69
Teac/Toshiba 1.2 MB 85
Teac/Toshiba 3½" 720K 79
Teac/Toshiba 3½" 1.44 MB kit 90
XT 20 MB Miniscribe
8425 (65ms) 279
8425 w/controller 319
XT 30 MB Miniscribe
8438 (65ms) 299
8438 w/controller 349

DISK DRIVES (Continued)

AT 40 MB MiniScribe 3650 (61ms) 339
AT 40 MB MiniScribe 3053 (25ms) 489
AT 71MB MiniScribe 6085 (28ms) 649
AT (MFM) HD & FD Controller card DTK 110
WD 127
AT RLL HD & FD Controller 189

MONITORS

EGA/CGA (Autoswitch .31 dot) 385
CGA/EGA/VGA MultiSync (.31 dot) 495
CGA Color 249
Amber 12" TTL 89
Green 12" TTL 89
VGA Analog (Mitsubishi .28 dot) 549
Color/Graphics/Par 49
Mono/Graphics/Par 49
CGA/EGA 175
VGA Analog/Digital 249

KEYBOARDS

Enhanced 101 54
Keytronic KB101 67
Focus 101 Tactile, Switchable, Control Caps Lock, Dust Cover 89
(#1 find by MicroC Staff)
★ All keyboards, XT/AT switchable) ★

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

BUILDING YOUR OWN CLONE V2.1
******FREE BOOKLET******

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

Tech Calls: (503) 388-1194

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

XT SYSTEM.. $49
(0K installed)
AT SYSTEM... $99
(0K installed)

MicroSphere INC.
COMPUTERS "HARDWARE MANUFACTURER SINCE 1983"

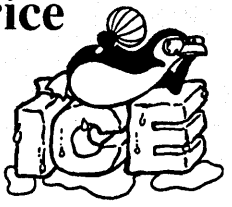
Orders Only: 1-800-234-8086
855 N.W. WALL • BEND, OREGON 97701

Reader Service Number 2

SERIOUS DEBUGGING *at a* REASONABLE PRICE

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

Soft-ICE



Hardware-level break points

REAL-TIME break points on memory locations, memory ranges, execution, I/O ports, hardware and software interrupts. More powerful break points than ANY software-only debugger on the market. Soft-ICE gives you the power of an in-circuit emulator on your desk.

Break out of hung programs

With a keystroke - no external switch necessary. Even with interrupts disabled.

Breaks the 640K barrier

Soft-ICE uses ZERO bytes of memory in the first 1MB of address space. This is especially useful for those subtle bugs that change when the starting address of your code changes. With Soft-ICE your code executes at the same address whether the debugger is loaded or not.

Works with your favorite debugger

Soft-ICE can be used as a stand-alone debugger or it can add its powerful break points to the software debugger you already use. You can continue to use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated.

Solve tough systems problems too

Soft-ICE is ideal for debugging TSRs, interrupt handlers, self booting programs, DOS loadable device drivers, non-DOS operating systems, and debugging within DOS & BIOS. Soft-ICE is also great for firmware development because Soft-ICE's break points work in ROM.

How Soft-ICE Works

Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to provide real-time hardware-level break points.

"Soft-ICE is a product any MS-DOS developer serious enough to own a 386 machine should have."

Dr. Dobb's Journal — May 1988

RUN CODEVIEW IN ONLY 8K!

MagicCV

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

Don't let 640K be your limit!

If you are closing in on the 640K limit and would like the power of CodeView, MagicCV is for you.

Don't let the debugger hide the bug!

Even if you're not closing in on the 640K limit, running CodeView with MagicCV makes your debugging environment much closer to the end user's program environment. You can use CodeView to locate subtle bugs that only occur when there is plenty of free memory, or those difficult bugs that only occur when your program is running with a couple of TSRs loaded.

How MagicCV works

MagicCV uses the 80386 to create a separate virtual machine for CodeView. MagicCV uses between 4K & 8K of conventional memory as a bridge between the DOS environment and CodeView.

MagicCV is easy to use

If you are a CodeView user, you already know how to use MagicCV too. Just type MCV instead of CV; everything else is automatic.

Save $86

MagicCV	$199
Soft-ICE	$386

Buy Both and Save $86!

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

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

NU-MEGA TECHNOLOGIES

P.O. BOX 7607 • NASHUA, NH 03060-7607

Reader Service Number 110

MagicCV with Soft-ICE

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

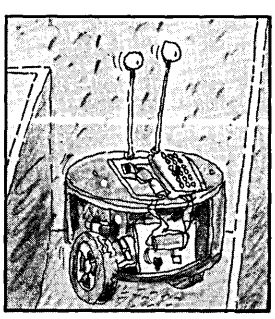
Both require 80386 AT compatible or IBM PS/2 Model 80. MagicCV requires at least 384K of extended memory. CodeView is a trademark of Microsoft Corporation.

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

MAY/JUNE 1989 - ISSUE NO. 47

FEATURES



COLUMNS

FUTURE TENSE

COVER

Cover Illustration by Rob Sanford.

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Editor and Publisher
David J. Thompson

Associate Editors
Gary Entsminger
Larry Fogg
Cary Gatton

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

Director of Advertising & Distribution
Jackie Ringsage

Accounting
Sandy Thompson

Order Department
Tammy Westfall

Graphic Design
Carol Steffy

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

SUBSCRIPTION RATES:

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

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

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

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

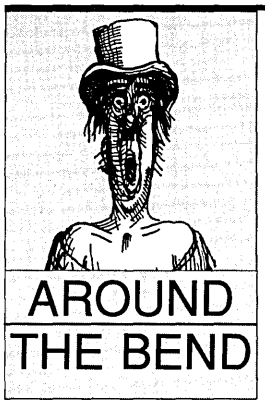
CUSTOMER SERVICE:
For orders and subscription problems call 503-382-8048, 9 am to 5 pm, Pacific time, M-F.

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

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

Copyright 1989 by Micro Cornucopia, Inc.
All rights reserved
ISSN 0747-587X

Audit Bureau of Circulations



By David J. Thompson

Over The Edge

I've gotten into some strange research lately. Unlike most of my tangents, this isn't something new. As usual, I've had my face stuck to some pretty serious books, many of them hot off the presses. But the ideas, let me tell you about the ideas... Galileo could have brought them over on the Mayflower.

I've also begun trying things. Yesterday I attended my first Yoga class and I came out walking on air. Wow, the old body gets heavy after six months parked in front of a terminal. An hour and a half of class had me hearing from parts of my body I'd ignored for years.

"Hello up there, it's me, I feel great." (Obviously, the very best body language.)

Part of Yoga instruction is physical, that's the human pretzel thing. (They don't force you to bend anything that no longer moves, so I'm doing fine.) Another part of Yoga deals with the psyche, with letting go of all the "stuff" we class members have ferreted away (boy, do I have a collection) and with finding our bliss.

Unloading the heavy mental is enlightening, but what about bliss?

Your bliss is what you'd do, given a free choice to do anything. By coincidence, it's also the thing you'd do best. If you follow your bliss, things are supposed to work out, no matter how off the wall it might be. (I don't, however, think they're including the ultimate

Continued on page 70

UserSoft/C Means Business

UserSoft/C Means Business

UserSoft's Business Development Tool Package is a **flexible, high productivity-oriented grouping of run-time libraries and sub-routine functions.**

THE THREE MAJOR COMPONENTS OF THE BUSINESS DEVELOPMENT TOOLS ARE:

1. S/AM

A low level truly Distributed Relational Database Management System.

- May be the most user-friendly run-time library for a truly relational database product.
- No limitations on the number of alternate or secondary keys, with automatic management of alternate keyfiles (virtual tables).
- Mainframe-type data-base security and hierarchy, encryption for files, records, data fields, down to byte level.
- High-language access for non-image data, and mixed (fixed and variable) data field/key length for storage compression.
- No limitations on types and size of data, and the size of the database can exceed the physical limitations of hardware by many times.

2. S/UPERIOR

The first real productivity breakthrough in development of "C" language for the writing of business application software. **The more than 300 routines and extensive run-time libraries in Superior include functions for:**

- Complex matrix operations, which are easy to use, while being very advanced.
- Basic statistical routines
  - — regression
  - — standard deviation as well as functions for conversion and data manipulation.
- Implementation and simplification of best sub-routines.
- Over 100 functions in the format of Basic, Fortran, PL/1 and Cobol, alone.
- Numerous powerful conventions for data and image conversion, and string manipulation, with easy manipulation of screen and window.
- For first time in "C" language, the easy usage and manipulation of financial notations in most currencies.

3. S/CREEN

A screen and window manipulation tool which bridges DOS and UNIX.

- More than 200 screen and window manipulation functions providing capabilities far beyond UNIX *curses.*
- Software has multiple windows and such windows direct screen read/write and auto CGA/Monochrome capabilities.
- Allows multiple window buffers with size exceeding physical screen size, thus programmers can display the view port for development spreadsheets, editors and word processing programs.

UserSoft's S/AM, S/UPERIOR and S/CREEN products are totally integrated and compatible with each other.

PRICE AND AVAILABILITY

The BUSINESS C DEVELOPMENT TOOLS is **available immediately from UserSoft by calling the company's toll-free number: 1-800-663-0322.**

BUSINESS C DEVELOPMENT TOOLS (for DOS computers, consisting of SUPERIOR, S/AM and SCREEN) retails for $299.99.

SYSTEM REQUIREMENTS

For the IBM PS/2™ and the IBM® family of personal computers and all 100 percent compatibles PC DOS (MS-DOS) 2.0 or later 384 K RAM.

USERSOFT

Reader Service Number 148

UserSoft Systems Corporation
Suite 1512, 409 Granville Street, Vancouver, BC, Canada V6C 1T2
Telephone 604/681-8872 Fax 604/682-5450
Sales Order Desk 1-800-663-0322

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 UserSoft products are trademarks or registered trademarks of UserSoft Systems Corporation.
Other brand and product names are trademarks or registered trademarks of their respective holders.
Copyright ® 1988 UserSoft Systems Corporation.

QEdit *Advanced*

the "quick editor"

By SEMWARE

QEdit is one of the smallest, fastest, most easy to use multi-file editors available for IBM Compatible Computers with many features to enhance your productivity.

- ☐ *Extremely Fast*
- ☐ *Completely Configurable (Including Keyboard!)*
- ☐ *Easy to Use Macro Facility*
- ☐ *Optional "Pop Down" Menu System*
- ☐ *Recover Deleted Text*
- ☐ *Edit Dozens of Files Simultaneously*
- ☐ *Utilize up to 8 Windows to View Files*

QEdit *Advanced* requires an IBM PC/XT/AT or Compatible, IBM PS/2 30-80, 128KB of available memory. Please specify 5.25" or 3.5" diskette.

$54.95 MASTERCARD & VISA ACCEPTED

Add $3.00 Shipping & Handling. Georgia Residents Include 4% Sales Tax.

To Order:
SEMWARE, 730 Elk Cove Court
Kennesaw, Georgia 30144
Phone Orders: (404) 428-6416

Reader Service Number 127

Letters

A Rabid SOG Supporter

Sweat dogs from the planet Boron will surely rule if there is no SOG this year. Darkness and chaos, liver flukes, the King's evil, and worse, no doubt. What if we actually *paid* for the seminar like they do in uptown otherplaces?

Jack Pedersen
Sweet Home Chiropractic
P.O. Box 65
Sweet Home, OR 97386

FOG?

I am appalled that you would even consider letting SOG fade away. I have yet to get to one, but to me SOG has been a marvelous vision of magic and sharing. It does my soul good to know that it exists. Do what you must to keep it afloat.

You might consider the example of science fiction conventions which are run with volunteer labor much in the spirit of SOG. They charge $15-$25 typically. I wouldn't mind a charge for SOG, considering what I'd have to spend to get out there.

Maybe it's time to move from SOG to FOG (Fully Official Get-together)?

John Prenis
5339 Knox St.
Philadelphia, PA 19144

Editor's note: Thanks for the letter. We'll be having four regional SOGs; one's practically in your back yard. See the end of the editorial in this issue for the rundown. We'll also be keeping the latest details of all the regionals in the file SOG.TXT on the Micro C BBS — (503) 382-7643, 24 hrs, 2400-1200-300, N, 8, 1. Feel free to download the file, post it at work, and pass it on to other boards.

Former Employee Rebutted

This regards the editorial in *Micro C* Issue #45 in which a now ex-employee of Rotating Memory Service commented on Seagate disk drives. In his last statement he said, "Under no circumstances buy Seagates." I am the owner and was not aware that this interview took place. I believe the technician should have used better judgement.

Maybe say that Seagate makes more drives and that's why we see more of them. They aren't all bad; after all, a disk drive is an electromechanical device. They all seek and spin and all have problems from time to time.

I've spoken to many people at Seagate, but no one seems to want to help; it's not their department or they don't want to make waves. But they refer a lot of potential customers to us, which we welcome.

My personal feeling about Seagate and a few other manufacturers is that they should help support the third party companies. After all, we deal with the end user. The manufacturers should help supply the small guys with factory parts so we wouldn't have to use cross reference parts. They should offer classes that certify technicians for different models. They should provide schematics and tech support (some companies do this already).

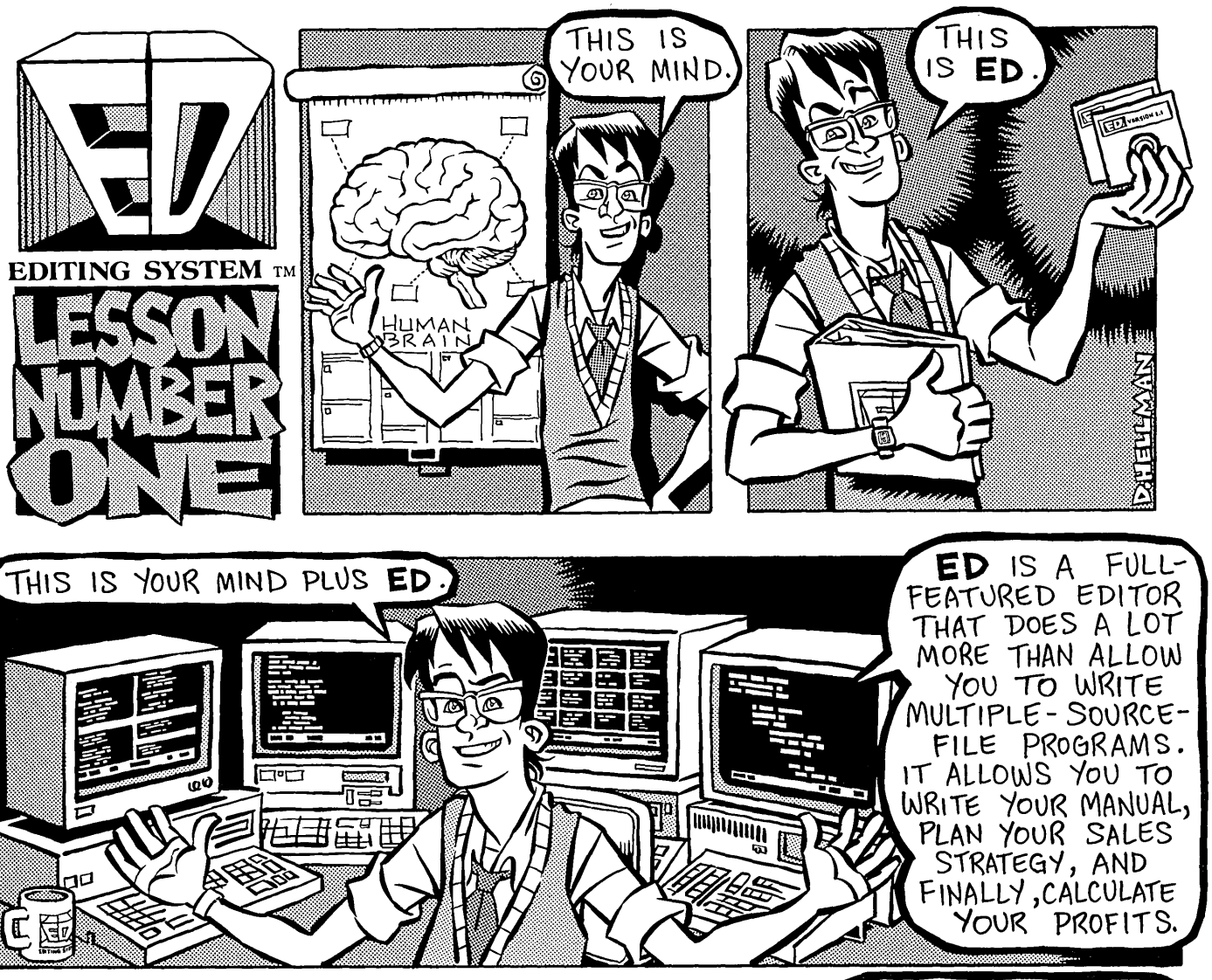
This is my only complaint against Seagate.

Thomas A. Lusi
Rotating Memory Service
473 Sapena Ct. #26
Santa Clara, CA 95054

Individually Wrapped For Your Protection

Perhaps it was just that I finally had time to look at what I was doing, but

Continued on page 68



Enter the world of ED and experience limitless editing power and amazing speed.

ED is an object-oriented open architecture system for editing programs, manuscripts, and data files; and creating sophisticated sorts and filters, customized applications, and snazzy demos.

With ED's Libraries and Source Code, a programmer can manipulate objects such as editors, windows, data entry windows, macros, menus, browsers, popup directories, dynamic arrays, regular expressions, and finite state automata through normal C function calls.

ED • EDIT UNLIMITED NUMBER OF FILES IN MULTIPLE WINDOWS • **POPUP DIRECTORY FACILITY** • VIEW FILESPEC/SUBDIRECTORIES • FIND FILE ON DISK • SEARCH DISK FILES FOR TEXT • BROWSE • FILE STATUS • EDIT • EXECUTE • SORT • **SEARCH AND REPLACE** • FORWARD/BACKWARD • CASE SENSITIVITY • REGULAR EXPRESSIONS • WITHIN BLOCK • ACROSS ALL BUFFERS • **BLOCKS** • COLUMN/LINE/STREAM BLOCKS • SAVE • CUT • DELETE • BLANK • SEARCH-FOR • FORMAT • JUSTIFY LEFT/RIGHT/CENTER/BOTH • UPPER/LOWER CASE • REMOVE • OVERLAY • REACTIVATE • TAB • DRAG • SORT • EXECUTE • **SPREADSHEET STYLE MATH** • ADD/SUB/MULT/DIVD/AVG • **MACROS** • MENU DRIVEN • AUTO-EXECUTE • TIME-DELAY • NESTED • RECURSIVE • INTERACTIVE• MULTIPLE KEYSTROKE • SCREEN DISPLAY CONTROL • AUTOMATIC MENU GENERATION • **MORE FUNCTIONS** • PULLDOWN USER MENU SYSTEM • SELECTIVE UNDO • RECONFIGURE COLOR AND WINDOW STYLE • ETCHA-SKETCH STYLE DIAGRAM/BOX DRAW • TEMPLATE EDITING • COMPILE WITHIN ED • TIME/DATE STAMP • BRACKET MATCHING • GRAPHICS CHARACTERS • DOS SHELL • POPUP ASCII TABLE • USER DEFINED POPUP FILES • KEYBOARD CONFIGURATION • PRINT BUFFER WITH PAGINATION • ON-LINE MANUAL, TUTORIAL, AND HELP • **SUPPORTS** • DOS • PC/XT/AT, PS/2, 386 • CGA, MDA, EGA, HERCULES, VGA, MDS GENIUS, WYSE 700/AMDEK 1280 • RUNS IN ALL VIDEO MODES, NO FLAGS OR DRIVERS • 256K • TURBO AND MICROSOFT C •

Version 1.1 • $265.00 • THE PROGRAMMER'S EDITION (W/LIBRARIES AND SOURCE CODE) $365.00 •

MASTER CARD, VISA, C.O.D. and P.O.'s

THE AMERICAN COSMOTRON 80 HOLMES ST • PO BOX 128 • BELLEVILLE, NJ 07109

Reader Service Number 107

The LIMBO Project, Part 1

I Move Therefore I Am

By Bob Nansel
816 North 105th St.
Seattle, WA 98133

This is Part 1 of the great robot project. You can build the robot yourself or, even better, do it as part of a group. Then you can write software to help this undiapered rug rat run a maze (and freak the house cat).

Bob begins with a history on his mechanical friends and then gives us a peek at his project. I'd hoped he'd be able to give us a complete schematic and mechanical diagram this issue, but parts problems (Murphy) delayed final development beyond the magazine's deadline.

Try to remember the last time you really got excited about a network or the announcement of Desktop Spin Doctor (v1.2). Where did the excitement go? Has the acceptance of the micro in the Fortune 500 destroyed all our fun? I don't think so, because I've found a technology that, while it may become useful in a few years, will never be controlled by a DP manager.

I'm talking about Robots, and I feel confident that these autonomous creatures will supply boundless hacker manna for decades. You can't hook a network to a robot; they move around too much. Because they can't be networked, the marketing types aren't interested. IBM will not call the tune on this one, my comrades.

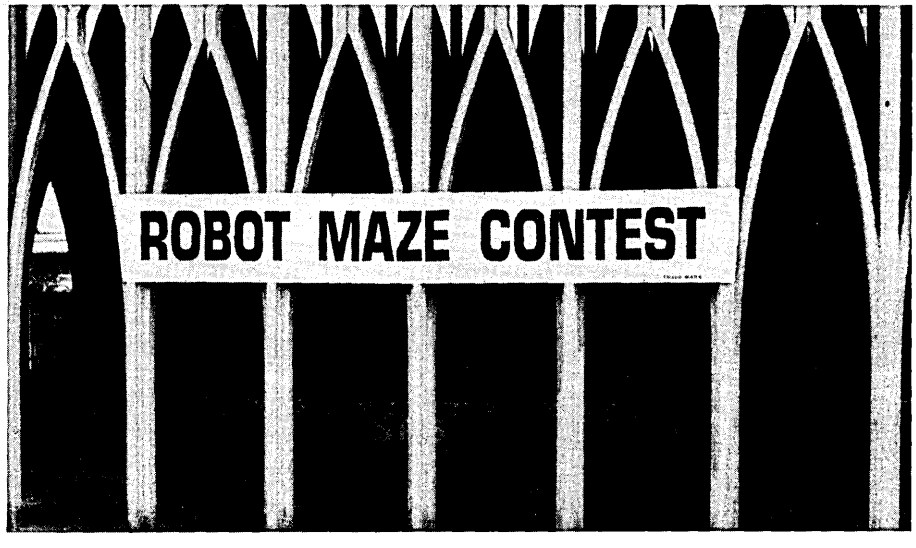
In The Beginning

The dream of building autonomous mechanical creatures has lived fondly in the hearts of tinkerers for centuries, certainly since the dawn of clockbuilding in Europe during the 14th century. We've been building wonderfully curious electromechanical toys for over 200 years, and we've called them Robots since practically the day after Karel Capek, the Czech playwright, coined the term in 1923.

The desire to build Real Robots has been with us much longer than any of our other technical aspirations, except for controlled human flight, and that one was solved 86 years ago.

Though inventors had loads of fun, progress didn't really begin until this century. The biggest advances in the first half of the century included the development of cheap permanent magnet motors and radio control.

Most early robots were nothing more than radio controlled puppets, such as "Mr. Elektro and his dog Sparky." It was popular in the fifties to refer to any automatic electronically controlled device as a robot. Guided missiles were invariably known in the popular press as "Robot Rockets."



These days we distinguish between mere servomechanical devices (slave machines) and real robots. Anyhow, during the 1960s and 70s hackers got valuable experience (fun) playing with servomechanisms and control systems. Solid state RC gear for model airplanes was the big breakthrough during that period.

Solid state was the theme of the sixties in more than just RC. We got infrared emitting diodes along with phototransistors and photodiodes sensitive to infrared.

In The Sixties

Back in the days when women wore dresses and men wore trousers, it became possible to build a robot which could distinguish the sex of a nearby person by measuring the IR energy emitted at shin height; women's uncovered legs emitted more infrared (things are a bit more complicated today).

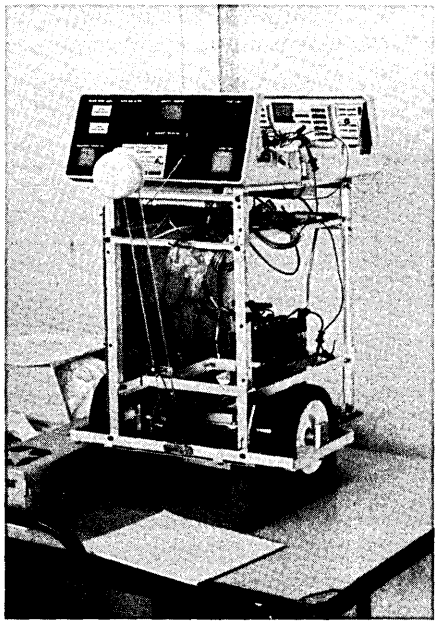
The March 1962 issue of *Popular Electronics* contained an article detailing construction of a simple solid state robot called Emily (Electro-Mechanical Inebriated Ladybug). Emily used two photocells, two motors, a transistor, and

a relay to track a white line on the floor; she could also track flashlights. Subtitle of the article: "The Robot With a One Track Mind." However, during this period, there was a real shortage of practical articles and books.

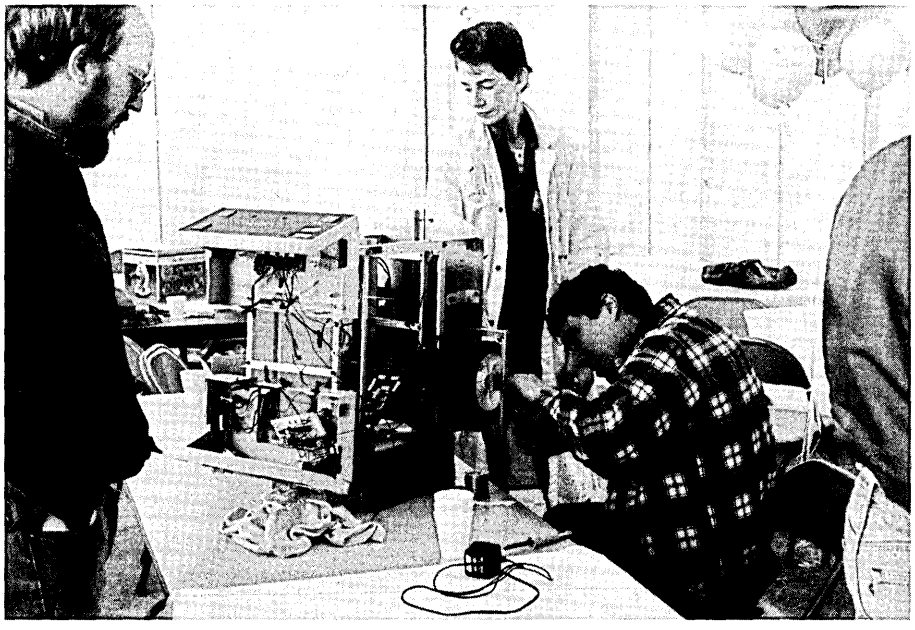
Then The Seventies

The seventies changed that. There were two triggers for the ensuing steady stream of robotic books and articles. First, Hollywood produced two very influential movies: *Silent Running* (1972?) and *Star Wars* (1976).

Silent Running inspired me to build my first robot in 1974 for a science fair project. I patterned it after the movie's wedge-shaped "Drones." I controlled it via audio tones from a cassette tape player. I also used a crude card reader in combination with a reed tone decoder salvaged from a garage door opener. The beast rode on two toy bulldozers and sported a hinged arm with a sheetmetal scoop for an end effector.



Mike Thyng's Robot "MYRT".



Mike Thyng (seated on right) and MYRT. MYRT isn't feeling well after the first attempt at the maze.

The movies started it, but the micro computer revolution made it possible. Now we had the computational power to rule the world. Enthusiasm bubbled forth.

Early Articles

We finally began seeing a groundswell of information on homebrew robots and robot components. *Popular Electronics* began the surge when it published an all-solid-state TV camera project. They called that project Cyclops (published in February '75). They followed up in July '76 with Lou Garner's description of a one chip sonar project.

BYTE ran several robotic articles in their early (better) years: "Designing a Robot from Nature," Feb & Mar '79; "NEWT: A Mobile Cognitive Robot," that same year; Ciarcia's columns "I've Got You in My Scanner!" Nov '78, "Home In on the Range!" Nov '80, "DC Motor Controls: Build a Motorized Platform," May '81. To name a few.

Interface Age (remember?) used to devote its entire August issue to robotics. *Radio Electronics* printed a few too: "Design Your Own Android," Jan & Feb '80; "Build the Unicorn 1 Robot," eleven installments from Aug '80 to June '81; "Build the R-E Robot," thirteen installments from Dec '86 to Jan '88.

And Books!

- *Build Your Own Working Robot*, by David Heiserman, 1976.
- *Build Your Own Working Robot Pet*, by Frank DaCosta, 1979.
- *How to Build your own Self-Programming Robot*, by Heiserman, 1979.
- *Android Design: Practical Approaches for Robot Builders*, by Martin Weinstein, 1981.
- *Robot Intelligence, with Experiments*, by Heiserman, 1981.
- *How to Design and Build Your Own Custom Robot*, by Heiserman, 1981.
- *Apple II/IIe Robotic Arm Projects*, by John Blankenship, 1985.
- *How to Build Your Own Working Robot: The Second Generation*, by Heiserman, 1987.
- *The Robot Builder's Bonanza*, by Gordon McComb, 1987.

(This is by no means a complete list, and I'd be delighted to hear about others).

The Eighties

The eighties were the boom years. Heathkit introduced its educational robot, the Hero-1, in '83, the Hero Junior in '85, and the superb Hero-2000 in '86. Robots even had their own magazines — for a while anyway.

Robotics Age carried ads for: The Arctec Systems Gemini robot, Rhino Robots, Robot-O-Grams, and Cybot. *Robotics Age* was the flagship of personal robotics and articles ranged from how to do your own metal casting to building a remote controlled blimp.

This 36-page, slick mag was a cornucopia of robotics. I started reading the magazine with volume 4 in 1982, but I eventually lost track of it at volume 8 in 1986. Alas, the mag tried to turn professional that year, changing its name to *Robotics Engineering*. They folded soon after.

Homebrew Robotics also tried serving the niche. It appeared in 1986 but didn't last long. Then came *Personal Robotics Magazine* and *Personal Robotics News*. You see, the first generation of personal robotics rode the wave created by the microcomputer revolution. Everyone as-

sumed that robots would come rolling in, hot on the heals of the micro revolution.

A computer in every home? Why not a robot in every home? The slogan of '85 was, "Robots are where micros were ten years ago."

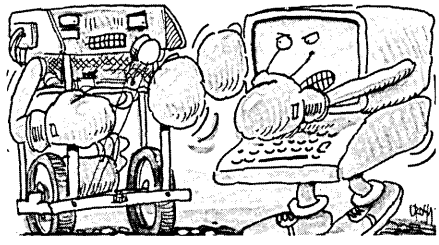
In 1986, Dr. John Billingsley of England's Portsmouth Polytechnic held a Robot Ping-Pong contest. With one-armed robots playing passable ping-pong, what couldn't we accomplish?

The naive comparison between robots and micros created high expectations. Of course, robots proved to be fundamentally unlike microcomputers. It turned out that silicon was a lot cheaper than iron; and unlike micros, the hard problems in robotics wouldn't go away with a wave of the software wand. The shakeout began in '84, and now, in 1989, only Heathkit markets a personal robot, the Hero-2000.

But it's still a fun technology! In fact, I find it more fun now that the hype has died down.

The question I dreaded most used to be, "What does your machine do?" Or, "What is it good for?" Now I have an answer: "It's entertaining and it has a future."

The naive comparison between robots and micros created high expectations. Of course, robots proved to be fundamentally unlike microcomputers.



The first wave of robots was limited by:

- Insufficient processing horsepower and memory;
- Inadequate sensors;
- Lack of high level languages and development tools;
- No operating systems (many used cassette tape "mass" storage);
- No support for realtime multitasking.

We're working on these problems as the second wave begins. Recent improvements in crucial areas fuel the renewed personal robotics movement: integrated silicon sensors, cheap CCD imaging chips, smart power semiconductors, more powerful processors, advanced AI techniques and development systems, vastly cheaper optical shaft encoders and tachometers, high efficiency direct drive motors, and new approaches such as neural network architectures.

Robots are fun and they will stay that way for many years. You can't network a robot nohow, so enjoy.

Editor's note: Maybe they haven't networked a mobile 'bot. Yet. But with a little radio equipment ...

"All I can say about the Gibson Research people is that they did their homework. SpinRite is what the word MUST was invented for."
– Richard Grehan, BYTE MAGAZINE

SpinRite™

"SpinRite describes itself as 'a truly new generation of hard-disk utilities,' which is a marvel of understatement!"
– Stephen M. Leon, MICRO/SYSTEMS

Our users' experiences has shown: If you use SpinRite™ every two or three months, you'll never have ANY PROBLEMS with your hard disk.

The low-level format of your hard disk drive is probably the last thing you want to think about, let alone worry about. But like the foundation of your home, you depend upon it every day without ever giving it a second thought... until something goes wrong.

Every byte of data stored in your hard disk rests upon the drive's low-level format foundation. When that foundation weakens, DOS begins reporting errors:

BOOT FAILURE
SECTOR NOT FOUND
BAD SECTOR ERROR
GENERAL FAILURE READING DRIVE
ABORT, RETRY, IGNORE

That's how your vital data becomes hard to recover or lost forever. ***This problem makes our personal computer hard disk drives the least reliable components in our computers.***

Today you have two choices: Sit around worrying about the safety of your data, backing up the drive continually to minimize the extent of the loss *when* it occurs...

Or cure the problem at its source by preventing your drive's low-level foundation from *ever* weakening and crumbling.

SpinRite completely eliminates the problem of gradual low-level format deterioration by quickly low-level reformatting any DOS hard disk while leaving all its data in place...

But SpinRite goes FAR BEYOND JUST THAT!

SpinRite's Main Features:

SpinRite is an all-in-one, total low-level, format maintenance, repair optimization utility.

- Non-destructively low-level reformats any DOS hard disk drive in minutes. Backup & restore are not required!
- Fully automatic surface defect management utilizing the industry's most extensive worst-case data pattern analysis.
- "On-the-fly" instant sector interleave optimization establishes the maximum possible drive data transfer rate.
- Recovery and repair of correctable and completely uncorrectable (unreadable) data!
- Identification, diagnosis and repair of every form of data and format damage.
- MFM, RLL, ARLL and ERLL compatible.

- In a matter of minutes it gives any DOS drive a completely new, clean, stable and solid low-level format *WITHOUT* requiring a tedious backup & restore operation.
- It detects and eliminates all data-threatening hard disk errors *(which DOS can't see)* long before they become data-damaging.
- It *instantly optimizes* and resets the drive's sector interleave, guaranteeing maximum possible data transfer rates.
- It locates and isolates *all* data-threatening surface defects. *(Two to three times more than ANY other surface testing software!)*

SpinRite is offered with a 30-day money-back satisfaction guarantee. It is *extremely easy to use* with a simple user-interface, on-line help, on-line index and a short 40-page owner's guide. SpinRite is immediately available from:

Gibson Research Corporation
22991 LaCadena, Dept-MC
Laguna Hills, CA 92653
(714) 830-2200

Credit card orders, personal checks and COD orders welcome. Send: $59 plus $2 shipping and handling. California residents please include 6% state sales tax.

Reader Service Number 138

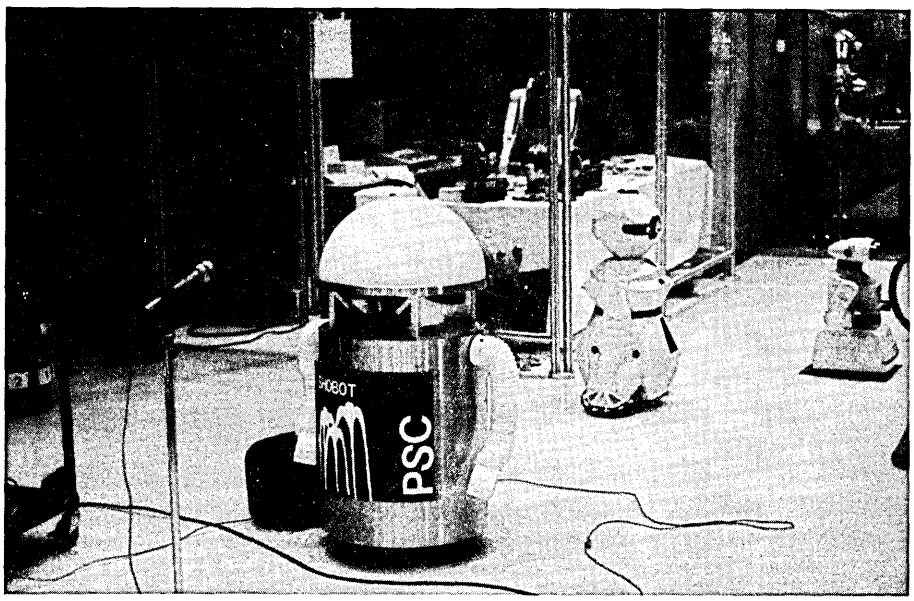
The Seattle Robotics Society

I kept it hidden from all who mattered. I had been a member of the Seattle Robotics Society (SRS) for more than four years, but none of my friends suspected my secret shame — or so I thought.

It all came out of the closet shortly after I became president of the Society last June. Imagine, president of the very club that someday will overrun the world with robot vacuum cleaners and android begonia trimmers. A well educated man (B.S. Robotics Engineering, University of Washington, 1987), an inspiration to tens of budding robot nuts, and rouser of robotic rabble! And now I've been exposed.

A maze contest is a spectacle: part circus, part Grand Prix. Sometimes it's more a matter of which robot survives longest.

Unknown robot. Typical of homebrew robot.



Commercial "showbots" inside the Pacific Science Center. Exhibit of these non-functional sculptures, or "art" robots, coincided with the SRS maze contest.

I had never built a robot mouse to run through the club's Robots-Thru-the-Maze contest. Sure, I had more than my share of advice for those who did enter the maze, but I had an awkward lack of, er ... Hardware ... of my own. So, now it's time to put my servos where my mouth is and build a maze runner.

The Contest

A maze contest is a spectacle: part circus, part Grand Prix. Those Amazing Young Men and their Mechanical Machines battle it out to see which will find its way through the maze in the shortest time. Sometimes it's more a matter of which robot survives longest.

For several years the only machine that regularly made its jerking way through the SRS maze was a mazebot called Murphy. Lance Keizer, Murphy's builder, disdained the use of microprocessors, or even transistors, in his early robots. Two relays wired up to some microswitch bumper contact sensors were all that controlled Murphy.

Murphy would scallop his drunken way through the maze, blissfully unaware of how dimwitted he actually was, while more intelligent microprocessor-based mazebots lost their way and disconsolately spun their wheels. It was embarrassing. I decided to find out why the micros kept losing.

Smart Vs. Dedicated

For one thing, the "smart" mazebots tended to lose their bearings (directional and ball) too easily; Murphy didn't care. Some of the smart mazebots used infrared proximity sensors so they could follow walls without touching them; Murphy detected walls by running headlong into them.

Keizer built Murphy to take this punishment, but Murphy's smarter cousins (who crashed when they lost direction) couldn't. Even robots equipped with sonar-type sensors were easily fooled because smooth walls acted as acoustic mirrors. Moral: non-contact sensing is fine, but bumper contact sensors make a wonderful backup.

I began to compile a list of features a winning smart mazebot should have. I wanted to build a robust robot whose success or failure would depend on the strength of its software, not the infirmities of its hardware.

If we learned anything from Murphy, it was that directness is best, so I resolved that I would take direct approaches to everything. That meant using highly reliable sensors and a simple drive mechanism.

The robot would be steered tank fashion by two stepper motors driving independent wheels. I would use layers of sensors, contact and non-contact.

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers)	$500
	Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
	CQL Query System (SQL retrievals plus windows)	$325
	GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$290
	Greenleaf Data Windows (windows, menus, data entry, interactive form design; specify compiler)	$220
NEW!	Assembler Kit (by John Zarrella; includes listing generator & loader; requires signed license agreement)	$175
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF; specify compiler)	$175
	TurboTEX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTEX)	$170
	Sherlock (C debugging aid)	$170
NEW!	AT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for ATs)	$160
	Greenleaf Functions (296 useful C functions, all DOS services; specify compiler)	$160
	WKS Library Version 2.0 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	ME Version 2.1 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
	TurboGeometry (library of routines for computational geometry)	$125
Updated!	Install 2.0 (automatic installation program; user-selected partial installation; CRC checking; Version 1.0 still $25)	$120
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$115
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$105
	Minix Operating System (U**x-like operating system, includes manual)	$105
NEW!	HyperText Viewer (simple hypertext system; multi-file documents; includes Tiny Curses)	$100
	PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
	Tele Operating System (TeleKernel, TeleWindows, TeleFile, & TeleBTree by Ken Berry)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
	Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	$80
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
	Polyglot Lisp-to-C Translator (includes Lisp interpreter, Prolog, and simple calculus prover)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
NEW!	Kinetic Image Synthesizer (3-D animation system ... Saturday morning on your PC!)	$75
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
NEW!	Heap Expander (use LIM-standard expanded memory as an extension of the heap)	$65
	lp (flexible printer driver; most popular printers supported)	$65
	Quincy (interactive C interpreter)	$60
NEW!	Symtab (general-purpose symbol table construction and management package)	$60
	PTree (general-purpose parse tree construction and management package)	$60
NEW!	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	$50
	Icon-Tools (full-featured icon display and editing system)	$60
NEW!	Polyglot TSR Package (includes reminder, bookmark, virus catcher, cache manager, & speech generator)	$50
	HELP! (pop-up help system builder)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
	Make (macros, all languages, built-in rules)	$50
	Coder's Prolog (inference engine for use with C programs)	$45
	Virtual Memory System (least recently used swapping)	$40
	C-Notes (pop-up help for C programmers ... add your own notes)	$40
	Heap I/O (treat all or part of a disk file as heap storage)	$40
	Biggerstaff's System Tools (multi-tasking window manager kit)	$40
Updated!	OOPS (collection of handy C++ classes by Keith Gorlen of NIH; Version 2.2)	$35
Updated!	Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor; now includes documentation)	$25
	PC-XINU (Comer's XINU operating system for PC)	$35
	CLIPS (rule-based expert system generator, Version 4.2)	$35
	Tiny Curses (Berkeley curses package)	$35
NEW!	Polyglot RAM Disk (change disk size on the fly; includes utilities)	$30
	SP (spelling checker with dictionary and maintenance tools)	$30
	Clisp (Lisp interpreter with extensive internals documentation)	$30
	Translate Rules to C (YACC-like function generator for rule-based systems)	$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
	Pascal P-Code Compiler & Interpreter or Pascal-to-C Translator (Wirth standard Pascal)	$25
	ICON (string and list processing language, Version 7.5)	$25
Updated!	FLEX (fast lexical analyzer generator; new, improved LEX; Version 1.1)	$25
	LEX (lexical analyzer generator; an oldie but a goodie)	$25
	AutoTrace (program tracer and memory trasher catcher)	$25
	Data Handling Utilities in C (data entry, validation & display; specify Turbo C or Microsoft)	$25
	Arrays for C (macro package to ease handling of arrays)	$25
	ANSI Forms (forms manager based on ANSI codes)	$20
	C Compiler Torture Test or Benchmark Package (checks a C compiler against K & R or measures speed)	$20
	A68 (68000 cross-assembler)	$20
	List-Pac (C functions for lists, stacks, and queues)	$20
	XLT Macro Processor (general purpose text translator)	$20
	C/reativity (Eliza-based notetaker)	$15
	Data	
	GenBank DNA Sequences (GenBank 57.0; includes demo disk of Pearson FAST/A programs)	$150
	Protein Sequences (over 10,000 sequences; includes demo disk of Pearson FAST/A programs)	$60
NEW!	Smithsonian Astronomical Observatory Subset (right ascension, declination, & magnitude of 258,997 stars)	$60
	Dictionary Words (234,932 words in alphabetical order, no definitions)	$60
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
	KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
NEW!	King James Bible (Old and New Testaments)	$25
	USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	$20
	NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
	U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

acw!info@uunet.uu.net

Voice: (512) 258-0785
BBS: (512) 258-8831
FAX: (512) 258-1342

Free surface shipping for cash in advance **For delivery in Texas add 7%** **MasterCard/VISA**

Reader Service Number 4

Most important — I decided to endow my maze runner with an internal sense of direction via a solid state compass. Finally, I thought this mazebot should support easy software changes, so I would include a ZIF socket to allow anyone to pop in a new brain (EPROM). *(Editor's note: Everyone should have a ZIF socket.)*

This mazebot would be a boon for those too busy to build hardware; software would be written, burned into EPROM, and raced on my machine. With a level electromechanical field, the race would be won or lost by software.

Software In Limbo

The result of all this cogitation is a design in progress called LIMBO. LIMBO stands for Lost In Maze But Optimistic, and will be cheap enough for almost anyone to duplicate (under $400), but you need only to invest in an EPROM.

LIMBO will include both infrared range detectors and bumper contact switches, so if the IR sensors get confused, the bumpers will save the walls. I'm improving the reliability of the IR sensors, too, using low noise PIN photodiodes with built-in IR daylight filters.

When I've completed LIMBO, we'll hold a competition open to all who want to participate.

Check out the upcoming LIMBO articles, build a LIMBO, write ROMable software, mail in the EPROM, and late summer or early fall the Seattle Robotics Society will conduct a maze contest with all the entries. To make this work, I'll have to present very detailed schematics, drawings and listings. Unfortunately, none of these are ready yet, so this time I'll talk about theory, and next time we'll build machines.

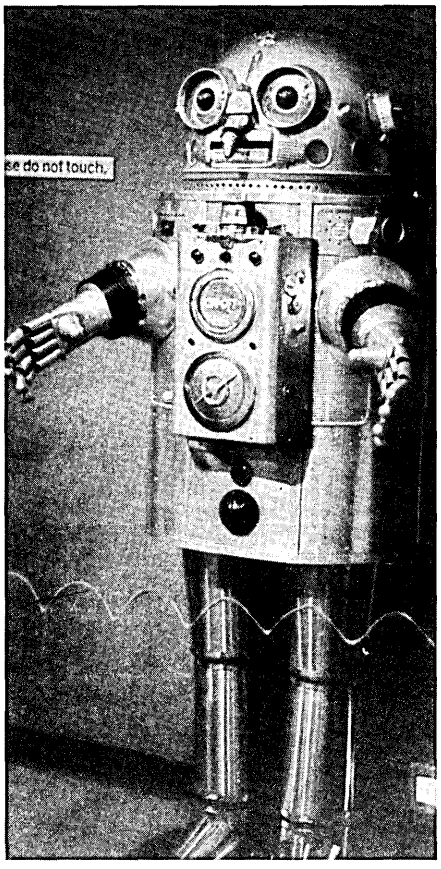
Home, Home On The Range

One of our biggest hurdles is range detection.

The problem with IR proximity detectors, as currently used by fellow robot builders, is that changes in surface reflectivity can be interpreted as changes in distance from the surface.

I've come up with a new twist in IR proximity sensing so I can read the absolute distance to the maze wall, regardless of surface reflectivity.

To understand how I'll do this, consider the way light propagates. As the beam spreads out, the power intercepted in a unit area goes down by the inverse square of the distance from the source. If the beam is reflected back to the source by some target (the maze wall), then the beam traverses twice the distance from the source to the wall.



Commercial Showbot.

The return trip is also subject to this same inverse square fading law. The result is that the radiant intensity received back at a sensor placed next to the original source is the product of these two inverse squares.

The radiant intensity received then is proportional to the inverse fourth power of the distance. I say proportional because other factors — such as surface reflectivity — also affect the reflected light.

Now, if I aim two identical emitter/detector pairs at roughly the same point on the maze wall, the surface reflectivities, transmitted powers, optical cross sections, beam divergences, and overall optical transmission efficiencies are the same for both pairs.

All those factors cancel out (near enough, anyway). This means that if I place the two pairs at a known distance apart aimed in the same direction, then take the readings of reflected power at these two different distances from the wall, I can use the following equation to final the range to the wall measured from the closer of the two sensors:

$$R_1 = R_0/((P_2/P_1)^{.25} - 1)$$

Where:

R_1 = range to wall from sensor 1

R_0 = distance between sensors 1 and 2

P_1 = IR power received at sensor 1

P_2 = IR power received at sensor 2

I haven't tried this out yet, but I see no reason that it shouldn't work over distances of a few feet, which is all I need. The greater the range or the lower the target reflectivity, the lower the signal to noise ratio will be, especially for the farther sensor.

This just means that accuracy goes down the farther away both sensors are from the target. The important thing, though, is the ratio between the two received signals.

The two sensors give more than the minimum information needed to find the range, so I could also calculate the surface reflectivity that the sensors "see." I am not aware of literature describing this method of ranging. If you know of similar techniques being used elsewhere, I'd like to hear about it, as well as any comments on IR ranging in general.

Editor's note: It seems to me you could also factor in the change in signal based on distance moved. For instance, if a small movement creates a significant increase or decrease in signal strength, you can probably assume you're relatively close to the surface and moving reasonably perpendicular to it. Obviously, the program that does the best job of interpreting range data has a significant advantage.

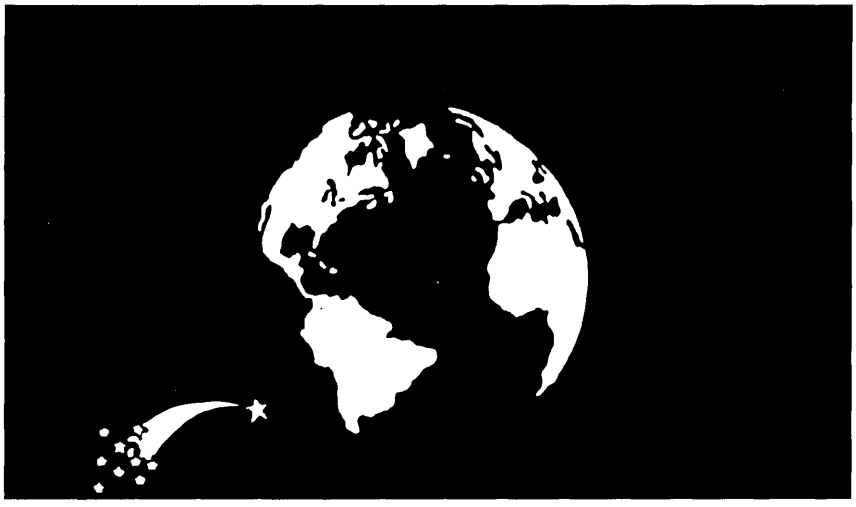
Summarizing, LIMBO will use a version of the Hitachi HD64180 CPU that can address up to 1 MByte of memory so there will be plenty of room for growth. And the highly integrated processor includes MMU, serial ports, DMA controller, timer/counters, and interrupt controller.

A monitor program has already been written in C by another SRS member, Jon Mandrell. I will extend this monitor with device driver routines for the stepper motors, IR rangers, bumper switches, and solid-state compass. So it will be possible to write maze software without hassling with the low-level details.

Next time we'll get down to the whys, wherefores, and wirewrapping of the LIMBO 1 design.

♦ ♦ ♦

You Asked for It All...



And **Tele**™ Appeared

Berry Computers presents The Tele Toolkit — a complete Operating Systems Kernel

Tired of playing games with DOS? Want to get at the internals to make it do what you want, efficiently? Then you need the Tele Operating System toolkit. *Tele is a complete operating system for computers based on any member of the 8086 family of processors.* It contains its own file system which is compatible with MS-DOS media.

Tele contains a preemptive task scheduler with real-time support. *Tasks may be scheduled to execute at specific times in the future.* The precision is determined by the system clock frequency, which may be tailored to each application.

Tele also contains an overlayed window manager to map console display output from several tasks to the system monitor. All display adapters are supported. The window system updates the physical display during the vertical blanking period. This provides flicker-free images on older display adapters. On newer adapters, the amount of time devoted to display update can be precisely limited.

The file system is modular and hierarchal. Device drivers written for MS-DOS may be used. If you write your own drivers, they need only read and write variable blocks at specific addresses. *In Tele, all of the intelligence is in the operating system,* not the firmware. Tele comes with support for MS-DOS media, but other media may easily be included.

Other features include a low level keyboard driver for XT and AT clones. This may be used to define key combinations not supported by the standard BIOS. Tele also contains a basic run time library to support the demonstration/diagnostic programs. There is no need for any other software - although you are free to include object modules from your compiler's run time library.

All source code is included and was developed using Microsoft's version 5.1 C compiler. The C code conforms to the ANSI standard for portability. The assembly code uses an extensive system of macros to hide the details of the specific compiler used. The macros may be used to write assembly code that can be used with any high level language.

A manual with over 450 printed pages comes with the software. The manual gives examples for using all external functions, describes the theory underlying all algorithms, and contains narratives of every subroutine. *Tele is an ideal vehicle for learning about the internals of modern operating system kernels.*

Telephone support is freely available.

The Tele Toolkit is available for $130 from:

Crosby Associates
48 Main Street
Sutter Creek, California
95685

CALL NOW TO ORDER:
(209) 267-0362

Visa, Mastercard, American Express accepted.

MS-DOS is a trademark of Microsoft Corporation.

Reader Service Number 147

How To Build And Use A System Profiler

Profiles have always had a bad reputation — long noses and poor eye contact — but taking a sideways look at your software might show a lot more than dirty ears. Herein Kent shows you how to use his profiler to walk through a Mandelbrot. (And Mandelbrot code does walk.)

Lurking in the dark regions of every programmer's mind has to be the question, "Now that it runs, how do I make it faster?"

You can make a good first step in optimizing your program by running it under the control of an execution profiler. The profiler will show you where your masterpiece spends its time, and let you (in theory, at least) focus on the routines that need more work. In practice, however, the information may not help much.

My first attempt at profiling an MS-DOS program suggests what can go wrong. The profiler I used reported only three categories:

```
strcpy  3%
System  3%
Other  94%
```

The "Other" category, as defined by this particular utility, consisted of anything in the upper regions of the 8086's address space (mostly video output). Further experimentation made me realize it's common for a program to spend most of its time waiting for DOS or the BIOS.

I decided to look at profiling from a different perspective. Instead of classifying timer ticks relative to the location of the code being executed, I chose to measure the time spent in each interrupt service routine. With this information, I was able to produce a picture (such as the one in Figure 1) showing the ways in which an application program uses the operating system.

Figure 1 — Execution Profile of CHKDSK on a 30M Hard Disk

Service	Number of Occurrences	Number of Timer ticks	Percent of Total Time
VIDEO BIOS - INTERRUPT 10H			
02 - Set Cursor Position	14	0	0.00
08 - Read Character and Attribute	14	0	0.00
0E - Write TTY	422	9	2.69
TOTAL	450	9	2.69
DISK SERVICES - INTERRUPT 13H			
02 - Read Sectors	1135	169	50.60
KEYBOARD BIOS - INTERRUPT 16H			
01 - Character Waiting?	210	0	0.00
DOS FUNCTIONS - INTERRUPT 21H			
0D - Reset Disk	2	0	0.00
0E - Set Default Drive	2	0	0.00
11 - Find First File FCB Mode	154	15	4.49
12 - Find Next File FCB Mode	3439	82	24.55
19 - Get Default Drive	5	0	0.00
1A - Set Disk Transfer Area	4	0	0.00
25 - Set Interrupt Vector	31	0	0.00
29 - Parse Filename	6	0	0.00
30 - Get DOS Version Number	3	0	0.00
32 - Service 50	1	0	0.00
35 - Get Interrupt Vector	15	0	0.00
37 - Service 55	3	0	0.00
38 - Get Country-Dependent Info	3	0	0.00
3B - Change Default Directory	306	27	8.08
3D - Open File, Handle Mode	2	0	0.00
3E - Close File, Handle Mode	32	0	0.00
3F - Read from File or Device	4	0	0.00
40 - Write to File or Device	42	2	0.60
42 - Move File Pointer	4	0	0.00
44 - I/O Control for Devices	6	0	0.00
47 - Get Default Directory	7	1	0.30
48 - Allocate Memory	4	0	0.00
49 - Free Memory	2	0	0.00
4A - Modify Allocated Memory	1	0	0.00
4B - EXEC	2	26	7.78
4C - Terminate Process	2	1	0.30
4D - Get Return Code	2	0	0.00
4E - Find First File, Handle Mode	5	1	0.30
4F - Find Next File, Handle Mode	4	0	0.00
60 - Service 96	1	0	0.00
TOTAL	4094	155	46.41
DOS ABSOLUTE DISK READ - INTERRUPT 25H			
	1	0	0.00

Total time: 334 timer ticks
18 seconds

♦ ♦ ♦

By G. Kent Cobb
8233 Lullwater Dr.
Dallas, Texas 75218

Figure 2 — Interrupts Intercepted by SYS_PROF

Function	Interrupt Number	Index into data tables	Number of Defined Services
Print screen	05H	1	0
Video	10H	2	20
Disk	13H	3	24
Communications	14H	4	4
Keyboard	16H	5	3
Printer	17H	6	3
DOS Functions	21H	7	99
DOS Absolute Disk Read	25H	8	0
DOS Absolute Disk Write	26H	9	0

♦ ♦ ♦

Figure 3 — Services Provided by Interrupt 60H and PROFCTRL

Value of AH	Interrupt 60H Function	Corresponding PROFCTRL Command
0	Report ON/OFF status	profctrl status
1	Turn accumulation on	profctrl on
2	Turn accumulation off	profctrl off
3	Return address of counter array in ES:BX	----------
4	Initialize counter array to zeroes	profctrl zero

♦ ♦ ♦

In this article, I'll describe my profiler and suggest some ways that you can use it to optimize performance. I've uploaded the source of the profiler (with detailed documentation) to the Micro C BBS so you can see how I did it.

Editor's note: The package is also available on the Issue #47 disk for $6. Call us at 800-888-8087.

How A System Profiler Works

I used a TSR to intercept MS-DOS and BIOS interrupts. (See the complete list of intercepted interrupts in Figure 2.) For each of these interrupts, the system profiler (SYS_PROF) adds an extra layer of code to the interrupt handler. This lets SYS_PROF:

1. Tabulate the number of occurrences of each interrupt/service combination;
2. Determine at each timer tick which incomplete interrupt/service combination was most recently requested.

The sequence of operations in this added layer varies little from interrupt to interrupt. In general, it:

1. Saves the previous interrupt/service combination on the stack.
2. Filters out any services within this interrupt that we aren't specifically watching. This keeps the data tables small, ensures the integrity of the data, and provides an easy way to handle interrupts with no distinct services (like print-screen, which only does one thing).
3. Changes the values of the variables INTERRUPT and SERVICE.
4. Increments the appropriate counter in the table of occurrences.
5. Executes the real ISR.
6. Restores the previous values of INTERRUPT and SERVICE from the stack.

Since many of the DOS and BIOS interrupts pass information to and from the application program in the 8086 registers and flags, it's important that the extra layer not alter the values of either.

These interrupt handlers are similar, so all nine can be implemented with a single macro, PERFORM_INTERRUPT. The only ones which require special handling are 25H and 26H, which perform absolute disk read and write functions.

For reasons I've never understood, the service routines for interrupts 25H and 26H terminate with a RET instruction (instead of an IRET or RET 2), leaving the flags on the stack.

SYS_PROF also installs a service routine for interrupt 60H, which provides a means of control after SYS_PROF becomes resident. Figure 3 lists the available services, and PROFCTRL (on the Micro C BBS and issue disk) contains the source code for a control program that implements these services.

You can also use SYS_PROF to profile an entire day's activities. Install and activate SYS_PROF in the morning, go through your daily routine, and generate an activity report at the end of the day. (Very effective if you're trying to justify a faster hard disk, more memory, etc.)

The timer tick information generated by SYS_PROF is similar to that produced by a traditional profiler — classification of a program's execution time according to activity type.

The execution counts for different services, however, provide information not normally produced by a profiler. Often, these tallies provide excellent insight into a program's design and methods. I'll try to illustrate uses of both types of information in the examples that follow.

Case #1 – What Hath Mandel Brot?

Let's profile some code, a program based on Larry Fogg's Mandelbrot Set

generator in *Micro C* issue #39. ("Drawing the Mandelbrot and Julia Sets," p. 8.) I've altered Larry's code slightly, but the guts of the test code come from his Figure 2.

The biggest differences between Larry's and the test code are —

1. I modified this version for 320 X 200, 4-color mode, and reduced the maximum number of iterations to 16. (I know all you high-res, 34010 purists retch at the thought, but you're not the one who had to sit through a few dozen test runs. Change a few constants, and you can run it your way.)

2. I used the video BIOS to implement the draw_point routine. I'd expect this to be slower than writing directly to video memory. This way SYS_PROF can tell us how much we can improve the code.

3. Once the program's drawn the entire Mandelbrot set, it'll generate a print-screen interrupt to dump the image to the printer. (GRAPHICS.COM must be loaded for this to work.)

I compiled the test code with Microsoft C V5.1, using the /Ot option to optimize for speed, and the default (/FPi) option for floating point operations. The program will use a coprocessor if it detects one, but runs fine without. For convenience I use batch files to run all the tests. These batch files are also on the Micro C BBS.

For the first test I used a Samsung AT clone, which can operate at 6 or 10 MHz. It has no coprocessor and runs DOS 3.2. I ran the first test at 6 MHz. I've summarized the results in Figure 4.

The DOS EXEC function uses most of the execution time (92%). This may seem peculiar at first, but it makes sense when you consider that this interrupt isn't complete until the program terminates. Timer ticks will accumulate whenever no other interrupts are incomplete. So all the time spent doing floating point calculations accumulates in the DOS EXEC.

Most of the remaining time (8%), the video BIOS routines read and write pixels. Note that writing directly to the screen would have had little effect on execution time — about 3% at best. But that applies only to this program, running on this hardware. As we'll see, the percentage of total time spent in this service can become significant.

The Read Dot services were generated from within the print-screen interrupt by GRAPHICS.COM. Curiously, it has to read 64050 pixels in order to print a screen of 64000. (Fifty cases of "Abort, Retry, Ignore"? Naaaahhh ...) It would be nice to know which pixels it read twice, but I haven't figured that out.

Figure 4 — Partial Results of Mandelbrot on 6 MHz AT Clone

Service	Number of Occurrences	Number of Timer ticks	Percent of Total Time
PRINT SCREEN - INTERRUPT 5H			
	1	74	0.26
VIDEO BIOS - INTERRUPT 10H			
0C - Write Dot	64000	1038	3.68
0D - Read Dot	64050	1052	3.72
KEYBOARD BIOS - INTERRUPT 16H			
00 - Read Character	2	0	0.00
01 - Character Waiting?	245	0	0.00
PRINTER BIOS - INTERRUPT 17H			
00 - Send Byte	48310	57	0.20
DOS FUNCTIONS - INTERRUPT 21H			
08 - Keyboard Input No Echo	2	0	0.00
4B - EXEC	2	25979	91.98
TOTAL	189	25982	91.99

Total time: 28244 timer ticks
1551 seconds

♦ ♦ ♦

Figure 5 — Partial Results of Mandelbrot on 10 MHz AT Clone

Service	Number of Occurrences	Number of Timer ticks	Percent of Total Time
PRINT SCREEN - INTERRUPT 5H	1	34	0.20
VIDEO BIOS - INTERRUPT 10H			
0C - Write Dot	64000	609	3.66
0D - Read Dot	64050	610	3.66
KEYBOARD BIOS - INTERRUPT 16H			
00 - Read Character	2	0	0.00
01 - Character Waiting?	58	0	0.00
PRINTER BIOS - INTERRUPT 17H			
00 - Send Byte	48310	70	0.42
DOS FUNCTIONS - INTERRUPT 21H			
08 - Keyboard Input No Echo	2	0	0.00
4B - EXEC	2	15284	91.80
TOTAL	189	15285	91.81

Total time: 16649 timer ticks
914 seconds

♦ ♦ ♦

The program spent very little time in the printer BIOS (my printer has a 6K buffer). The number of bytes sent to the printer is roughly 3/4 the number of pixels on the screen, so I suspect that one screen pixel translates into six printer pixels.

Case #2

Since the first test indicated a calculation-bound program, I addressed the problem directly — I switched the Samsung to 10 MHz. The only change in the batch file was that break was turned OFF. (More about this in a minute.) Figure 5 contains the results.

No surprises here. The time-consuming services took about 60% of the time required in Example 1, with the predictable exception of the printer BIOS services.

Nor are there many surprises in the

From C To Shining C

Greenleaf Has

The Best Reputation

Over the last five years the name Greenleaf has become synonymous with quality software.

As a leader in the professional C programming library market, we are recognized by the industry in general, and our customers in particular, as the premier supplier of quality products that consistantly lead the industry in performance.

Our customers rate our products, support and documentation EXCELLENT. It's a fact that almost 50% of our clients bought one or more of their Greenleaf products without ever considering the competition.

Greenleaf's dedication to the attainment of excellence is driven by our customers. They tell us what they need and want in order to perform their jobs as professionally as possible. It is this partnership in excellence that has made Greenleaf the most respected provider of C language libraries in the world.

If you don't own a Greenleaf product, talk to one of our users. We suspect that you will be a Greenleaf owner soon. If you already own Greenleaf products we thank you for your trust and support and we look forward to our continued partnership.

GREENLEAF Software©

Greenleaf Software Inc.
Bent Tree Tower Two - Suite 570
16479 Dallas Parkway
Dallas, TX 75248

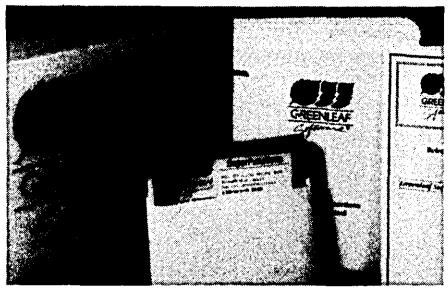
1-800-523-9830

Texas and Alaska: (214)248-2561
Telex: 559-068
Reader Service Number 146

The Best Products

Greenleaf SuperFunctions

An advanced collection of functions for the experienced C programmer. Functions include expanded memory support, extended mouse functions, advanced time manipulations, unique date functions, DOS critical error handler along with windows, menus and keyboard functions.



Greenleaf Business MathLib

The only product available that gives the programmer a complete set of math functions while providing the accuracy of BCD math to the C language. Business functions such as compound interest, internal rate of return, cash flow analysis, bond calculations, amortization, array processing, statistics, trigonometry and print flexibilty make financial output a breeze.

Greenleaf DataWindows

This easy to use and powerful windowing C library features overlaid logical windows, transaction data entry and three styles of menus. Functionality such as context sensitive help, menus within data entry, keyboard idle, list boxes and ROM BIOS redirection. Pop-up, pull-down and Lotus style menus are also included.

Greenleaf MakeForm

Working in conjunction with DataWindows, MakeForm will reduce the time it takes to design and build a human interface. Forms stored in special files enable you to change them without recompiling your application. Cut and paste between multiple forms while using the fully supported editing features.

Greenleaf Comm Library

XMODEM, XMODEM CRC, RTS/CTS, and XON/XOFF protocols are all available on up to 17 ports with this interrupt driven, asynchronous communications library. Your programs can fly at up to 19,200 buad.

Greenleaf Functions

The original. Still as popular and powerful as it was 5 years ago, Over 300 functions to support your programming efforts. DOS and BIOS calls, string color text, time, date, and graphics functions are just the beginning.

Greenleaf ViewComm

Turn your PC into a serial data analyzer. This combination of hardware and software lets you watch live RS-232 communications. Send data from file or keyboard, set "triggers" to control monitoring and capturing of data. A must for COMM debugging.

CONSULTANTS
PROGRAMMERS
ANALYSTS get

Dis•Doc™

the most advanced
SOURCE GENERATING
DISASSEMBLER AVAILABLE
for the IBM PC/XT/AT/PS2/compatibles

DIS•DOC can help you

- find and fix bugs in any program
- re-create lost source code
- a great companion utility to compiler and assembler
- able to give you a great source of professional programming examples.
- And Much Much More!

DIS•DOC is:

FAST

- Disassembles files up to 500kb in size OR RAM/ROM memory at a rate of 10,000 lines per minute.

ACCURATE

- Uses seven passes and over 20 SECRET algorithms to separate code from data to make the most accurate listing on the market.

FLEXIBLE

- Creates a MASM ready listing for easy re-assembly of your disassembly and only needs 320kb of memory.

TECHNICALLY ADVANCED

- The only disassembler that disassemles all instruction sets including 80386/80387.
- And has a built-in patcher to make changes without having to re-assemble.

DIS•DOC also includes BIOS labeling and its own word processor in its package.

DIS•DOC is a proven programmer's tool and is simply a must for the consultants, programmers or analysts.

We CHALLENGE you to find anything that can beat **Dis•Doc!**

DIS•DOC $99.95
EXE Unpacker $29.95
FREE DEMO DISK
add $4.00 for S&H in the USA
$10.00 for outside USA
30 day money back guarantee

To order or get more information
CALL 800-446-4656 today!

RJSwantek, Inc.
178 Brookside Road
Newington, CT 06111
(203) 560-0236

MasterCard VISA

Reader Service Number 142

Figure 6 — Mandelbrot on 4.77 MHz XT Clone with Coprocessor

Service	Number of Occurrences	Number of Timer ticks	Percent of Total Time
PRINT SCREEN - INTERRUPT 5H	1	162	2.98
VIDEO BIOS - INTERRUPT 10H			
0C - Write Dot	64000	1050	19.34
KEYBOARD BIOS - INTERRUPT 16H			
00 - Read Character	27	0	0.00
01 - Character Waiting?	255	1	0.02
PRINTER BIOS - INTERRUPT 17H			
00 - Send Byte	16156	212	3.90
Undefined services	1	0	0.00
DOS FUNCTIONS - INTERRUPT 21H			
4B - EXEC	3	3837	70.66

Total time: 5430 timer ticks
298 seconds

♦ ♦ ♦

Figure 7 — Partial Results of TRANSFER on 10 MHz AT Clone

Service	Number of Occurrences	Number of Timer ticks	Percent of Total Time
VIDEO BIOS - INTERRUPT 10H			
0C - Write Dot	64000	1026	74.24
PRINTER BIOS - INTERRUPT 17H			
00 - Send Byte	16156	31	2.24
DOS FUNCTIONS - INTERRUPT 21H			
4B - EXEC	2	266	19.25

Total time: 1382 timer ticks
75 seconds

♦ ♦ ♦

Occurrences column. The only difference is in the "Character Waiting?" service of interrupt 16H. This has dropped from 245 to 58, a decrease of 187, as a result of turning off BREAK.

The interesting aspect of this change is that 187 is very close to 189, the number of interrupt 21Hs that were generated. Even more suspicious is that the decrease equals the number of non-keyboard INT 21Hs. (245-58 = 189-2.)

The impact on the execution time was negligible, since relatively few DOS interrupts were generated. This isn't always the case, though. I've seen situations where turning BREAK on added as much as 5% to the total execution time. (That's a great situation to walk into. You can make a program run faster with very little effort and without spending any money. You can pretend to be a genius for at least 30 seconds.)

Case #3

Having milked as much performance as possible from the Samsung, I moved to a different computer for the third test. Computer #2 is a 4.77 MHz Corona XT with an 8087. It also differs from the Samsung in that it runs DOS 2.1 rather than 3.2.

This test is complicated by the fact that the Corona has no printer. Now, you're probably wondering why I couldn't move the printer from the Samsung. I suppose I could ... but that's a hardware solution, and this is a software article. Instead, I chose to use LPTX.COM, a printer redirection utility written by Mark DiVecchio. (It's available

from Micro C on disk #MS27.) This utility intercepts the printer interrupts and writes the data to a disk file.

The most obvious difference was the decrease in computation time (see Figure 6). The floating point calculations took 1/4 the time. As a percentage of the total execution time, EXEC dropped to 71%, and the video BIOS dot-writing service rose to 19%. (Note that in absolute terms, the time taken to write the pixels is very similar to that in Example #1 — 1050 timer ticks vs. 1038.)

Note two other major differences, compliments of GRAPHICS.COM. First, there were no occurrences of the video BIOS read-dot service, indicating that this version reads the pixels directly from the screen. This probably accounts for the increase in time spent in the print-screen interrupt.

Second, the number of bytes printed decreased to about 16000, which indicates (I believe) that each on-screen pixel translates into two pixels on the printer, rather than the six in Examples 1 and 2. (This is consistent with the size of the images printed.)

A few minor differences in this test are mildly interesting. The undefined printer service is probably an LPTX-induced phenomenon. (It's not unusual for such a program to define a new service, which it uses to determine its status.)

LPTX also creates the extra "Read Character" interrupts. Comparing the complete reports (available on the bulletin board) shows several small differences in the types of DOS and video BIOS services used by COMMAND.COM, as well.

Case #4

The obvious follow-up to test #3 would be a program that reads the file written by LPTX, displays it on the AT, and dumps it to the printer. The simplistic approach (transferring the file to the printer with the DOS COPY command) won't work because of the embedded Ctrl-Zs in the file.

MNDLBROT and LPTX see these as bit patterns, but DOS interprets them as EOF markers. Even if this approach did produce an acceptable printout, it still wouldn't display the results on-screen.

Fortunately, the file format isn't complicated, so it's fairly easy to write a program to display and print the image. You'll find the source code for this program, called TRANSFER, on the Micro C BBS.

As in previous tests, it took about 60 seconds to display 64000 pixels (see Figure 7). Unlike the other tests, this accounted for almost 75% of the execution time. I can't draw any conclusions about the printing time, since the numbers are relatively small. (It might be informative to rerun tests 1, 2, and 4 without the print buffer. Maybe another time ....)

The time spent in EXEC consists primarily of extracting pixels from the file before displaying them. Adding the execution times for tests #3 and #4, we see that we can display and print the Mandelbrot set in about 40% of the time required in test #2.

Wrap Up

My cases have been simple exercises to give you a feel for profiling. More complex programs lead to more dramatic profiles. Try profiling your own code — EMS, mouse, and floating-point-emulation interrupts, for examples. You might be surprised by the results.

♦ ♦ ♦

CAPITAL

DIAGNOSTICS

The Complete Diagnostics Solution for Your PC/XT, PC/AT, or Compatible

INCLUDES...

DRIVE TESTS—Complete diagnostics for Hard and Floppy drives, including controller cards. Tests read, write, and format capability as well as seek timings, hysteresis and rotation timings.

I/O PORTS—For both parallel and serial ports, confirms internal and external loopback capabilities at all baud rates and configurations.

MEMORY—Performs over eight different tests to check standard, extended, and expanded memory.

KEYBOARD—Verifies that all keys send correct key codes, including shift, CNTL, and ALT modes.

CPU & NUMERIC COPROCESSOR—Verifies that all single and multiple instructions perform correctly and accurately, as well as testing all internal registers.

VIDEO DISPLAY—Checks video controller cards. Confirms attributes, graphics, colors (if applicable), and CRT alignment patterns.

REAL TIME CLOCK—Verifies correct timing, all internal registers, and battery backed-up RAM.

...and many more features to insure the integrity of your computer.

PC/XT System Diagnostic Software	**$ 29**
PC/XT Disk Diagnostics (w/ test diskettes)	**$ 29**
PC/XT I/O Loopback Test Plugs	**$ 19**
COMPLETE PC/XT DIAGNOSTICS SET (save $28)	**$ 49**
PC/AT System Diagnostic Software	**$ 29**
PC/AT Disk Diagnostics (w/ test diskettes)	**$ 29**
PC/AT I/O Loopback Test Plugs	**$ 19**
COMPLETE PC/AT DIAGNOSTICS SET (save $28)	**$ 49**
BOTH PC/XT and PC/AT SETS (save $75)	**$ 79**

Capital Software presents the definitive disk-based diagnostics package for the IBM PC AT and XT. Atechnical tool detailed enough for the repair technician. A friendly interface that places problem-solving skills in the hands of the end user. An uncompromising solution.

SEND CHECK OR MONEY ORDER TO
CAPITAL SOFTWARE
951-2 OLD COUNTY ROAD SUITE 224
BELMONT, CALIFORNIA 94002
FOR INFORMATION CALL:
(415) 592-9076

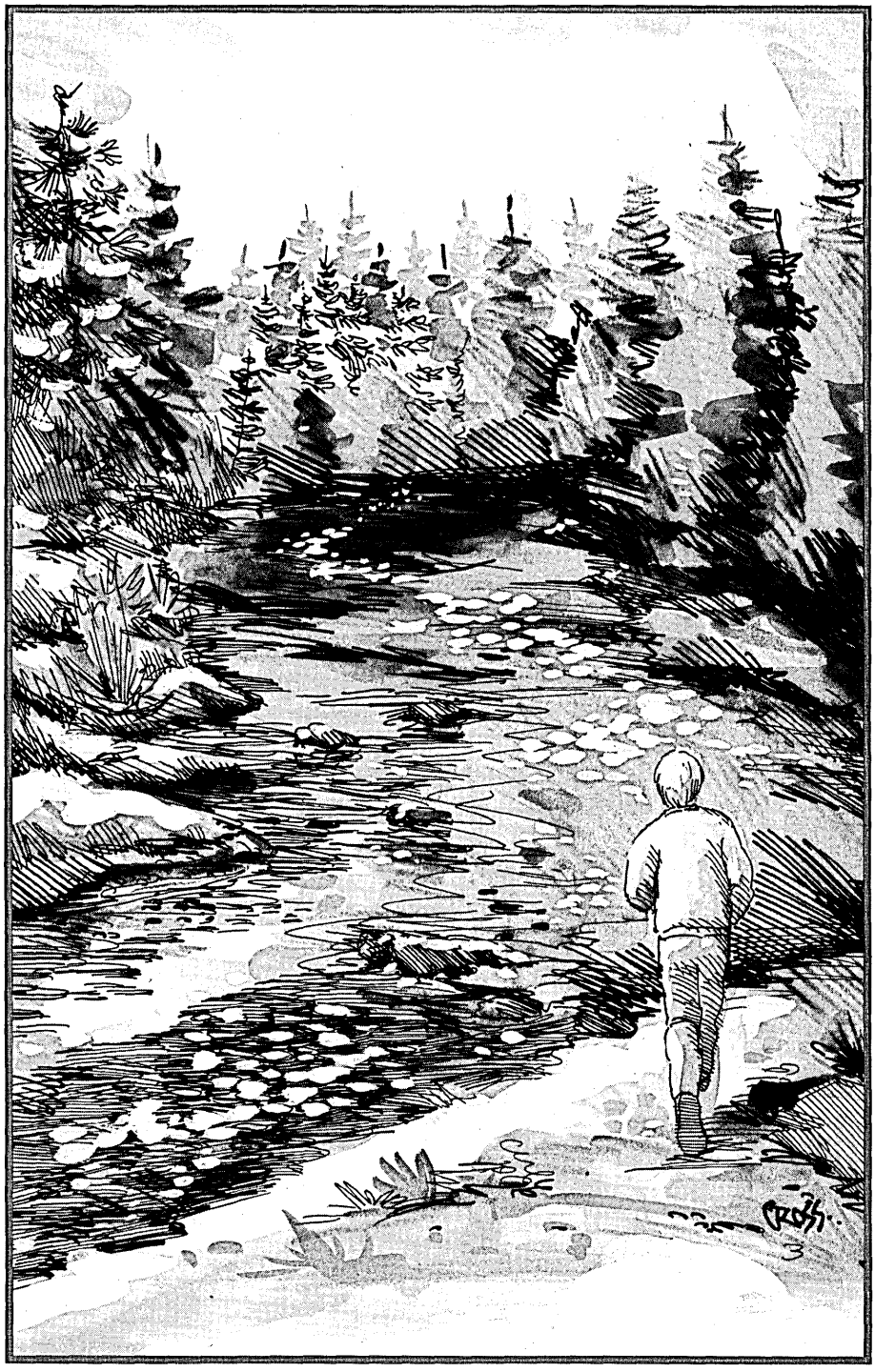
USE YOUR VISA OR MASTERCARD
CALL TOLL FREE (800) 541-0898

VISA MasterCard

Problem Solving And Creativity

A Look Behind The Scenes

If you were hoping this would be something useful like practical fractals — I'm sorry. If you wanted another travelog — Sorry again. However, if you're looking for the soul of creativity, then by all means, read on. (After you finish, you'll only need a mirror.)



It only takes three or four drop-kicks to seriously reduce the effectiveness of a keyboard. I'd just finished kick number seven. To hell with programming!

I shut down the system, put on my warmest sweats, and bolted out the door for a run through the half-foot of Central Oregon's New Year snow. The first couple of miles couldn't still my frantic thoughts. One potential bug after another trampled through my mind leaving large cold footprints, but no solutions.

Mile three and still no luck. But the logical battles faded away — banished to some dark, dank cerebral filing cabinet. As I followed the Deschutes River upstream through Ponderosa stands, the rhythm of the run and the beauty of the river's flashing white water and snowy banks took over. Nothing then but the moment. Body active, mind at rest — peace...

Then, with no warning: The Answer.

The Element Of Surprise

These creative flashes always come as a most happy surprise. They appear during runs, while idly swapping lies with a friend, after a few days sequestered in a snow cave, or simply while quietly paying attention to nothing in particular. In similar situations, we've all had the answer to some pressing question sneak up behind us and demand attention.

Jerome Bruner goes so far as to define a creative act as one which produces "ef-

By Larry Fogg
Micro C Staff

fective surprise."[1] I like his definition. In a very slippery area (thinking about thought), it nails down one thing that most people can agree upon as an element of creativity.

Bruner's surprise signals the end of an aspect of problem solving that may seem foreign to the technically oriented mind: the intuitive, irrational element. But artistic *and* scientific history overflows with accounts of this moment of creative surprise.

Mathematician Jules Henri Poincaré: "One day, as I was crossing the street, the solution of the difficulty which had brought me to a standstill came to me all at once."

The poet Amy Lowell once said that she had no idea how her poems came about. "What I do know about them is only a millionth part of what there must be to know. I meet them where they touch consciousness, and that is already a considerable distance along the road of evolution."

Friedrich Nietzsche too describes an involuntary process. "One hears — one does not seek; one takes — one does not ask who gives: a thought flashes out like lightning, inevitably without hesitation — I have never had any choice about it."

And Mozart: "When I am ... completely myself, entirely alone ... or during the night when I cannot sleep, it is on such occasions that my ideas flow best and most abundantly. Whence and how these come I know not nor can I force them."

"Imagination is more important than knowledge," according to Einstein. Al and I think alike on this. Not that I'm comparing myself to Einstein; he had a much better mustache. But the creative, imaginative problem solving process shows great consistency over a wide variety of individuals.

Historically, western civilization has rebelled against mystical, nonlinear modes of thought. Scientific method, logic... That's where the answers come from. But problem solving can and does happen at decidedly illogical levels. Innovative thought demands another level of understanding where logic and intuition work together towards solutions.

> "I think and think for months and years. Ninety-nine times, the conclusion is false. The hundredth time I am right."
> Albert Einstein

"The rational part of research would, in fact, be useless if it were not complemented by the intuition that gives scientists new insights and makes them creative."[2]

Creativity Research

Studies of creativity often break the process into four distinct stages —

Preparation. The flash of insight can't occur without a lot of initial work. While answers may come in an instant, a great deal of groundwork has probably paved the way.

Incubation. An appropriate term. Ideas ferment without conscious effort or intervention. I run. You settle into your favorite rocking chair on the porch and watch the day go by. Whatever the method, we no longer worry the problem. We let it go.

Illumination. Aha!

Verification. This solution that appeared from out of the blue — does it make any sense? Back to logic now. Let's test the solution and spruce it up if necessary. Often an insight gained through this path is incomplete, or just shows a starting point.

And like all answers, it might be wrong. Creative insight carries no guarantee of correctness. As Einstein once said, "I think and think for months and years. Ninety-nine times, the conclusion is false. The hundredth time I am right."

Soothing The Savage Beast

Not so long ago Bruce, Melinda, Gary and a ragtag group of hangers-on spent a weekend in Bend skiing, discussing C++, arguing writing styles, trashing my house, and playing guitar.

Few pastimes give me more pleasure than picking out a tune with Gary. The communication between musicians absolutely defies any kind of logical explanation. It's a fine example of the creative process.

Again, surprise slides into the act. In the middle of a piece we've played countless times, we change direction. The song takes on a new flavor that neither of us planned. In a flash we've jumped from the ordinary to the unusual.

What a joy; I'll take unusual over ordinary any day. An obvious question: how can we encourage these moments of creativity and insight?

Enhancing Creativity

We can talk about two things here: how to encourage incubation of a particular idea, and how to enhance creativity in general. Let's incubate first.

This article began by relating an example of my own problem solving process. Crafting a bowl from an exotic hardwood, skiing down a high Cascade ridge with only the grey jays and bunnies for company, or 15 minutes in a head stand... Like running, all these have the effect of shifting my conscious attention from a problem.

That's the key. Distract the conscious long enough for the subconscious to get

involved. You can urge that detached, freewheeling subconscious towards participation in a variety of ways.

Don't Worry, Be Happy

Here's one of my favorites. Studies at the University of Maryland-Baltimore County have shown that laughter promotes creativity.[3] Alice M. Isen and associates presented a problem in need of a creative solution to two groups of students. Earlier, each group had viewed a different film — one of TV bloopers, and the other a mathematics film called "Area Under a Curve." (Oh boy!)

Out of the good-time blooper watchers, 75% solved the problem. But only 20% of the morose mathies came up with the answer. So it looks like a happy disposition breeds creative thought.

Isen believes that, since positive memories seem to be retained and interconnected more easily, "...being happy may cue you into a larger and richer cognitive context, and that could significantly affect your creativity."

Creative Boozing

Since some of our most creative artists and writers have been known to take more than an occasional nip from the 'shine jug, we might guess that alcohol could lubricate the creative machine. Not necessarily so, according to psychologist Geoff Lowe.[4]

Lowe studied a group's creativity before and after a round of vodka and tonics. He found that those participants with low initial scores on a standard creativity test did indeed improve after the V&T. But the normally creative folks lost their creative edge when inebriated.

I'd classify *Micro C* readers as a pretty creative bunch. So alcohol ain't the way.

Editor's note: I'll drink to that.

Creativity Down The Road

Exercise, besides fostering the incubation of a particular idea, can improve overall creativity. Joan Gondola and Bruce Tuckman studied the effects of exercise on three groups of college students.[5]

Two of the groups jogged twice a week while the third loafed. They tested each of the three groups before and after the six-to-eight-week regimen. The joggers showed a significant improvement in creativity after completing the exercise course. And as a group they outperformed the sluggards.

Editor's second note: Apparently something had jogged their memories.

Just about anything that yanks you out of familiar patterns will act as grist for the creative mill. Like travel. Some have suggested that I spend an inordinate amount of time on vacation. But I see these jaunts as providing the experiences necessary to fuel my imagination. The importance of variety and breadth of experience can't be overemphasized.

Bound to your home by such mundane considerations as finances or responsibilities of family and work? Read. Travel in your mind. An author can do little more than provide a framework for the imagination. You fill in the details. So reading both takes you to new places and exercises your creativity once you get there.

Programmers and engineers may tend to become detached from nontechnical life, and even from technical areas outside of their immediate interest. This specialization of interest and experience has a stifling effect on creativity.

While you might think that constant grinding on a problem produces the quickest solution, just the opposite is true. Certainly, plenty of preparation and effort must go into the process. But if the unusual, the seemingly unrelated, don't become part of the effort — if the subconscious never gets a chance to work on the problem — then you may work far too long and end up with a mediocre solution.

Does this mean you should read the collected works of Shakespeare, chant yogic mantras, or climb the nearest mountain when a problem's got you down? Well, yes. Perhaps not these particular activities, but varied experience breeds unconventional and creative responses to any problem.

And who knows when experience of questionable value might become useful? One of my most consoling thoughts while suffering through Complex Variables in school was, "There's no such thing as useless knowledge." True, true. Complex math reared its ugly head fifteen years later in the form of the Mandelbrot Set — now one of my favorite playthings.

Editor's next note: Obviously Larry has found an imaginative way of proving that complex variables are both useful and impractical.

Hallmarks Of Creativity

What makes creative individuals tick? They seem to:

- prefer the unusual, irregular, and complex in nature and art;
- have a wide range of interests and activities;
- show a high degree of independence (Q: Why did the creative person cross the road? A: Because someone told her not to.);
- be highly active, productive, and energetic;
- like to push themselves to the edge — to take risks;
- find motivation in the joy of their work, not in any form of reward.

Some of these traits lie under the control of the individual, which implies at least a limited ability to foster creativity. So experiment on yourself. Paying attention to these characteristics may lead you to more creative paths.

Adieu

I like to lump creativity and problem solving together. Not because they necessarily require each other, but because the best (read most fun) solutions are the creative ones.

Any fool with enough perseverance can solve a problem — I've proven that often enough. But a truly innovative approach to a solution raises the level of the game, makes it more fun to play. And, "fun's where the fair's at..."[6]

I leave you with a few choice words of wisdom.

"There is a correlation between the creative and the screwball. So we must suffer the screwball gladly."

-Kingman Brewster

"No one has ever had an idea in a dress suit."

-Sir Frederick G. Banting

References

1 Bruner, Jerome S., "The Conditions of Creativity," in *Contemporary Approaches To Creative Thinking*, ed. Gruber, Howard E. et al (New York: Atherton Press, 1964), LC# 62-19403

2 Capra, Fritjof, *The Tao of Physics* (New York: Bantam Books, 1977), ISBN 0-533-10868-9.

3 Russo, C., "Crosstalk," *Psychology Today*, Vol. 21, September 1987, p. 21.

4 Roberts, M., "Crosstalk," *Psychology Today*, Vol. 20, September 1986, pp. 68-9.

5 Greene, C., "Crosstalk," *Psychology Today*, Vol. 20, October 1986, p. 75.

6 The Firesign Theatre, *I Think We're All Bozos On This Bus* (New York: Columbia Records, 1971), C 30737

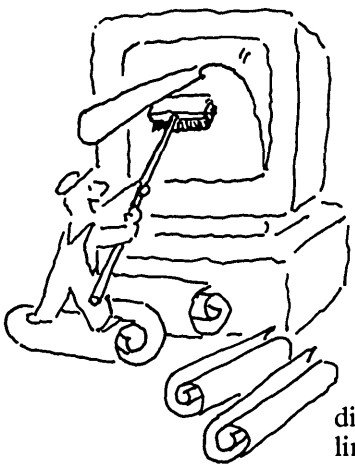
Weisburd, S., "The Spark: Personal Testimonies of Creativity," *Science News*, Vol. 132, November 7, 1987, pp. 298-300.

Various authors, Entire issue on creativity, *Scientific American*, Vol. 199, September 1958.

♦ ♦ ♦

Two great tools.

SAYWHAT?!
The lightning-fast screen generator.



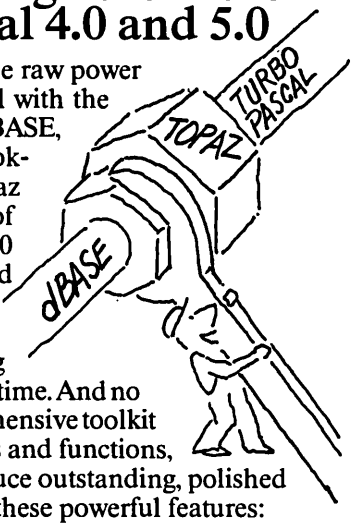
Whether you're a novice programmer longing for simplicity, or a seasoned pro searching for higher productivity, you owe it to yourself to check out Saywhat. You see, with Saywhat, you can build beautiful, elaborate, color-coded screens in minutes! That's right. Truly *fantastic* screens for menus, data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

Here's what you get:

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

$49.95

TOPAZ
The breakthrough toolkit for Turbo Pascal 4.0 and 5.0



If you'd like to combine the raw power and speed of Turbo Pascal with the simplicity and elegance of dBASE, Topaz is just what you're looking for. That's because Topaz (our brand new collection of units for Turbo Pascal 4.0 and 5.0) was specially created to let you enjoy the best of *both* worlds. The result? You can create truly dazzling applications in a very short time. And no wonder. Topaz is a comprehensive toolkit of dBASE-like commands and functions, designed to help you produce outstanding, polished programs, fast. Check out these powerful features:

- Over 200 routines all with easy-to-use, dBASE-like syntax.
- Data entry routines like **SAY, GET, PICTURE, RANGE,** color selection, unlimited data validation.
- Open up to 10 DBF files, with up to 7 indexes each with database routines like **USE, SELECT, SKIP, APPEND, PACK, INDEX ON, SET INDEX TO,** and **FIND.**
- No need to buy dBASE. **CREATE, BROWSE** and **REPORT** utilities included.
- Easily implement Saywhat and Lotus-style moving bar menus.
- **BROWSE** any DBF file with just one line of code! Programmable and windowed too.
- Comprehensive Time & Date math in 7 international formats.
- Powerful *code* and *REPORT generators* included!
- Comes with a complete 250 page manual, plus sample programs to get you started.

$49.95

Special introductory price! (Until March 31, -then $74.95)

ORDER NOW.
YOU RISK NOTHING.

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

Special limited-time offer! Save $15. Buy Saywhat?! and Topaz together before March 31, 1989 for just $85 (plus $5 shipping & handling).

Visit your nearest dealer or call toll-free:

800-468-9273

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

The Research Group
100 Valley Drive, Brisbane, CA 94005

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

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

Guaranteed!

THE RESEARCH GROUP

Reader Service Number 129

Turn Your XT Into A Controller

Part 2—Creating An Event Timer With C++

I knew Bruce was working on his C++ book with a vengeance. Well, he is and we're seeing the first fruits (and they're appealing). Here, Bruce presents example C++ code for turning an XT into an event driven controller.

This project, a time-based control system, is a complete software shell for turning your XT into an embedded controller. It's adapted from my book *Using C++* (Osborne/McGraw-Hill, June 1989).

The program (CONTROLR) parses an ASCII "script file" to create a list of timed events. CONTROLR continually checks the event list. When an event is ready, CONTROLR runs it, and then removes it from the list.

I wrote the program in C++, but if you know C you should be able to modify it for your own purposes. I give several examples to make modifying easy. If you're an experienced C or C++ programmer, I think you'll find many interesting techniques (in both object-oriented and vanilla C).

The CONTROLR package on the Micro C BBS (and the Issue #47 disk) includes source, a little bit of documentation, and a compiled CONTROLR.EXE, so even if you don't have C++ you can try it out.

If you want to dig in and modify the code, you might start by sprucing it up to include hardware events (as well as clock events). I'll suggest how later. You might also add interrupt support.

This project is more complex than the ones I usually tackle for *Micro C*. In particular, it's quite complete; so with a little customization, you can turn it into a deliverable package which allows your customer to configure his own control system. A plain vanilla XT with minimum RAM and one floppy (and whatever control hardware you need) becomes a standalone, time-based controller!

Chaotic Programming

Two factors helped me pull this project off. One — my programming skills have improved (at least a little), and two (the more significant aspect) — my programming tools (in particular, C++) give me better control over the programming process.

Larry and I were chatting about chaos the other day, and the idea of "system sensitivity to initial conditions" popped up. "Programming," we concluded, can be viewed as a chaotic process. A program is certainly a sensitive system. When we extend a program, we try to do it bit by bit (i.e., we make small changes). But the effects of a single bit can be dramatic.

One of the big advantages of an object-oriented language like C++ is that it has built-in constructs for "localization." This means the effect of a change is restricted to an area, so changes won't destabilize the whole system. My experience (and that of other programmers) with object-oriented programming leads me to believe that we can make bigger changes in a program while maintaining control.

Portability

This code should run on any C++ system based on ANSI C (either a C-code generator like Glockenspiel C++ associated with an ANSI C compiler like Microsoft C 5.1, or a native-code compiler like Zortech C++). I've used only ANSI C library calls, and the screen control functions work with any ANSI terminal (available on most UNIX machines) or a PC with the ANSI.SYS device driver loaded (put the line "Device = ANSI.SYS" in your CONFIG.SYS file).

The ANSI C standard is happening! Although the ratification process drags on, no one expects any significant changes. Compiler makers claim conformance, and ANSI C books now fill the bookstores.

The greatest thing about ANSI C is that it defines a standard for portable programs, particularly for library functions. With pre-ANSI C, you never really knew which library functions would be available, so porting code was hit-or-miss. Now you can select ANSI C library functions and (in the future when everyone conforms) know that your application can easily move from one platform to another. This applies to C++ also.

I've discovered a little book called *The Waite Group's Essential Guide to ANSI C* by Naba Barkakati, which I like a lot. It's small but complete, has an alphabetical library function index (inside front cover) and a subject index (inside back cover), and it's only $6.95. Because it's so succinct, it wouldn't be a bad introduction to C if you've programmed in another language. Recommended. (Published by H. W. Sams, Indianapolis, ISBN 0-672-22673-1.)

What's The Good Word?

A little battle is waging between C++ vendors over terminology. Here's the current bottom line: all programming languages are implemented using translators, which translate source code into some other form.

C++ has two types of translators: C-code generators, which take C++ and translate it into C, and native-code compilers, which take C++ and translate it into code specific for the target machine (assembly language or an object file). Both types have their advantages, but they're definitely different technologies. I'll use the more precise terminology to avoid confusion.

For the record — if a C-code generator and its C compiler are so well integrated that you don't notice the separation during installation and use, I'll call it a native-code compiler.

By Bruce Eckel
Eisys Consulting
501 N. 36th Street, Suite 163
Seattle, Washington 98103

Using CONTROLR

Figure 1 is the "manual page" for CONTROLR. To modify the program for your own use, edit the file CONTROLR.CXX (see Figure 2). I've used two preprocessor macros: EVENT_TYPE and MAKE_EVENT. Before you add an event to the system, look over the example definitions at the beginning of CONTROLR.CXX (light_on, light_off, bell, etc.).

Each definition consists of a call to the EVENT_TYPE macro followed by member definitions for name::action() and name::description(). Name is the new event. The controller runs action() when the event is "ready" (here, when the clock time exceeds or equals the event time). Description() displays the event.

Remove all the event definitions except for system_restart. Then create your own definitions. If you want to know how to write C code for the action() definition to control hardware, look over my articles in back issues of *Micro C* or in my book *Computer Interfacing with Pascal & C* available from Micro C.

For each EVENT_TYPE macro call and its associated definitions, you also need to add a MAKE_EVENT call further down in the program. If you search for the label CREATE: you'll see a while(1) loop containing several MAKE_EVENT definitions. These correspond to the EVENT_TYPE definitions at the beginning of the file. The arguments to the macro are the class name and a string (which contains the command name used in the script file).

Remove these macro calls and replace them with your own. Recompile the program with C++ by typing "make," and you've got your controller. That's all there is to it.

The rest of this article describes the internals of the program.

Quick C++ Review

From a C perspective, a class is a

Figure 1 — Manual page for the CONTROLR program

Usage: controlr <scriptfile> [r | n]. The second argument is optional. 'r' reprints the scriptfile in a form readable by controlr and quits. 'n' turns off the event display to speed things up. Warning: reboot is required to get out of program when the 'n' flag is used!

<scriptfile> is an ASCII file containing commands and comments. Each line in the file can be empty, or contain a single command (including a comment) or contain a comment alone. A comment is started with a single quote (') and continues to the end of the line. There are two types of commands: system commands (which control the execution of the program) and controller commands (which control the target devices). Available system commands are:

```
cycle(CC:CC:CC)
force(FF:FF:FF)
align(AA:AA:AA)
```

CC:CC:CC is the cycle time. The cycle command restarts the system after each cycle period.

FF:FF:FF is the force time. The force command restarts the system at the force time.

AA:AA:AA is the align time. If the align hours are nonzero, the system restarts on the hour (even if the align hours are greater than zero). If the align hours are zero but the align minutes are nonzero, the system restarts every even multiple of the align minutes. If align hours and minutes are zero, the system restarts every even multiple of the align seconds. If there are no system commands, the events in a script will only run once.

It doesn't make any sense to have more than one "cycle" command (only the shortest cycle will ever get used) but you can use as many "force" commands as you want. If any of the force commands are in the future, they're added to the list.

Controller commands consist of the command word (specified by the programmer), optional modifiers and optional comment. The command word and modifiers may appear in any order on the line. If a command word appears alone, as in LIGHT_ON, that command is executed immediately upon startup. Thus it is an initializer.

A controller command may have three types of modifiers. All consist of a single character and a time argument. If the character is a:

'+': the event occurs at the startup time plus the modifier time.

'@': the event occurs at the modifier time.

'R': the event time (relative or absolute) is randomized by adding a time between zero and the modifier time.

An example script:

```
'This is a comment @align(00:00:15) ' repeat every 15 sec, on the 15-sec mark

LIGHT_ON 'start with light on
LIGHT_OFF +00:00:02 'light off 2 seconds after start
LIGHT_ON +00:00:02 R00:00:08 'on between 2 and 10 seconds
    +00:00:12 LIGHT_OFF ' order doesn't matter
LIGHT_OFF @12:00:00 'light off at noon
LIGHT_ON @12:45:00 R00:30:00 'light on at 12:45 + up to 30 minutes
```

♦ ♦ ♦

Figure 2 — CONTROLR.CXX: The Main Controller

```
// Parses a file of commands and creates a list of events.
// When it is time to run an event, the event's action is
// executed and the event is removed from the list.

#include <stream.hpp>
#include <string.h>
#include <setjmp.h> // setjmp() & longjmp()
#include "evlist.hxx"

jmp_buf system_restart_buf; // for setjmp & longjmp

event_el event_list; // make only one named event_el list!

// To add a new type of event, mimic the following defs
// (call macro EVENT_TYPE and create an action for your
// new event) and add a new call of MAKE_EVENT in the
// "while(1)" loop with the comment "CREATE." That's all
// there is to extending the system. (The description()
// definition is optional).

EVENT_TYPE(light_on);
void light_on::action() {
  // put hardware control code here to physically
  // turn on the light.
}
void light_on::description() {
  puts("light is on");
}
// End of a user-defined event definition.
// More user-defined event definitions:
EVENT_TYPE(light_off);
void light_off::action() {
  // put hardware control code here to physically
  // turn off the light.
}
void light_off::description() {
  puts("light is off");
}

// an example of an action() which inserts a new
// one of itself into the event list:
EVENT_TYPE(bell);
void bell::action() {
  // ring bell every 10 seconds:
  cout << chr(7); cout.flush();
  time_point now;
  new event_el(new bell(now + time_point(0,0,10)));
}
void bell::description() {
  puts("ring bell");
}

EVENT_TYPE(greenhouse_water_on);
void greenhouse_water_on::action() {
  // put hardware control code here
}
void greenhouse_water_on::description() {
  puts("greenhouse water is on");
}


EVENT_TYPE(greenhouse_water_off);
void greenhouse_water_off::action() {
  // put hardware control code here
}
void greenhouse_water_off::description() {
  puts("greenhouse water is off");
}

EVENT_TYPE(thermostat_night);
void thermostat_night::action() {
  // put hardware control code here
}
void thermostat_night::description() {
  puts("thermostat on night setting");
}

EVENT_TYPE(thermostat_day);
void thermostat_day::action() {
  // put hardware control code here
}
void thermostat_day::description() {
  puts("thermostat on day setting");
}
// The above EVENT_TYPEs are just examples, but the
// following is used by the system:
EVENT_TYPE(system_restart);
void system_restart::action() {
  event_list.reset();
  // remove all entries from the list:
  event_el * ep;
  while ( (ep = event_list.next()) != 0)
    delete ep;
  // "nonlocal goto" back to beginning of main():
  longjmp(system_restart_buf,1);
}
void system_restart::description() {
  puts("system restart");
}

// This specifies when an event is to happen.
// relative: from system startup time
// absolute: 24-hour clock time
enum whenis { now, relative, absolute, unassigned };

// A "token" structure to hold the information in
// the line from the event description file:
struct tk {
  whenis when;
  int randomize;
  char * descriptor;
  time_point etime;
  time_point randomization;
  tk() : when(unassigned),
         randomize(0), descriptor("") ,
         etime(0), randomization(0)
         {}
  ~tk() { if ( when != unassigned )
            delete descriptor;
  }
  void display();
};


// output the event description in such a way
// that it can be re-parsed by this program:
void tk::display() {
  if ( when != unassigned ) {
    cout << descriptor;
    int dl = 25 - strlen(descriptor);
    for (int i = 0; i++ < dl; cout.put(' ') )
      ;
    if(when == relative) cout << "+";
    if(when == absolute) cout << "@";
    if ( when != now) {
      etime.display(); cout << "\t";
    }
    if(randomize) {
      cout << "R";
      randomization.display();
    }
    cout << "\n";
  }
}

main (int argc, char * argv[]) {
  if ( argc != 2 && argc != 3 ) {
    cerr << "Usage: controlr <scriptfile> [r|n]\n"
      "Second argument is optional. 'r' re-prints the\n"
      "scriptfile in a form readable by this program.\n"
      "'n' turns off event display to speed things up.\n"
      "Warning: reboot required to get out of program\n"
      "when the 'n' flag is used!\n"
      " <scriptfile> is an ASCII file containing\n"
      " controller commands\n";
    exit(1);
  }
  int evdisplay = 1; // flag means "display events"
  int reprint = 0; //flag means re-print scriptfile & quit
  if ( argc == 3 ) {
    if ( *argv[2] == 'n' )
      evdisplay = 0;
    if ( *argv[2] == 'r' )
      reprint++;
  }
```

```
// set the restart buffer so longjmp comes back here:
  setjmp(system_restart_buf);
  time_point startup_time;  // time at startup or restart
  { // forces istream eventscript out of scope at closing
    // brace, which closes file so it is re-opened when
    // this scope is entered again.
  filebuf fl;
  if (fl.open(argv[1], input) == 0) {
    cerr << "cannot open %s\n" << argv[1] << "\n";
    exit(1);
  }
  istream eventscript(&fl);
  // for use by strtok(), the token grabber.  Tokens are
  // whitespace or paren delimited:
  const char * delimit = " \t\n\r()";
  const int BSIZE = 100;
  char buf[BSIZE], c;
  while( eventscript.get(buf,BSIZE) ) {
    eventscript.get(c); // throw away newline delimiter
    { // to force tk token out of scope after each loop
    tk token; // to save information about event
    char * tokptr = strtok(buf,delimit); //get 1st token
    while (tokptr != NULL) { // do for all tokens in line
      if ( * tokptr == '\'' ) { // start of comment
        break; // throw away to end of line
      }
      if ( strcmp(tokptr,"cycle") == 0 ) {
        // make cycle time from next token:
        time_point cycle_time(tokptr =
                              strtok(NULL, delimit));
        if (reprint) {
          cout << "cycle(";
          cycle_time.display();
          cout << ")\n";
        }
        cycle_time = cycle_time + startup_time;
        // enter it into the event list:
        new event_el(new system_restart(cycle_time));
      } else
      if ( strcmp(tokptr,"force") == 0 ) {
        // make force time from next token:
        time_point
          force_time(tokptr=strtok(NULL,delimit));
        if (reprint) {
          cout << "force(";
          force_time.display();
          cout << ")\n";
        }
        // if we aren't already past the force_time,
        // enter it into the event list:
        if ( ! (startup_time >= force_time))
          new event_el(new system_restart(force_time));
      } else
      if ( strcmp(tokptr,"align") == 0 ) {
        // make align time from next token:
        time_point
          align_time(tokptr=strtok(NULL,delimit));
        if (reprint) {
          cout << "align(";
          align_time.display();
          cout << ")\n";
        }
        if (align_time.hr()!=0) { //XX:00:00 align to hrs
          align_time.hr() = startup_time.hr() + 1;
          align_time.min() = align_time.sec() = 0;
        } else {
          align_time.hr() = startup_time.hr();
          if (align_time.min()!=0) { //00:XX:00 align min
            int next_min =
              ((startup_time.min()/align_time.min()) + 1)
                * align_time.min();
            if (next_min > 60 )
              align_time.hr()++;
            else
              align_time.min() = next_min;
            align_time.sec() = 0;
          } else { // 00:00:XX align seconds
            align_time.min() = startup_time.min();
            if (align_time.sec() != 0) {
              int next_sec =
                ((startup_time.sec()/align_time.sec())+1)
                  * align_time.sec();
              if (next_sec > 60 )
                align_time.min()++;
```

Continued on page 30

DBASE™ *On Line*

"Pop-Up" DBASE Reference System Powered by The Norton Guides™

Only $99

DBASE *On Line* includes
The Norton Guides reference
Engine compiler and linker

Plus reference databases for
- Clipper
- Quicksilver
- dBASE III Plus
- dBXL
- dBASE IV
- FoxBASE+

Instant "pop-up" Reference

DBASE *On Line* is a "pop-up" quick reference system that provides you with instant access to all aspects of the DBASE language and programming environment.

No More Searching Through Books

Each DBASE *On Line* reference database is a complete and thorough reference library eliminating the need for tedious time-consuming searches through books and manuals. All reference databases are organized so you can find relevant information *fast*.

Comprehensive DBASE Reference

DBASE *On Line* gives you quick reference to important information such as; syntax, description, options, notes, library, example of use and complete cross reference to all related keywords.

Replaces Books & Manuals

Reference topics include; Commands, Functions, Operators, Cursor Navigation Keys, Error codes and messages, Config settings, Technical specifications, dbf file structure, descriptions for all utility programs along with tables for reserved words, Inkey(), Readkey(), ASCII codes and line drawing characters. The Clipper and Quicksilver databases also provide complete reference on such topics as; Compiling, Linking, Debugging and full reference to the Clipper Extend System including C interface functions, Assembler macros and much much more.

Powered by The Norton Guides reference engine

To power DBASE *On Line*, We have included the Norton Guides "reference engine". It features ease of use, small size and fast automatic lookup capability. It instantly "pops-up" information on your screen, right next to your work, right where you need it. In either full screen or movable half screen.

DBASE *On Line* is powered by
the Norton Guides reference
engine
- Memory requirements 74K
- Can run in resident or pass through mode
- Can be popped up anytime inside any program
- Automatically looks up keywords selected on the screen
- Full or half screen display
- All available Norton language guides will run with the Norton Guides reference engine
- Also includes the Norton Guides reference database compiler and linker allowing you to create your own reference guides

DBASE *On Line* ... Is Easy To Use

You will be using DBASE *On Line* productively in less than 5 minutes. Guaranteed!

Create Your Own Reference Databases

In addition to our databases, you can also create your own reference databases with The Norton Guides reference database Compiler and Linker which is included in every DBASE *On Line* package. Our well written manual will take you through each step of the creation process.

60 Day Guarantee — Free Upgrades

DBASE *On Line*

The Reference System of Choice for DBASE Users.

To Order, Call Toll Free: 1-800-622-6435

SofSolutions 440 Quentin Drive San Antonio, TX 78201

Trademarks: Norton Guides/Peter Norton Computing, dBASE III Plus IV/Ashton-Tate, Clipper/Nantucket, dBXL Quicksilver/Wordtech Systems, FoxBASE+/Fox Software, On Line/SofSolutions

Reader Service Number 108

```
            else
              align_time.sec() = next_sec;
          }
        }
      }
      align_time.adjust();
      new event_el(new system_restart(align_time));
    } else
    if ( *tokptr == '+' ) {
      token.when = relative;
      token.etime = time_point(++tokptr);
    } else
    if ( *tokptr == '@' ) {
      token.when = absolute;
      token.etime = time_point(++tokptr);
    } else
    if ( *tokptr == 'R' || *tokptr == 'r' ) {
      token.randomize++;
      token.randomization = time_point(++tokptr);
    } else {
      // it wasn't anything else -- assume a name
      token.descriptor = new char[strlen(tokptr) + 1];
      strcpy(token.descriptor,tokptr);
    }
    tokptr = strtok(NULL, delimit); // get next token
  }

// line parsed.  Now perform actions based on token.
  if(!*token.descriptor)
    // Line with empty descriptor means comment line
    // or system cmd (which was executed in parser).
    // Go back to head of while loop, get next line:
    continue;
  // If the token isn't empty but it isn't relative or
  // absolute, it means the event should happen now:
  if(token.when != relative && token.when != absolute)
    token.when = now;
  if(reprint)
    token.display();

  // Add event to list. First, a macro which creates
  // the right event based on token information. This
  // saves a bunch of typing and mistakes. The final
  // break gets out of the while(1) loop which follows.
  // The macro goes here instead of at the beginning of
  // the file (as is conventional) because it makes the
  // code easier to understand.
  //    STRING is the string the parser finds in the
  // script file. ENAME is the name of the class.
#define MAKE_EVENT(STRING,ENAME) \
if (strcmp(token.descriptor,STRING) == 0) { \
  switch(token.when) { \
    case relative:  \
      if (token.randomize)  \
        new event_el
    (new ENAME( startup_time, \
                              token.etime, \
                              token.randomization)); \
      else \
        new event_el \
              (new ENAME(startup_time,token.etime)); \
      break; \
    case absolute: \
      if (token.randomize) \
        new event_el(new ENAME(token.etime, \
                              time_point(0), \
                              token.randomization)); \
      else \
        new event_el(new ENAME(token.etime)); \
      break; \
    case now: \
      new event_el(new ENAME); \
  } \
  break;  \
}
// End of macro definition


// Create the event and add it to the list, using the
    // information parsed from the script file.
    // CREATE:
    while(1) {
      // When a match is found, a "break" statement in the
      // macro jumps out of the "while(1)" loop.
      MAKE_EVENT("LIGHT_ON", light_on);
      MAKE_EVENT("LIGHT_OFF", light_off);
      MAKE_EVENT("BELL", bell);
      MAKE_EVENT("GREENHOUSE_WATER_ON",
                 greenhouse_water_on);
      MAKE_EVENT("GREENHOUSE_WATER_OFF",
                 greenhouse_water_off);
      MAKE_EVENT("THERMOSTAT_DAY", thermostat_day);
      MAKE_EVENT("THERMOSTAT_NIGHT", thermostat_night);
      // if there wasn't a match, it's an error:
      cerr << "unrecognized line in " << argv[1] << "\n";
      token.display();
      exit(1);
    } // end of "while(1)"
    } // end of tk token scope (token destroyed)
  } // all lines in file are parsed
  } // end of scope enclosing istream eventscript
  if (reprint) exit(0);  // just reprint script
  cout << chr(27) << "[2J"; // ANSI clear screen
  // The list of events has been completely built.
  // Now loop through the list and look for events to run.
  while(1) {
    { // this { forces "current_time" out of scope so it
      // is destroyed and created each pass through loop
      time_point current_time;
      if (evdisplay) {
        cout << chr(27) << "[s"; // ANSI save cursor loc
        cout << chr(27) << "[1;40H"; // cursor to 1,40
        cout << chr(27) << "[7m"; // reverse video
        current_time.display();
        cout << " started: "; startup_time.display();
        cout << chr(27) << "[m"; // no attributes
        cout << chr(27) << "[u"; // restore cursor
      }
      event_list.reset();
      event_el * ep;
      while ( (ep = event_list.next()) != 0) {
        if ((ep->data())->ready(current_time)) {
          if(evdisplay) {
            cout << "\nRUNNING an event:\n";
            (ep->data())->description();
          }
          (ep->data())->action(); // run the event
          delete ep; // remove event from list
          // print remaining list times and descriptions:
          if(evdisplay) {
            cout << "\n\n Event list:\n";
            event_list.reset();
            while ( (ep = event_list.next()) != 0) {
              (ep->data())->display();
              cout << " ::: ";
              (ep->data())->description();
            }
          }
        }
      }
    } // close of "current_time" scope
  } // end of while(1) looping through event list
}
```

♦ ♦ ♦

struct with a few extras. A function inside a class definition gets treated just like any other structure member; you dereference a member with a dot (for a structure variable) or an arrow (for a structure pointer).

Members can be public (available to everyone) or private (only member functions can change or use them). A structure in C++ is a class with all members public (so a structure can have member functions).

All classes and structures have special functions called constructors (a member function with the class name), called when a variable (object) is created, and a destructor (a tilde followed by the class name), called by the compiler when an object goes out of scope.

Inheritance & Member Objects

The controller uses two object-oriented concepts: "inheritance" and "member objects." Let me explain this further.

There are two ways you can use an old class to make a new class in C++.

Figure 3 — EVENT.HXX

```
// Each event object has a scheduled time, and a virtual
// function to be executed at that time. The event class
// should be derived into a class with the desired "action"
// function; a list of events is managed in main().

#include <stdio.h>

// A class to manage time:
class time_point {
  int hours;
  int minutes;
  int seconds;
public:
  time_point(); // get current time
  void adjust ();
  // set a specific time:
  time_point ( int hr, int min = 0, int sec = 0 ) {
    hours = hr; minutes = min; seconds = sec;
    adjust();
  }
  // the copy-initializer: (create one point from another)
  time_point (time_point & rv);
  time_point (char *);  // from string, i.e.: "09:45:23"
  void randomize(time_point & random_f);
  // compare one time point to another:
  time_point operator=(time_point & rv); // assignment
  int operator>=(time_point & rv);
  time_point operator+(time_point & rv);
  void display(){
    printf("%2.2d:%2.2d:%2.2d",hours,minutes,seconds);
  }
  int & hr() { return hours; }
  int & min() { return minutes; }
  int & sec() { return seconds; }
};

// class to manage events which occur at particular times:
class event {
  time_point event_time; // when the event should happen
public:
  // note the initialization of the member object:
  event() : event_time() {}; // no arguments -- do it now
  event(time_point & tp)  : event_time(tp) {} ; // absolute
  event(time_point & tp, time_point & rst_time) //from rstrt
    : event_time(tp + rst_time) {}
  event(time_point & tp, time_point & rst_time,
        time_point & random_f)
    : event_time(tp + rst_time) {
      event_time.randomize(random_f);
    }
  // "ready" is true if event is ready to run:
  int ready(time_point & now) {
    return now >= event_time;
  }
  void display() { event_time.display(); }
  // The following function is re-defined for each specific
  // subclass:
  virtual void action() { // what happens at event_time
    fprintf(stderr, "error -- base class used; %s\n"
      "no action specified for this event");
  }
  // This is optionally re-defined so you can see what
  // events are waiting to be run:
  virtual void description() { puts("no description"); }
};

// This macro derives a new class from class event. It saves
// typing and errors when you make a new type of event.
// To use it: EVENT_TYPE(event_class_name);  you must also
// define a function for the new action which is to happen
// when the event is ready to run:
// void event_class_name::action() { /* defintion here */ }
// See the examples in CONTROLR.CXX

#define EVENT_TYPE(ENAME) class ENAME : public event { \
  public: \
  ENAME() : () {} \
  ENAME(time_point & p) : (p) {} \
  ENAME(time_point & p, time_point & s) : (p,s) {} \
  ENAME(time_point&p,time_point&s,time_point&r):(p,s,r){} \
  void action(); \
  void description(); \
};
```

♦ ♦ ♦

One (probably the one you've heard about) is "inheritance." When you inherit a new class from an old one, you make a new version of the old class. You can also use a member object in a class. A member object is simply another class element.

When you use an old class to make a new one, you must initialize the old class by calling the constructor. You call the constructor for the base class (if you're inheriting) or the member object before the code for the new class constructor executes. To show this, C++ syntax has the constructor calls after the closing parenthesis of the argument list, but before the opening brace of the constructor function body. For example —

```
class old {
 int i;
public:
 old(int j = 0) { i = j; }
};

class mem_ob {
 int i;
public:
 mem_obj(int j = 0) { i = j; }
};

class derived : public old {
 mem_obj member_object;
public:
 derived(int p = 0, int q = 0) :
  (p) , member_object(q) {}
};
```

Class derived is inherited from class old and contains an object of the class mem_obj. There's no object name associated with the base class, so no name is used to call the base class constructor.

Object-Oriented Event Control

CONTROLR is "truly object-oriented." It manipulates generic objects. Each object belongs to the general base class, "event" (see the file EVENT.HXX, Figure 3), and to a specific derived class that has special properties in two virtual functions, description() and action(). A virtual function is declared in the base class and defined in the derived class, so two objects can have the same interface but different implementations.

We keep objects of the base class in a list and ready() constantly checks them to see if they're ready to run. Here, ready() only ties to the clock. But you can tie it to hardware.

For hardware events, make ready() a virtual function which defaults to checking the clock. When deriving a new event, change the definition of ready() so that it checks the state of some hardware

instead of the clock — very simple.

When an event is ready, its action() is performed and the event gets removed from the list.

A Class To Manage Time

Class event contains a member object called event_time which belongs to class time_point, defined just prior to class event. A time_point object holds a single point in time.

Points can be assigned, added, compared, and randomized (i.e., increased by a random amount of time or bounded by a random factor). Once you've defined class time_point, you can ignore the details of time calculations.

Each object in class event relates to a single time_point. When the clock time equals or exceeds the event_time, the member function, ready(), returns true. So you can ask an event if *it's* ready to run.

To create a new type of event, inherit a new class (from event). Then if you want, redefine the virtual functions, description() and action(). If you don't they default to their definitions in the base class.

The process of inheriting a new class changes only in name from event to event. So the code to inherit a new subtype is packaged in the macro EVENT_TYPE, defined in EVENT.HXX, and used in CONTROLR.CXX. Notice that you can continue a macro as long as you keep putting backslashes at the end of each line.

Design Guidelines

When designing an object-oriented system, it's important that your concept of the object encompass all the object's possible uses. In this system, for example, an action() can be anything, including restarting the system (see EVENT_TYPE(system_restart) in CONTROLR.CXX). Notice also the flexibility of the system; the action of one event can add other events to the list, as shown in EVENT_TYPE(bell).

The system restart commands are handy because —

(1) you may want to create a process which repeats itself;

(2) and there may be a power failure where you'll want the controller to resynchronize itself.

A C++ Oriented Linked List

Completely object-oriented programs manage an arbitrary number of generic objects (from classes which use polymorphism, aka virtual functions). A linked list seems to be the best way to handle these objects, so I've been trying to create a better linked list for C++.

My best try so far, the list in EVLIST.HXX (see Figure 4), uses the unique features of C++ to advantage, in particular constructors, destructors and static class variables.

Many linked lists define a link element, and then a "container" to manage the links. The class event_el in EVLIST.HXX contains itself by using a static class variable for the head pointer (the pointer to the beginning of the list). A static class variable allows all objects in a class to share common information economically. Space for only one variable of that name is defined, and all objects share the same data space for the variable.

As in C, you use a global static class variable with a hidden name known only to members of the class (assuming a private element).

Unlike C, you shouldn't initialize static class variables when they're declared. Initialization tells the C++ transla-

Figure 4 — EVLIST.HXX: A Self-contained Linked List

```
// The first object must be created as a named variable with
// no arguments. This is the name of the list. All the rest
// are created with "new" and arguments, but the return val
// of "new" is never used (normally a no-no). Because "head"
// is static, only one list name can be used in a program.
#include "event.hxx"
#include <stdio.h>

class event_el {
  // the start of all members of this class:
  static event_el * head;
  // used to step through the list:
  static event_el * cursor;
  event_el * next_el; // link to the next element
  event * ddata; // holds the information in this node
  void error(char * msg = "") {
    printf("list error: %s\n", msg);
    exit(1);
  }
public:
  event_el() { // create a named list (don't use "new"!)
    next_el = 0;
    ddata = 0; // to mark end of list
    // point to yourself as the only element:
    cursor = head = this;
  }
  event_el(event * info) { // only use "new" on this!
    ddata = info;
    // insert this element at the head of the list:
    next_el = head;
    head = this;
  }
  ~event_el() {
    event_el * current = head, * old = head;
    while ( current != this ) {
      // find our place in the list, and the element before
      if(current==0) error("can't delete nonexistent link");
      old = current;
      current = current->next_el;
    }
    if ( this == head )
      head = next_el;  // move head to next link
    old->next_el = next_el; // unlink this from list
    delete ddata; // does nothing if ddata == 0
  }
  const event * data() { return ddata; }
  void reset() { cursor = head; }
  event_el * next() { // step through the list
    // return the current element and step forward one,
    // if we can.
    event_el * llp = cursor;
    if (cursor->next_el != 0)
      cursor = cursor->next_el;
    if (llp->ddata != 0)
      return llp;
    else // tail element has empty ddata ptr (don't use it)
      return 0;
  }
};
```

♦ ♦ ♦

CP/M, NorthStar, Macintosh, Apple II, MS-DOS, and PS/2

Don't let incompatible diskette formats get you down — read them all with your PC!

Teach your PC to speak CP/M

UniDOS Z80 Coprocessor Board by MicroSolutions

Run your Z80 and 8080 code programs at LIGHTNING speed on your PC or AT with the UniDOS 8MHz. Z80 coprocessor board. UniDOS automatically switches from MS-DOS to CP/M mode when your CP/M program is executed. UniDOS emulates most common computers and terminals such as Kaypro, Xerox 820, Morrow, Osborne, and VT100. All standard CP/M system calls are supported. Includes UniDOS and UniForm-PC.
UniDOS Z80 Coprocessor Card ... $ 169.95

UniDOS by Micro Solutions

Equip your PC/XT with an NEC V20 chip and run your favorite CP/M programs without taking up another card slot. Runs 8080 code directly on the V20, and uses emulation mode for Z80 code or systems without a V20.
UniDOS by MicroSolutions ..... $ 64.95
UniDOS w/UniForm & V20-8 chip ..... $ 135.00

UniForm-PC by MicroSolutions

How have you ever wished you could use your CP/M diskettes on your PC? Now you can access your CP/M files and programs on your MS-DOS computer just as you would a standard MS-DOS diskette. Install UniForm and use standard DOS commands and programs right on your original diskette without modifying or copying your files. UniForm-PC allows you to read, write, format, and copy diskettes from over 275 CP/M and MS-DOS computers on your PC, XT, or AT. With UniForm-PC and the Compaticard, you can use 5¼" high density, 96TPI, 3½" (720k/1.44M), and even 8" drives.
UniForm-PC by MicroSolutions ... $ 64.95
Also available for Kaypro, & other CP/M computers

CompatiCard by MicroSolutions

THE universal four drive floppy controller board for the PC or AT. Run up to 16 disk drives (4 per CompatiCard), including standard 360K, 96 TPI, high density 1.2M, 8" (SSSD or DSDD), and 720k/1.44M 3½" drives. Comes with its own MS-DOS driver and format program. Use it with UniForm-PC for maximum versitility.
CompatiCard Board ..... $ 119.95
CompatiCard with UniForm-PC ..... $ 179.95
8" Drive adaptor ..... $ 15.00
External 5¼" drive cable ..... $ 15.00

Compaticard II by MicroSolutions

Two drive version of the CompatiCard, sorry no 8" or single density.
CompatiCard II ..... $ 89.95
CompatiCard II with 1.2M or 3½" internal drive ..... $ 199.95

Megamate by MicroSolutions

This is the 3½" drive package that you've been waiting for. Run 720k or 1.44M diskettes in this attractive external drive. Comes complete with its own controller card. Easy to install, just plug it into your PC or AT and go.
Megamate ..... $ 329.95

MatchPoint-PC by MicroSolutions

The MatchPoint-PC board for the PC/XT/AT works with your standard controller card to let you read and write to NorthStar hard sector and Apple II diskettes on your PC. INCLUDES a copy of the UniForm-PC program, as well as utilities to format disks, copy, delete, and view files on Apple DOS, PRODOS, and Apple CP/M diskettes.
MatchPoint-PC Board $179.95

MatchMaker by MicroSolutions

Now you can copy your Macintosh diskettes right on your PC/XT/AT with the MatchMaker. Just plug your external 3½" Macintosh drive into the MatchMaker board and experience EASY access to your Mac diskettes. Includes programs to read, write, initialize, and delete files on your single or double sided Mac diskettes.
MatchMaker Board ..... $ 139.95
MatchMaker w/External Mac Drive ..... $ 325.00

Hard Disks for CP/M systems

Pep up your CP/M computer with hard disk performance. Our simple to install kits allow you to connect up to two 5¼" hard drives to your Z80 system. The Winchester Connection software customizes your system from an easy to use menu, with flexible drive parameters, partition and block size, and includes complete installation and diagnostic utilities. A complete system requires a HDS daughter board, WD1002-05 hard drive controller board, hard drive, software package and cables.
HDS Host Board with Software ... $ 79.95
HDS Board, WD1002-05, and software ..... $ 245.00
WD1002-05 Controller Board only . $ 185.00
External drive cabinet with power supply ..... $ 139.95

Parts and accessories for the Kaypro and Xerox 820-1

Plus2 ROM Set for Xerox 820-1 ... $ 39.95
Plus2 ROM with X120 bare board . $ 49.95
KayPLUS ROM for Kaypro 2, 4, 10 - specify ..... $ 69.95
Kaypro 2X Real-time Clock parts kit ..... $ 29.00
Kaypro 2X Hard disk interface parts kit ..... $ 16.00
Kaypro 10 Hard Disk controller board ..... $ 185.00
Kaypro four drive floppy decoder board ..... $ 35.00
QP/M Operating System - bootable - specify system ..... $ 64.95
QP/M without CBIOS (installs on any Z80 system) ..... $ 49.95
Complete parts and repair services available

Special Purchases!!

PC-Mastercard by Magnum Computer

This is probably the BEST multi-function card on the market. Use mixed banks of 64k and 256k chips to install up to 1.5 Megabytes of RAMDISK, and PRINT SPOOLER (or fill your system up to 640k). Serial, parallel, game ports, and real time clock installed! Comes with complete software.
PC-MASTERCARD (0k installed) ..... $ 69.95

Turbo Editor Toolbox by Borland International ... $ 29.95

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 or compatible and Turbo Pascal 3.0.

COPY II PC by Central Point Software ..... $ 24.95

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.

Printer/Data Switches

Quality with economy. These boxes switch all 25 lines so they can be used with either RS232, or IBM parallel (DB25) printer cables.
Four port data switch ..... $ 39.95
Two port data switch ..... $ 34.95
IBM style Parallel Printer Cable .... $ 12.00
Three cable set ★★ Special ★★ ... $ 30.00

MicroPro Manuals

WordStar V3.3 Manual ..... $ 12.00
InfoStar Set (DataStar & ReportStar) ..... $ 18.00

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

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

EMERALD MICROWARE

(503) 641-8088

VISA MasterCard

P.O. Box 1726 • Beaverton, OR 97075

tor to make space for the variable, so if you include the header containing the class definition in more than one place, they'll conflict when you try to link the program.

In C++ lingo, let a special constructor initialize static class variables.

This special constructor in class event_el is the constructor without arguments. Because of the static variable, you can only create one linked list for each unique class name. So you can only call this constructor once — at the beginning of CONTROLR.CXX. If you want to make another list, copy the class definition to a new file and give the class a new name.

The event_el class contains itself via the static class variable —

```
static event * head;
```

A new list called event_list is created in CONTROLR.CXX with the line —

```
event_el event_list;
```

This points head to an empty element (indicating the end of the list). When you call the constructor with an argument, event_el::event_el(event * info), call it with the new keyword. This creates an object on the free store (heap).

New normally returns the address of the object it created, and in most cases if you lose this address you can never release that heap space. Here, however, the constructor inserts the element into the linked list at the head, so the address isn't lost.

What would normally be a disastrous statement —

```
new event_el(new event);
```

is the proper way to add an element to the list. Also, notice the static cursor pointer. This means only one cursor serves the entire list.

Most linked lists have a function to unlink elements. After struggling for a while, I realized that C++'s destructor could neatly remove objects from the linked list (I haven't seen it used this way anywhere else). The compiler automatically calls the destructor when an object goes out of scope.

The user can explicitly call the destructor for an unscoped object (i.e., one created on the free store) using the keyword delete. If you call delete with a pointer to an event_el, the destructor removes that event_el from the linked list and deletes the data. This produces very tidy code (see the last part of CONTROLR.CXX).

Parsing The Script

In the file CONTROLR.CXX (just after the class deriving code), you'll see a definition for a struct called tk (for token). A tk holds all the information about a particular event while the parser parses a line. When the parser finishes a line, CREATE creates a new event based on the information in the tk.

This approach has several advantages. It generally separates the analysis from the action (except for system commands, which you can easily add to tk) making the code easier to understand and maintain.

A large if-else statement analyzes each token until the entire line has been parsed.

By creating an internal representation instead of executing events as soon as you figure out what they are, you allow the possibility of saving the internal representation of the entire file instead of handling it a line at a time, as I've done. You can easily modify this design for faster system restarts.

We use the C++ stream class for console and file I/O. main() opens the script file as a stream, and the member function, get(), reads one line at a time from the stream until the file ends. Since get() only reads to a terminating character (and pushes the terminator back on the input stream), an extra get() of a single character is necessary before the next line get().

get() is an overloaded function. The same function name can take several different types of arguments. This terminator defaults to "\n."

The ANSI C function, strtok(), divides a line into pieces. strtok() really shines when you want to parse input. Literally, it breaks a string into tokens, where a token equals any text. strtok() looks for a single character terminator from among the characters you give it in the third argument. The possible terminators (given by the constant string delimit) include white space (space, tab, linefeed, carriage return) or an opening or closing parenthesis.

The first time you call strtok() for a line, give it the starting address of the buffer as the first argument. For subsequent calls using the same line, give it NULL as the first argument. strtok() will return a pointer to a null-terminated string token until it can't find any more (in which case it returns NULL). A large if-else statement analyzes each token until the entire line has been parsed.

You can see that the system commands (force, cycle, align — commands which modify the control system) are handled differently in the parser than the control commands (which modify the system being controlled). The parser executes system commands ("execute" here means "an event is added to the event list"). Control commands require string matching and argument checking before an event can be added to the event list.

You can improve the design of the parser by adding cycle, force and align to the enumeration whenis. During parsing simply assign token.when to the type of system command and token.etime to the time, and delay the event creation until after completion of parsing. This improvement is also the first step necessary if you want to represent the entire file in memory.

One system command you want to add is commandfile, which changes the name of the controller script file used when the system restarts.

Adding An Event To The List

After each line is parsed, an event is added to the list based on the information in the line. The macro MAKE_EVENT compares the token.descriptor to a string; if it matches, it makes a new event.

This occurs inside what appears to be an infinite while loop. You can think of it as a case statement that matches strings (instead of simple integers, as an ordinary case does) — implemented this way to simplify the addition of new event types.

An if-else construct would not have fit neatly into a macro (since else would occur at the end of the macro). By using a break at the end of the macro to jump out of the while(1) loop, the code fits together nicely.

Managing The List

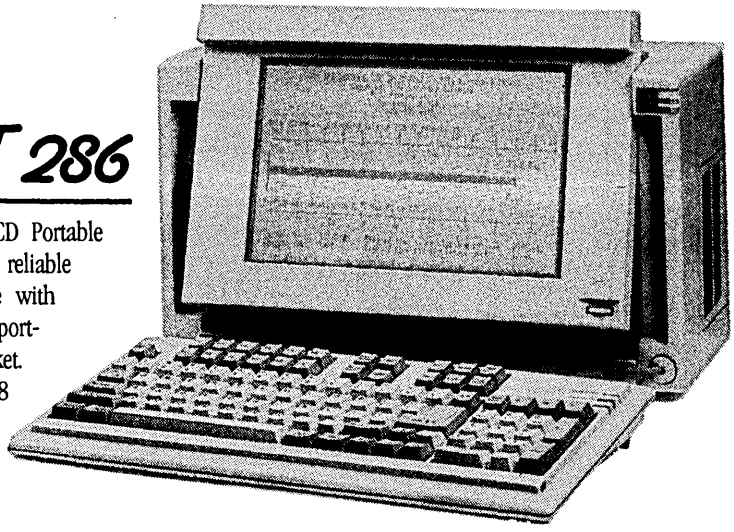
After parsing the script file and constructing the list, the program loops through the list looking for events to run. At the beginning of each loop, a new current_time object is created and all the event objects in the list are tested with

McTek

LCD Portable

(Also available in PLASMA PORTABLE)
(Including Hard Disk Only 19 lbs.)

RABBIT 286



The McTek Rabbit-286 LCD Portable combines the fastest, most reliable AT motherboard available with most visible full-size LCD portable screen on the market. Running at a switchable 8 or 10 MHz Ø wait state, it includes a 20MB hard disk, 1.44MB 3½" floppy drive, parallel & serial ports, Award 3.03 bios, 640k & turbo indicator LCD. The screen is a fantastically readable, electroluminescently backlit, 80-column by 25-line, high resolution 640x400 super twisted LCD with adjustable intensity and screen-angle. The screen size is 9.5"x6". It's as readable as a CRT. You can also plug in a digital or analog color monitor or a digital or composite monochrome monitor. Included also is an external 5¼" floppy port for reading and converting to 3½" disks (5¼" external drive w/case: $179 when purchased with LCD Portable). The McTek Rabbit-286 LCD Portable comes fully assembled with our one-year parts & labor guarantee, and sells for an amazing, complete price of only ***$1799!*** 386-20 MHz W/1MB ***$2599!***

$1299.00
12MHz/0 Wait State

Assembled & Tested IBM® AT Compatible
MS-DOS® OS/2® Compatible
- 80286 12/6 MHz
- Phoenix BIOS
- 640K of RAM Expandable to 4MB
- 0 Wait State
- 200W Power Supply
- 1.2MB Floppy Drive
- Ports: 1 Serial, 1 Parallel, 1 Game
- Dual Floppy/Dual H.D. Controller
- Monochrome Graphic Card
- 101 Key Enhanced Keyboard
- TTL Monitor 12"
- 20MB Hard Disk

Options:..... Call

$1899.00
16MHz/0 Wait State

Assembled & Tested IBM® AT Compatible
MS-DOS® OS/2® & UNIX® Compatible
- 80386 16/8 MHz Norton V4.0 SI 17.6
- Phoenix BIOS
- 1MB expandable to 16MB RAM
- 80387 Coprocessor Socket
- 200W Power Supply
- 1.2MB Floppy Drive
- Ports: 1 Serial, 1 Parallel, 1 Game
- Dual Floppy/Dual H.D. Controller
- Monochrome Graphic Card
- 101 Key Enhanced Keyboard
- TTL Monitor 12"
- 20MB Hard Disk (28MS)

Options:..... Call

$2199.00
20MHz/0 Wait State

Assembled & Tested IBM® AT Compatible
MS-DOS® OS/2® & UNIX® Compatible
- 80386 20/8 MHz Norton V4.0 SI 22
- Phoenix BIOS
- 1MB expandable to 16MB RAM
- 80387 Coprocessor Socket
- 220W Power Supply
- 1.2MB Floppy Drive
- Ports: 2 Serial, 1 Parallel, 1 Game
- Dual Floppy/Dual H.D. Controller
- Monochrome Graphic Card
- 101 Key Enhanced Keyboard
- TTL Monitor 12"
- 40MB Hard Disk (28MS)

Options:..... Call

McTek Systems, Inc. • 1521 San Pablo Avenue • Berkeley, CA 94702 • 415-525-5129

DISK DRIVES

Fujitsu 360k	$69
Fujitsu 1.2MB	$89
Teac	$75
Teac 1.2MB	$95
Toshiba 3½" 720K	$89
Teac 3½" 1.4MB	$105
20MB Hard Disk Kit	$279
30MB Hard Disk Kit	$309
ST-225	$215
8425 Miniscribe	$249
8438 30MB Miniscribe	$259
3650 40MB Miniscribe	$349
3675 60MB Miniscribe	$379
ST-157 49MB 3½"	$479

PRINTERS

Citizen CD 120	$149
Citizen CD 180	$189
HPLASAR Serial2	$1699
Epson LX-800	$219
Epson LQ-500	$379
Toshiba 321 XL	$519
NEC P2000	$369

Call for prices of other brands

MODEMS

Everex int. 300/1200	$79
Everex 2400 external	$195
Everex 2400 internal	$179

MONITORS

Samsung amber	$79
Samsung EGA color	$359
Samsung RGB color	$259
NEC Multisync	$559
Sony Multiscan	$619
HGC-compat.mono card	$49
Color graphic card	$49
EGA Paradise 480	$149
VGA Paradise	$279
Genoa Super VGA	$299

MOUSE

Logimouse C7	$69
Logimouse H1 Res	$99

PC/XT

640k TurboMothrbrd	$80
10MHz TurboMothrbrd	$85
Multi I/O w/disk contrlr	$59
640k RAM card	$39
2MB Expansion card	$89
RS232 2-port card	$35
4-serial port card	$79
Game I/O card	$15
384k Multifunction card	$69
FCC-app. slide XT case	$29
150W power supply	$49
XT keyboard	$42
Clock Card	$19
Floppy Controller	$19

PC/AT

McTek286-20MHz	$449
Baby McTek 286B-AT 8/10 O-wait	$249
McTek 386-16MHz	$799
McTek 386-20MHz	$859
McTek 386-24MHz	$999
Locking slide case	$59
200W power supply	$65
Enhanced keyboard	$59
WD FD/HDC	$129
DTC FDC/HDC 1:1	$189
3MB EMS (0K)	$99

DESKTOP

MISC.

Kingtech CRT Portable Kits: XT/AT (power supply, case keyboard, monitor)	$380/$410
Eprom burner 4-socket	$139
LCD Portable	$759
Plasma Portable Kits	$999
AC power strips	$15
Diskette file box	$9
Printer or serial cable	$8
Archive Tape Backup 40MB	$339

XT 10MHz 640k
2 Drive System $699

Reader Service Number 42

their ready() functions, using current_time as the argument.

Notice the extra set of braces in the final while(1) loop. The only purpose of these braces is to force current_time out of scope at the end of the loop. When it goes out of scope, it calls the destructor, and when the loop starts again, it calls the constructor. This updates current_time. The program uses the trick of forcing an object to go out of scope (thus calling a constructor or destructor) in several places.

If you ignore the display code in the list manager, the remaining code is surprisingly compact:

```
while(1) {
 {
 time_point current_time;
 event_list.reset();
 event_el * ep;
 while((ep=event_list.next())!=0)
  if((ep->data())->
    ready(current_time)) {
   (ep->data())->action();
   delete ep;
  }
 }
}
```

The list is reset, and each element is tested against the current time until there are no more elements in the list. If it finds one, it's run and removed from the list. This abbreviated code is executed when the "n" command-line option is used.

I used object pointers, so arrows dereference the member functions.

Restarting The System

To make a system command just another type of event, system_restart::action() must be able to jump to the beginning of the main() function. This isn't as easy as using goto, since any labels in main() are outside the scope of system_restart::action().

C solves this problem with a concept called non-local goto, implemented with the library functions setjmp() and longjmp(). The setjmp() function stores the contents of the stack and the program counter in a type of structure called a jmp_buf. When longjmp() is called with that same structure as an argument, it goes back to the point where setjmp() was called and restores the same stack.

Except for the return value of setjmp(), it looks exactly like the first time setjmp() was called. Notice that longjmp() is called inside the system_restart::action() function, which has no idea where the jump will end up. Because

Figure 5 — EVENT.CXX: definitions for EVENT.HXX

```
// (class event functions are all in-line).
#include "event.hxx"
#include <time.h>
#include <stdlib.h> // for rand(), srand(), atoi()

#define db(var) printf(#var " = %d\n", var)

// Correct the time so hours < 24, minutes & seconds < 60
void time_point::adjust () {
  if ( seconds >= 60 ) {
    minutes += seconds / 60; // integer division
    seconds = seconds % 60; // integer remainder
  }
  if ( minutes >= 60 ) {
    hours += minutes / 60;
    minutes %= 60; // short form of " = minutes % 60 "
  }
  if ( hours >= 24 ) {
    hours %= 24;
  }
}
time_point::time_point() { // get current time
  time_t ltime;  // holds encoded time
  struct tm *t; // the time and day
  time(&ltime); // get time
  t = localtime(&ltime); // convert time
  hours = t->tm_hour;
  minutes = t->tm_min;
  seconds = t->tm_sec;
}
time_point::time_point (time_point & rv) {
  hours = rv.hours;
  minutes = rv.minutes;
  seconds = rv.seconds;
}
time_point time_point::operator=(time_point & rv) {
  hours = rv.hours;
  minutes = rv.minutes;
  seconds = rv.seconds;
}
time_point::time_point (char * ts) {
  hours = atoi(ts);
  ts += 3; // move pointer past first ':'
  minutes = atoi(ts);
  ts += 3; // move pointer past second ':'
  seconds = atoi(ts);
  adjust();
}
void time_point::randomize(time_point & random_f) {
  time_point now;
  // seed the random number generator:
  srand(now.hours + now.minutes + now.seconds);
  // create a random number between 0 and 1:
  float r = (float)rand()/(float)32767;
  seconds += (int)(r * random_f.seconds);
  minutes += (int)(r * random_f.minutes);
  hours += (int)(r * random_f.hours);
  adjust();
}
int time_point::operator>=(time_point & rv) {
  if ( hours > rv.hours )
    return 1;
  if (hours < rv.hours)
    return 0;
  // here, hours == rv.hours
  if ( minutes > rv.minutes )
    return 1;
  if ( minutes < rv.minutes )
    return 0;
  // here, minutes == rv.minutes
  if ( seconds >= rv.seconds  )
    return 1;
  return 0; // seconds < rv.seconds
}
time_point time_point::operator+(time_point & rv) {
  time_point sum(0);
  sum.seconds = seconds + rv.seconds;
  sum.minutes = minutes + rv.minutes;
  sum.hours = hours + rv.hours;
  sum.adjust();
  return sum;
}
```

♦ ♦ ♦

setjmp() is called at run-time, and can be called in several different spots, not only can you jump anywhere, but you can decide at run-time where you will jump.

Increasing Restart Speed

My code reparses the controller script file every time you restart the system. If this isn't as quick as you'd like (if you restart at very short intervals), then you can rewrite the parsing section to store the information in a linked list of tk pointers (copy the file EVLIST.HXX and modify the linked list).

When the system restarts, you won't have to open and reparse the file. Just build the new list of events from the list of tk pointers. If you're a very good programmer, you can create an image of the list and duplicate it using memcpy() when restarting the system. This is more complicated than it sounds.

A second, much quicker alternative involves creating a small disk cache, just large enough to hold the script file.

If you're concerned about the execution speed, use the "n" command-line flag to eliminate display. The display does crawl. This option, unfortunately, requires you to reboot to get out of the program (or kill the process, in UNIX). You can fix this in MS-DOS by adding the non-ANSI C statement if(kbhit()) exit(0); inside the last while(1) loop.

Future Events

I think this system is ideal for any situation which requires timing. Use it to turn any $350 single-floppy XT into something useful!

Clock time is just the beginning. I plan to control more interesting hardware events. And wouldn't it be nice if you didn't have to have a monitor, keyboard, or even a floppy drive on your XT control system?

To solve these problems and others relating to embedded code, I'll be back for a look into the internals of the XT ROM-BIOS. Stay tuned. P.S. Figure 5, EVENT.CXX, is not referenced.

Editor's note: Bruce often refers to articles in previous issues. You can get a book of his hardware articles, Computer Interfacing with Pascal & C, *and a disk of source code by sending a check for $30 to Micro Cornucopia, P.O. Box 223, Bend, OR 97709, or phoning (800) 888-8087 with Visa/MC.*

If you want to learn more about C++ programming, you can get both disks #1 & #2 of the C++ Source Code Library by sending a check for $25 to Bruce Eckel, Eisys Consulting, 501 N. 36th Street, Suite 163, Seattle, WA 98103.

♦ ♦ ♦

Commenting Disassembler!

SOURCER™

■ **SEE HOW PROGRAMS WORK**
■ **EASILY MODIFY PROGRAMS**

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Includes a definition file facility to include your own remarks and descriptive labels, force data types, and more. Complete support for 8088/87 through 80286/287 and V20/V30 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

On my list of programs that I simply won't do without!
—Robert Hummel, Senior Technical Editor, PC Magazine, 4/26/88

SAMPLE OUTPUT

Fully automatic
Program header
Assembler directives
Determines data areas and type
Detailed comments
Simulator follows segment changes
Easy to read format

```
resetprn.lst    ResetPRN v1.01              Sourcer Listing    2-Mar-89  4:46 pm     Page 1

                        PAGE  60,132
                        ;-------------------------------------------------------------------------
                        ;                               RESETPRN
                        ;
                        ;       Created:   15-Apr-88
                        ;       Version:   1.01
                        ;       Passes:    3          Analysis Flags on: H

= 0008                  data_1e         equ     8               ; (0040:0008=378h)

                        ;-------------------------------------------------------------------------- seg_a -----
                        seg_a           segment para public
                                        assume cs:seg_a, ds:seg_a, ss:stack_seg_b

                        resetprn        proc    far
658E:0000               start:
658E:0000  EB 23                        jmp     short loc_1
658E:0002  52 65 73 65 74 50            db      'ResetPRN v1.01', 0Dh
658E:0008  52 4E 20 76 31 2E
658E:000E  30 31 0D
658E:0011  0040         data_2          dw      40h
658E:0013  0D 0A 52 65 73 65 data_3     db      0Dh, 0Ah, 'Reset Printer? $'
658E:0019  74 20 50 72 69 6E
658E:001F  74 65 72 3F 20 24
658E:0025                       loc_1:
658E:0025  0E                           push    cs
658E:0026  1F                           pop     ds
658E:0027  BA 0013                      mov     dx,offset data_3  ; (658E:0013=0Dh)
658E:002A  B4 09                        mov     ah,9
658E:002C  CD 21                        int     21h             ; DOS Services  ah=function 09h
                                                                ;  display char string at ds:dx
658E:002E  B4 01                        mov     ah,1
658E:0030  CD 21                        int     21h             ; DOS Services  ah=function 01h
                                                                ;  get keybd char al, with echo
658E:0032  3C 79                        cmp     al,79h          ; 'y'
658E:0034  75 16                        jne     loc_3           ; Jump if not equal
658E:0036  8E 1E 0011                   mov     ds,data_2       ; (658E:0011=40h)
658E:003A  8B 16 0008                   mov     dx,ds:data_1e   ; (0040:0008=378h)
658E:003E  83 C2 02                     add     dx,2
658E:0041  B0 08                        mov     al,8
658E:0043  EE                           out     dx,al           ; port 37Ah, printer-2 control
                                                                ;  al = 8, initialize printer
658E:0044  B9 8000                      mov     cx,8000h
658E:0047                       locloop_2:
658E:0047  E2 FE                        loop    locloop_2       ; Loop if cx > 0
658E:0049  B0 0C                        mov     al,0Ch
658E:004B  EE                           out     dx,al           ; port 37Ah, printer-2 control
                                                                ;  al = 0Ch, init & strobe off
658E:004C                       loc_3:
658E:004C  B4 4C                        mov     ah,4Ch          ; 'L'
658E:004E  CD 21                        int     21h             ; DOS Services  ah=function 4Ch
                                                                ;  terminate with al=return code
                        resetprn        endp
                        seg_a           ends

                        ;-------------------------------------------------------------------- stack_seg_b ----
                        stack_seg_b     segment para stack
6593:0000  00C0[FF]                     db      192 dup (0FFh)
                        stack_seg_b     ends

                                        end     start
```

(Source code output and inline cross reference can also be selected)

BIOS SOURCE

■ **CHANGE AND ADD FEATURES**
■ **CLARIFY INTERFACES**

for PS/2, AT, XT, PC, and Clones

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

SOURCER Commenting disassembler	$99.95	**UNPACKER**™ Unpack packed EXE files and more	$39.95
SOURCER with BIOS Pre-Processor	139.95	**ASMtool**™ Assembly source analyzer and flowcharter	89.95

USA Shipping & Handling $3; Outside USA $15; CA Res. add sales tax; PS/2, AT, XT, and PC are trademarks of IBM Corp.

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

VISA **1-800-662-8266** MasterCard

V COMMUNICATIONS, INC.

3031 Tisch Way, Suite 905, Dept. M3, San Jose, CA 95128 (408) 296-4224

Reader Service Number 62

Writing Code For Two Operating Systems —

OS/2 And DOS

Confused by all the OS/2 hype? Not sure you're ready for the world of multitasking, whips, and late binding? How is this going to affect you? Your programs? Your sanity?

How would you like to ignore all the OS/2 versus DOS gibberish and write programs that run under both operating systems? Well, you can, if you're willing to pay a price and willing to read on.

First, I'll review the pertinent part of the Microsoft operating system's family history. Then I'll try to explain some of the new OS/2 capabilities and how our DOS programs can use them.

The Shortcomings Of DOS

MS (and PC) -DOS programmers have always worked around the problems of this single-task operating system. To accomplish multitasking, we dynamically link-in tasks via TSRs (Terminate and Stay Resident programs).

Some TSRs are spoolers (true multitasking programs). Some are pop-up applications (multitasking programs with a manual context switch). Others are more of a shared resource that can be used by other applications. These shared-resource TSRs have an API (Application Program Interface) so that programs can use their services. Sidekick Plus, Btrieve, and the NetWare shell, for example, use APIs.

TSRs that interface to application programs must do so through some sort of dynamic link. Usually they store their API entry point in the interrupt vector table. Applications can then use an INT instruction to call the TSR. I know at least one TSR that makes its residence known to an application and passes the address of its API entry point back to the application via the INT 2Fh multiplexed interrupt. This is similar to OS/2's approach to dynamic linking.

There's another side to this. Since its early versions, DOS has changed (some say, matured) and grown considerably. But for compatibility's sake, all the old parts of the DOS API must be carried over from release to release. This complicates matters considerably. If the APIs were separately loadable, then you wouldn't need to load unused sections. (When was the last time you used any of the FCB series of DOS function calls?)

> If DOS handled the context switching, we wouldn't have all those hot keys tripping over each other.

Of course, it would be easier and cleaner if DOS handled the multitasking and dynamic linking for the application program. For instance, if DOS handled the linking, it could determine which TSRs were needed by an application and then preload the TSRs, if they weren't in memory already. If DOS handled the context switching, we wouldn't have all those hot keys tripping over each other.

The problem with this wish list is that once DOS starts accommodating everyone's changes, other shortcomings surface and cry for improvement as well. (This has already happened on a minor scale.) Just a few more changes and DOS won't be DOS at all. "Creeping featurism, Hobbes, a huge, twitching mass of binary dreams priced out of an average user's grasp."

Enter Uncle Bill & Baby Colossus

So Microsoft took one wish list and developed OS/2. OS/2 has multitasking and dynamic linking built-in, so everything happens transparently. Even the API to OS/2 is implemented as dynamic link libraries. Parts can be independently improved; unused parts can be left out. Is OS/2 dream-DOS? Maybe. Maybe not.

Before we get into that, and since I've already begun using what are probably new terms for many of you, let's define a few commonly used ones in the OS/2 world.

Static Linking: Garden variety linking, resolving external references among a group of object modules.

Static Link Libraries: Files that consist of one or more object modules, which are extracted from the library during static linking to become part of an executable file. For example: runtime libraries for a compiler.

Dynamic Linking: We load an application program, and it tells OS/2 which routines it needs from which Dynamic Link Libraries. OS/2 loads the library (if necessary), and then resolves the references to those functions.

Dynamic Link Library, or Dynalink Library, or DLL: A collection of functions that OS/2 can load into memory for use by other (one or more) applications. These functions must be reentrant. In DOS terms, these are TSRs (with APIs) which can be used by applications.

Exported Functions: These reside in a DLL that can be called by an application. Not quite the same as public. A public routine in a DLL can be called by any other module within the DLL, but not by an application unless it's "exported."

Import Libraries: These perform the same function as a static link library (i.e.,

By Dale Olds
Novell Inc.
122 E. 1700 S.
Provo, Utah 84601

Figure 1—Application Structures for DOS and OS/2

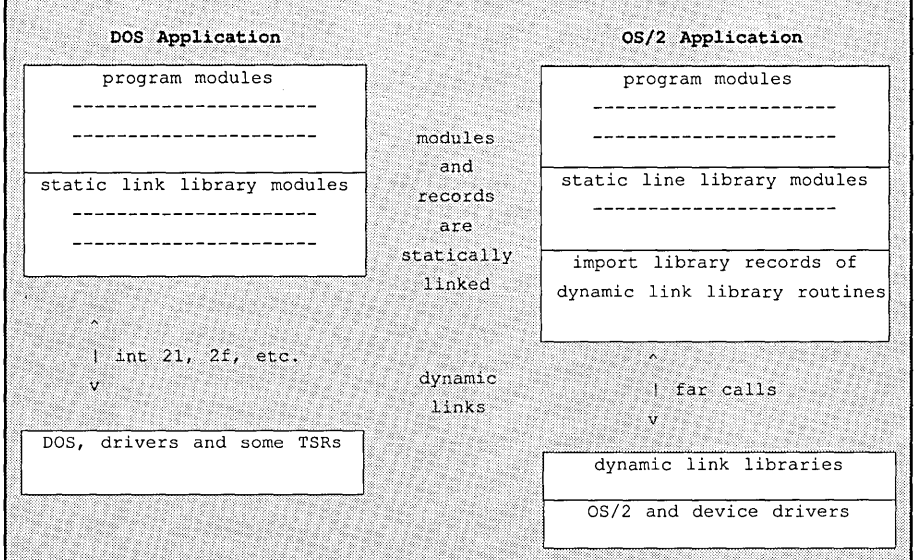
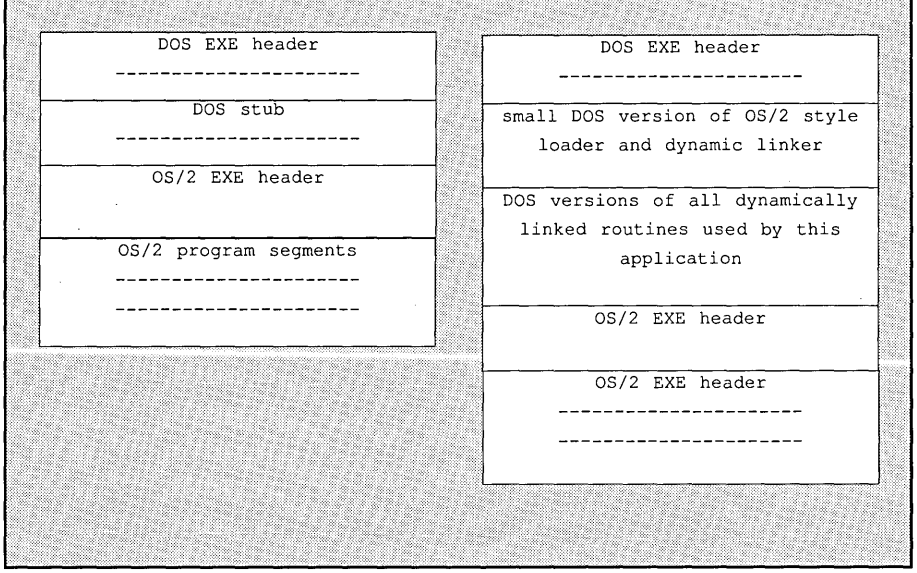


Figure 2 — OS/2 File Structures



they're used by a linker to resolve external references among a group of object modules). They differ, though, because import libraries don't actually contain code. They contain information about routines in a corresponding DLL. The linker uses this information to resolve the external references and to produce records in the executable file so that OS/2 can perform the dynamic link at run time.

Trade Deficit

OS/2 does much more than make up for the problems of DOS. It uses more resources, and it costs. For starters see the comparison of DOS and OS/2 application structures in Figure 1.

The Roots Of OS/2

How well do the two operating systems get along?

OS/2 is new, of course, but its heritage is DOS. Microsoft developed them both. They run under (most of) the same processors. They (at least currently) use the same file systems. They're family, right?

So what happens if you try to execute a DOS program under OS/2? OS/2 knows what a DOS program header looks like and refuses to run it.

And if you try to execute an OS/2 program under DOS? OS/2 executable files begin with a special DOS header and a small DOS program, called a stub (see Figure 2), which writes an error message and terminates. The special DOS header looks just like any other DOS header except for a bit which DOS ignores. So DOS loads the special DOS header and then executes the stub. OS/2 recognizes the special DOS header and skips to the OS/2 header following the stub.

What About All This Code?

Most programs don't need many of the new OS/2 features. A simple utility

SOG is back!

And more regional than ever...

Plan to attend our:

BC SOG . . .

July 7 & 8, 1989
Port Alberni, British Columbia

Rocky Mountain SOG . . .

July 28-30, 1989
Gunnison, Colorado

SOG East . . .

September 7 & 8, 1989
York, Pennsylvania

Long Horn SOG . . .

October 13 - 15, 1989
Dallas, Texas

Read All the Details at the end of Dave's editorial, pages 75-77

to move files from one directory to another doesn't need multiple threads and a task scheduler. Even something like a word processor that's written and functional now in DOS would be nice to have in OS/2. There's a lot of good code written for DOS, and none of it uses any of the new OS/2 features.

All the OS/2 systems I've seen coexist with DOS, so the management of executable files is a problem. Many programs have one executable file for DOS and another one for OS/2, but both (files) have the same name.

One possible solution to this problem would be to replace the DOS stub with the entire DOS application. This still means two programs, each from its own source within a single executable file. Here, each program is identical except for its interface to the operating system.

The solution that the OS/2 development kit provides for both of these problems is essentially to replace the DOS stub with a small OS/2 style loader and dynamic linker and the DOS equivalent for each OS/2 dynamically linked routine.

When DOS executes such a program, this loader/linker reads the OS/2 header and resolves all references to dynamic link routines to the equivalent DOS routines included within the file. When OS/2 executes the program, it skips to the OS/2 header and resolves all references to dynamic link routines to the routines in the separate DLL file.

Bound & Gagged

Called binding, this process is accomplished with the BIND utility included with the OS/2 developers kit or Microsoft C v5.10. One input to BIND is an OS/2 program that only makes use of OS/2 calls for which an equivalent DOS routine exists.

Another input to BIND is complimentary sets of libraries. Each set consists of an import library of functions for OS/2 and a static link library of functions identical in name and calling sequence for DOS.

Also BIND can take a list of functions that it's instructed to ignore. The program assumes the responsibility that the wrong operating system will never call the functions in this list.

Remember that the OS/2 API itself is implemented as a DLL. The import library for these functions provided in the OS/2 developers kit is called DOSCALLS.LIB. Also provided is API.LIB, a static link library of functions to mimic the OS/2 API under DOS.

So DOSCALLS.LIB is actually the OS/2 API, and API.LIB contains the DOS calls to mimic the OS/2 API. Got that? Many of the OS/2 functions begin with the letters "Dos." Reasons for this may be that those functions usually deal with the Disk Operating System as opposed to video or keyboard functions, or that OS/2 is really DOS v5.0. Either way, it's confusing.

API.LIB is simply a static link library that contains functions for DOS identical in name and calling sequences to some of the functions found in DOSCALLS.LIB. DOSCALLS.LIB is an import library of functions found in the OS/2 dynamic link libraries. The functions in API.LIB are therefore callable from OS/2 (through DOSCALLS.LIB) or DOS (through API.LIB) and are referred to as the Family API.

One interesting side effect is that *you can use the API.LIB functions by themselves* in DOS-only programs. This includes some of the "VIO" series of functions that do direct video writes. You can also develop programs under DOS using OS/2 function calls, and then switch them to OS/2 by merely linking in DOSCALLS.LIB rather than API.LIB.

BIND isn't restricted to the set API.LIB and DOSCALLS.LIB. Any DLL that also has a static link library of identical DOS functions has its own "family API." Library developers note.

The Catch

You can bind an OS/2 program (one source) that only uses calls from matched sets of OS/2 import libraries and DOS static libraries to produce a single executable program. The cost of these "bound" or "family mode" programs is consistent with OS/2's cost: more size, less speed.

The executable file grows because it must include the DOS loader/linker and all the code to implement the DOS versions of the dynamic link functions. In my experience this has averaged about 10K. The size increase relates to the number of different dynamic link functions called, not the size of the code.

Loading an application under DOS is slower because the OS/2 style loader/linker now handles it in pieces, then adds the dynamic link step. The application itself executes somewhat slower because the dynamic link functions are always far calls and must pass far pointers. However, in the bound applications that I've used, the speed degradation hasn't been noticeable.

♦ ♦ ♦

GEMS SUPER DISCOUNT* PRICES

MOTHERBOARDS

XT/8088
4.77/8mhz TURBO, 640K,OK $69
4.77/10mhz TURBO, 640K,OK 76
4.77/10mhz TURBO, 1mb, OK, S/W 79
4.77/15mhz SUPER TURBO, w/1mb MEMORY, 2/3 size, 110% faster than 6mhz IBM AT . 420

AT/286 AT SIZE
6/10mhz, 1mb OK, DTK, Ows 229
6/10mhz, 1mb OK, PC CALC, Ows 259
6/12mhz, 1mb OK, PC CALC, ows 289
6/12mhz, 1mb OK, EW, 12mhz CPU, Ows..289
6/16mhz, 1mb OK, PC CALC, Ows 399

AT/286 XT SIZE
6/10mhz, 1mb OK, DTK, Ows 239
6/10mhz, 1mb OK, ELTECH, Ows 229
6/12mhz, 1 or 4mb, OK, MS, Ows 265
6/16mhz, 2 or 8mb, OK, NOVAS 489
6/20mhz, 2 or 8mb, OK, NOVAS 549

AT/386 AT SIZE
16mhz, w / 1mb MICRONICS 1399
20mhz, w / 1mb, MICRONICS 1449
20mhz, w / 1mb, 287 + 387 SOCKET,MS.1415
20mhz, w / 1mb, up to 16mb on M/B,1 par, 1 ser w / 2nd opt ..... CALL
20mhz MYLEX motherboard ..... CALL
25mhz 38C w/1mb ..... CALL

HARD DRIVES

ST-225 Seagate (DRIVE ONLY) $215
ST-225 with CONTROLLER 255
ST-238 Seagate (DRIVE ONLY) 229
ST-238 with CONTROLLER 275
ST-251 Seagate 40ms HARD DRIVE 339
ST-251-1 Seagate 28ms HARD DRIVE ... 419
MR535 Mitsubishi RLL 65mb 28ms 465
ST-125 Seagate 3-½ format 20mb 245
Miniscribe 3650 40mb (61 ms) MFM 315
Miniscribe 3675 63mb (61 ms) RLL 335
MAXTOR....Call For Special Prices ... **$CALL**
TOSHIBA MK130 65mb RLL 3-½ form .. 575
HARD CARD 20mb Plus Development ... 545
HARD CARD 40mb Plus Development ... 729

FLOPPY DISK DRIVES

Fujitsu 360K BLACK FACE PLATE $64
Fujitsu 1.2mb BEIGE FACE PLATE 83
Teac 360K 73
Teac 1.2mb 86
Teac 720K with 5-¼ mount bracket 92
Teac 1.44 3-½ w/ 5-¼ bracket 110
Mitsumi 360K BLACK 63
Sony 720K 3-½ w/ 5-¼ bracket 94
Sony 1.44mb 3-½ w/ 5-¼ bracket 119
Toshiba 720K 3-½ w/ 5-¼ bracket 99
Toshiba 1.44mb 3-½ w/ 5-¼ bracket 119
Mitsubishi 360K 69
Mitsubishi 1.2mb 84
Mitsubishi 720K 3.5 79
Mitsubishi 1.44mb 3.5 95

CASE

XT SLIDE $27
XT/AT Like Case 30
AT 3 Floppy Drive Front / 2 Button 54
AT 3 Floppy Drive with Digital Disp 72
XT SLIDE Heavy Duty, quality 27
XT/AT like, slide, heavy duty, qual 30
AT DT 3DR FT, 2 button, SOL 48
AT DT 3DR FT, 2 button, AS 54
AT DT 3DR FT, 2 button, DIG DISPLAY ... 72
AT DT 3DR FT, DIG DISPLAY/PC CALC 75

SPECIAL CASES

Transportable (case PS KB Mono Monit)$435
Transportable EGA VER 1250
LCD portable 640X400 Bracklite 18lbs ... 759
Carrying Case for portables 35
Tower Case 230W PWR SPLY, 6 half ht . 250
Tower Case w/digital display, 3 Dr. Front will hold 7, ½ht Dr. 220W 225

POWER SUPPLY

150 watt XT Compatible $36
200 watt XT Compatible 49
200 watt AT Compatible 59
220 watt AT Compatible 63
250 watt AT Compatible 69

KEYBOARDS

84 KEY At Style Keyboard $36
84 Key At Style Maxi Switch Keyboard 49
101 Key Enhanced Monterey 42
101 Key Tronics Keyboard 47
101 Maxiswitch Enhanced Keyboard 64
101 Tactile Enhanced Keyboard 49

MOUSE

Logitech M-8 Button $35
Logitech C-7 Button 69
Logitech HI-REZ 86
Microsoft Bus Mouse + Paintbrush/Menu . 99
Microsoft Ser Mouse + Paintbrush/Menu . 99

VIDEO CARDS

Monochrome W/ Printer Port $39
Monochrome 132 Column & Print Port ... 53
Color Graphics Adaptor 39
Color Graphics Adaptor 1/ print port 53
Micro EGA Auto Switching 129
Video-7 VGA 269
Paradise VGA Plus 259
Paradise VGA Professional 389
Chips and Technologies VGA Card 249

MODEM

1200 BAUD with Software $52
2400 BAUD with Software 95
Zoom HC 2400 w/Procomm Software ... 116
External 1200 w/PC Talk III S/W 62
External 2400 w/PC Talk III S/W 109
Everex External 2400 Modem 179

I/O CARDS

Multi I/O Ser Par Cal Clk Game Disk $48
Multi I/O Above (DFI) 66
Magic I/O Ser Par Cal Clk Disk Ctlr 360K, 720K, 1.2mb 49
I/O Plus.. Ser Par Cal Clk Game 42
Parallel Card (XT) 15
Serial Card 2nd Serial Optional 18
Port Serial Card w/2 Port & Opt 66
2nd Serial for XT 17
Game Card (2 Ports) 17
AT I/O CARD par, game, ser, 2nd ser opt . 42
2nd Serial Port for AT 19

Floppy/Hard Disk Controller

2 Drive Floppy CTLR (360K/720K) $17
4 Drive Floppy CTLR (360K/720K) 34
2 Drive Floppy Master (360K/1.2mb) 39
4 Drive FDC (360, 1.2/720, 1.44)SEF 89
4 Drive Super Special FDC W/Cables 89
WD FOX 2 Floppy Drive CTLR For AT 57
WD WX 1 Hard Disk Controller 8 Bit 57
WD XT/GEN Hard Disk CTLR For XT 52
WD 27X RLL Hard Disk Controller 59
WD WA-2 FD/HD CTLR For AT (MFM) ... 112
DTC 5287 FD/HD CTLR For AT (RLL) 154
NCL 5425 FD/HD CTLR For AT (MFM) ... 112
ADAPTEC 2372 FD/HD CTLR AT (RLL) .. 179
16 Bit 4 Floppy/2 Hard Drive Controller Modified 16 Bit DTC CTLR MFM & RLL . 295

TAPE BACKUP

Teac 60mb MT2ST/45 (CASS) INT $539
Everex 60mb Wangtek INT (Cart) 679
Colorado 44mb (Cart) Internal 289

MONITORS

Goldstar Amber Monochrome 720X348 . $58
Samsung 1252 Mono 720X348 TIL/SW .. 66
Samsung 1464 EGB 219
Eversion 14 Amber Flat Screen 119
Amdek 410A 139
Nec GS 199
Taxan Composite Amber/Green 89
Relisys EGA with TILT & SWIVEL 339
Relisys Multi Scan 499
Mitsubishi 1410XC EGA 379
Mitsubishi 1381A Diamond Scan 539
Nec Multiscan II 569
Sigma Designs Laserview 1901-PC 1749

CAD PRODUCTS

MITSUBISHI HA3905 19V/20" MONITOR, ANALOG/TTL, 1024 X 1024, 15.7 35.5khz $1725
MITSUBISHI HL6905 19" MONITOR, 1280 X 1024, 30% 64khz AUTO TRACKING 2415
HIGH RESOLUTION VIDEO CARDS FOR CAD
QDP VIVA 1280 1290 X 768 16 COLORS . $1289
QDP VIVA 2000 1024 X 1024 16 COLORS..1559
QDP VIVA 2000 2 2024 X 2024 16 COLORS.1895

CHIPS (Prices Subject To Change)

V-20 8mhz (replacement for 8088) $10
V-20 10mhz 16
8088-2 CPU 5
8087-2 MATH CO-PROCESSOR FOR XT . 149
8087-1 MATH CO-PROCESSOR FOR XT . 215
80287-8 MATH CO-PROCESSOR FOR AT . 229
80287-10 MATH CO-PROCESSOR FOR AT.279
80387-16 MATH CO-PROCESSOR FOR AT..425
80387-20 MATH CO-PROCESSOR FOR AT.CALL
64K 150ns DRAM 1.35
64K 120ns DRAM 3.25
64K 100ns DRAM 4.25
64K 120ns DRAM 4464 ..... 13.50
256K 150ns 10
256K 120ns 11.75
256K 100ns 11.95
256K 80ns 12.95
1mb 100ns 33
256K 100ns SIMM'S (SIP's) 125
256K 80ns SIMM'S (SIP's) 145
256K 100ns STATIC COLUMN RAM 17
256K 80ns STATIC COLUMN RAM 18

Desktop Publishing

Hi-Res 1024 15"MONITOR W/Video Controller 1024 X 768 Resolution $499

SCANNERS

Mitsubishi Hand Scanner Full Page MH216 200 DPI, 8.5" X 11" $638
Mitsubishi 400 DPI MH 130 ..... CALL
Mitsubishi Paper Feeder For Scanner 176
Abaton 300 FB Full Page Scanner 1349

ACCELARATOR CARDS

MICRO 286-10 OK $269
SOTA 286i 10mhz OK 298
SOTA 286i 12mhz OK 359

MOTHERCARDS

SOTA 5.0 MOTHERCARDS w/1mb, 10mhz.$779
SOTA 5.0 MOTHERCARDS w/1mb, 12mhz.879

Econo XT® Compatible

8mhz Turbo M/B,0K, case, Keyboard, 150w P/S, FLOP CTLR, 1 360K Drive, Mono video B.D.
MONITOR **$339**

SPECIAL

Lap Top Computer

The New Mitsubishi MP286L
12mhz,1.44,20mb.....**$2475**

Call for Discounted System Prices

***All prices shown are pre-paid or ordered by VISA or Mastercard. Charge card orders are subject to a 3% surcharge. For C.O.D. or term orders on parts above add 10%.**

XT® AT® & IBM® Are Registered Trademarks of International Business Machines.

GEMS computers

3446 De La Cruz Boulevard
Santa Clara, California 95054

In Business Since 1985
A member of the Better Business Bureau and Chamber of Commerce.

HOURS: M-F: 9 A.M. - 9:00 P.M. EST
7:00 A.M. - 7 P.M. PST / SAT: 8 A.M. - 6:00 P.M. PST

1-800-332-GEMS

IN CA 408-988-0161 TECH SUPPORT 408-988-0146
FAX 408-988-0609

Shipping: 4% plus $3.00 handling on all part orders (except cases, 9% + $3.00). APO/FPO orders add 8% plus $3.00 on part orders. Call for exact charges on Systems and Monitors. CA residents add 7% tax. Prices reflect 3% cash discount.

One Year Warranty On All Parts And Systems / All Orders Are F.O.B. Santa Clara.

Reader Service Number 130

Ghosts Of Comdex Past

It's Not The Place to See What's NeXT

I always look for blockbusters at Comdex. I'm still looking. This year I was most curious about NeXT. They weren't there. I expected something exciting from Borland. They weren't there either.

Boy, Comdex was a disappointment this year. The one time I wanted to see the majors, I couldn't find them. Every other year I couldn't find anything else.

So, in lieu of a major company I headed for Sony's booth to see the read/write optical drive that NeXT reportedly uses. (Dingely, or however he calls himself, wrote that Sony was delaying its announcement so that NeXT could have the limelight.)

Boy, was there a crowd at the Sony booth:

"Do you have a NeXT machine?"

"Can I see the drive?"

"Do you have one running?"

"How many times can you erase the media?"

"Why only a million times?"

"Where's the Jobs machine?"

"How many times can you erase the media?"

It turns out that Canon developed NeXT's 250 meg optical drive. It uses one head and one side of the media. Sony's drive also has one head, but you can use both sides of the disk. (It's a classic flippy.) Sony gets 600 megs onto their disks, 300 on each side.

As far as Sony can tell, you can erase and rewrite the media forever; but they've only done it a million times so that's what they're specifying.

I just received Sony's literature, dated February 3 (complete with credit application). According to this, they're currently shipping the drives in quantity. The 5 1/4" full height internal drive with RLL SCSI controller runs $4,650. You have to furnish the SCSI adapter for the host system. The external unit with power supply runs $5,250. Media will cost you $230 per disk (not exactly unbranded bulk prices).

They're predicting you can use a disk constantly for 100 years before seeing significant degradation in the read/write performance.

If you're interested, you can contact them at:

Sony Corporation of America
Optical Storage Systems Division
5665 Flatirons Parkway
Boulder, CO 80301
(303) 444-3200

How They Work

It turns out that Sony uses the same technology as both Canon and a West German disk media manufacturer named Hoechst. Hoechst sent real engineers to Comdex so I hung around their booth.

Their literature described the process as: " To record a digital bit, a small area of the film is heated by the laser. The polarity of the magnetisation is reversed by simultaneously applying an external magnetic field.

"When striking a recorded bit with a lower power read laser, the state of polarisation of the light changes (Kerr effect)...

"To erase, the write laser heats the magnetic layer beyond the Curie temperature, while the outer magnetic field is reversed. There's no limit to the number of read/write/reverse cycles.

"Except for the conditions of erasure, the recorded information cannot be altered by external magnetic fields. The special design of the barrier layers protects the recording layer from corrosion."

They offer 5 1/4" media that's similar to Sony's. They also have a 3 1/2" version which holds 150 meg per side. They're projecting costs of $5 to $10 to manufacture a 5 1/4" optical disk, so they predict retail prices in the $50 range.

I'm really excited about this technology. It looks like it'd take some effort to damage the information — you have interchangeable media and lots of space. If costs come down (there's no reason they won't) and if the system's reliable, this technology should completely replace standard hard drives. Eventually, it may also replace the common floppy.

Hoechst (U.S. Office)
Advanced Technology Group
One Main Street
Chatham, NJ 07928
(201) 635-3910

Logitech

Logitech announced desktop software, debuggers, and scanners, but the real news was their ClearCase Mouse. No gutless wonder, this 200 dpi rodent really showed its stuff. Never have I seen so many willing to take mouse guts in hand. (It's the perfect gift for your computerized sister-in-law.) But you gotta hurry; with winter here, these tiny creatures might grow fur.

Automatic Programmers

I saw two automated programmers. Sorta.

The first, called Cause, reminded me of those classic menu driven database packages. They generate instant applications with instant limitations.

"Ports? You're interested in ports? Check the index under wine-making applications."

However, Cause may well be as powerful as dBASE (okay, more powerful than dBASE). Plus, it lets you save your menu selections in a file. To create a program, you just run through the menu selections once. Cause will save your steps and replay them. They call the replays "Effects," and the company is offering to market selected Effects for authors. (Reminds me of Lotus' macros.)

Cause will run on both the PC and the Mac and any Effects you write should

By Dave Thompson
Micro C Editor

run identically on both.

I hadn't intended to be impressed by Cause, but it displayed an impressive instruction set (okay, an impressive list of menu selections). Plus, I talked to the lead software designer and some of his extensions sound awesome. (Ports? Did I hear ports?)

I'm on their list for an early copy (they said it would be out Monday, though they didn't say which Monday). I'll let you know if they meet expectations.

Maxem Corporation
1550 East University
Mesa, AZ 85203
(800) 336-6296

A Tool For The Rest Of Us

Meanwhile, I found a tool for real (quiche-eating) programmers.

You struggle through those nasty little logic problems, don't you? Admit it, we're talking about the heartbreak of Warnier-Orr. You diagram the logic, plot it, make sure you've left no ambiguity unresolved.

Once it works on paper, you write the code. It's that fun kind of code: if, else if, else if, else if, else if... sorta stuff. The stuff that guarantees you'll be finding funny features forever. (What job security.)

If you're lucky, that's only 20% of your code; and if you're really lucky, it'll work.

But perhaps all that is past; Sterling Castle showed off Logic Gem. You fill out a logic diagram (reminds me of a spreadsheet), and as you go it shows you which logic states remain ambiguous. Once you've removed all the ambiguities, Logic Gem writes the code for you.

What does it write? How about C, structured BASIC, interpretive BASIC, Pascal, FORTRAN, dBASE, and English? (Don't ask me about English, I have yet to order dinner in Pascal successfully.) Is the finished code readable? As readable as 27 layers of nested IFs will ever be.

TABLE: parser.tab LABEL 1
RULE COMP: 128 : 128

		M	1	2	3	4	5	6	7	8
C1	length == 1	2	y	n	n	n	n	n	n	n
C2	c == '\n'	2	–	y	n	n	n	n	n	n
C3	c == BACKSPACE	2	–	–	y	y	n	n	n	n
C4	i == 0	2	–	–	y	n	–	–	–	–
C5	typ == NUMERIC	2	–	–	–	–	–	y	y	n
C6	isdigit(c)	2	–	–	–	–	–	y	n	–
C7	i == length	2	–	–	–	–	y	n	n	n
A1	beep()				1		1		1	
A2	addch(c):refresh()		1					1		1
A3	str[i]= c		2					2		2
A4	str[i+1] = '\0'		3					3		3
A5	backup(1)					1				
A6	i++							4		4
A7	break		4	1						
	Frequency		20	10	1	2	5	20	2	40
	Cost		0	0	0	0	0	0	0	0

A Logically Complete And Mechanically Perfect Table from Logic Gem

Logic Gem doesn't generate your whole program, just the IFfy part. Written by the New Shoes Software Company (a very straight-laced outfit), you can get a runable (not bootable) package if you're well-heeled enough to come up with $198. (Includes step-by-step manual.) I understand they're not going to hike the price any time soon.

Logic Gem
Sterling Castle
702 Washington Street, Suite 174
Marina del Rey, CA 90292
(800) 722-7853
(800) 323-6406 (CA)

If You've Got The Time

Let's say you're producing a newsletter. And you'd love to do all those fancy things your desktop software would do if only you had a $4,000 PostScript printer. But if you had to pony up $4,000 for a silly printer, you'd be doing your fine dining at your Safeway dumpster.

However, there may be help (we're talking printing again). All you have to do is fire up your desktop package, tell it you've got a PostScript printer, and away you go. When it's time to print, tell it to create a file.

Then you use GoScript to translate that PostScript output file into something intelligible to your:

HP LaserJet Series II
HP LaserJet plus and emulations
HP DeskJet
Hewlett-PaintJet
Cannon BubbleJet
Epson FX series
NEC 24-pin Pinwriters
IBM Proprinter
Toshiba 24-pin
and more

Obviously, quality and speed of output vary depending on the printer. But Sandy saw this package run, and she reported seeing a silly little HP Ink Jet printer produce beautiful quality PostScript pages.

Unfortunately, the DeskJet took half an hour to produce a page. (I'd be curious about how well the package supports serious graphics and unusual type manipulations, but apparently it handles standard text pages just fine.)

The package with 13 type faces retails for $195. With 35 faces (the standard set built into PostScript printers), it's $399. It's not cheap, but it should keep you out of dumpsters.

Laser's Edge
800 W. Burlington
Fairfield, IA 52556
(515) 472-4789

♦ ♦ ♦

Designing A ROM Monitor

Karl's 68000 Surplus Board Gets A Brain

We've followed along as Karl decoded the hardware on his surplus 68000 bargain. Now he walks us through his design of a ROM monitor.

My surplus 68000 system needed a ROM monitor. I had written other monitors for other CPUs, so the actual coding wasn't going to be a problem. The design of the monitor presented the problem; which features did I want and what was the best way to implement them?

The Wish List

The design of the target machine, a Convergent Technologies Mini-Frame CPU board, dictated some of the features of my new ROM monitor.

The monitor I created does system initialization and a few simple hardware checks. It also handles character I/O for the two ports on the Intel 8274 serial chip and supports the Western Digital 2793 Floppy Disk Controller (FDC).

I also included some more general features. The ROM monitor uses a minimum of absolute RAM space. It provides formatted text output routines and lets you access it from either serial port. Programs access the monitor's routines via a vector table so I can modify the ROM with minimum impact on current programs.

Some requirements I imposed on the monitor design are a little hard to justify; I included them because they filled a sense of "rightness" about this program. It isn't absolutely necessary that a single-user ROM monitor be reentrant and recursive, but this one is. I wanted to play around with multi-user programming at the monitor level.

Where To Begin

Since this monitor was my first large 68000 program, I browsed through the local Waldenbooks for a text on 68000 assembly language programming that might provide help. I found what may be the best practical manual available to the hobbyist.

68000 Assembly Language, by Donald Krantz and James Stanley (published by Addison-Wesley), is more than your typical book on assembly language programming.

It's full of code for *large* working programs, complete with detailed discussions on the techniques. Besides being a quick, accurate reference to the instruction set, this book provides finished, debugged routines for handling tasks such as formatted text output and case constructs. Recommended for anyone doing 68000 assembly work.

The full monitor consists of several modules, assembled separately and linked before dumping into EPROM. I wrote the code on a PC clone using the Sidekick notepad editor before running it through 2500 A.D.'s cross-assembler/linker. The linker generated Motorola S1-S9 records, which I then sent to a DATA-I/O Model 19 EPROM programmer.

If you use a different assembler (such as a public domain program) that only supports a single large assembly, you will probably have to mash all these modules together, then weed out the duplicate labels. You'll also want to alter some of the weirder address references, inserted to keep the linker happy.

The assembler I used wants the LONG/WORD/BYTE pseudo-ops for storage allocation; these are analogous to the DC.L/DC.W/DC.B pseudo-ops used by Motorola. The only other thing to watch for is the character used for the program counter; this assembler uses $, where others might use *.

The Vector Table

The typical 68000 system follows a specific sequence after a hard reset (such as power-up). First, the CPU loads the 32-bit address from location $0 into the stack register, A7. Second, it loads the 32-bit address found at location $4 into the program counter as the address of the first executable instruction.

Finally, it jumps to that address to begin program execution. Much more goes on, but this covers what's important to starting the monitor. If you want a full description of the startup sequence, get a copy of the Motorola 68000 16/32-bit *Microprocessor Programmer's Reference Manual*, M68000UM AD4.

Since these two values must be immediately available on power-up, they are usually burned into ROM and appear at addresses $0-$7. Within a few clock cycles after reset, the ROM gets readdressed up out of the way of the CPU's RAM-based execution vector area. In the Mini-Frame, the monitor ROM moves to address $800000. Therefore, all following discussions about the monitor assume the code exists at $800000.

Figure 1 (JUMP.ASM) shows the monitor's jump vector table. Although the vector table is an old idea, it still solves some subtle problems associated with monitor design.

Most monitors (and this one is no exception) contain several powerful internal routines for tasks such as character I/O, system initialization, etc. Naturally, you want to be able to use these routines in your application programs.

If you write a program expecting a monitor's routines to always be at certain addresses, you quickly run into conflicts attempting to upgrade your monitor. If you move the routines within the monitor, all your applications programs have to be rewritten. If you try to change your monitor code without moving entry points, you eventually end up with ugly-looking, spaghetti code in your EPROM.

A jump table provides a simple solution. It gives you fixed entry points into the monitor's routines while giving you

By Karl Lunt
7349 W. Canterbury
Peoria, AZ 85345

the option of moving the routines anywhere you wish.

The COLDS, WARMS And WARMS1 Vectors

COLDS, WARMS and WARMS1 are restart entry points. A jump or branch to COLDS completely restarts the monitor. In fact, the reset execution address given to the CPU on power-up is that of COLDS. WARMS and WARMS1 will reenter the monitor without doing a lot of system initialization. The only difference between the two is that WARMS1 resets the stack pointer, while WARMS doesn't. Refer to Figure 2 (MONITOR.ASM) for more details.

Entry at COLDS first sets the four status LEDs to a known pattern, then tests the mapping RAM on the Mini-Frame. This RAM provides the translation between the physical addresses generated on the CPU's address lines and the logical addresses of the on-board 512K RAM.

If the mapping RAM is not working properly, chances are the Mini-Frame will not work. If COLDS cannot write a value to this RAM, then read back that same value, it assumes the mapping RAM is defective and dies after lighting a special pattern of LEDs.

Next, it activates and tests the lowest 4K block of on-board RAM. Again, a failure causes a halt with a distinctive LED display.

The monitor then calls ACTRAM to activate all available RAM (without checking to see if there really is RAM there) by writing the correct logical translation pattern for each 4K block of RAM into the proper mapping RAM address.

The call to SINIT sets up the two serial ports for 9600 baud. PRINTF then displays a sign-on message. Note that if the channel 0 serial port is inoperative, the machine will stop, displaying another unique LED display.

Next, the monitor calls TESTRAM to see how much RAM is available. The RAM test is nondestructive; after it tests a location, it restores the original contents.

I dislike destructive RAM tests, especially considering how little additional code is required to preserve the original data. This feature helps in tracking down those bizarre little bugs that force a system reset; the code you were working on doesn't go away after you hit the Big (reset) Button.

Following the call to TESTRAM, PRINTF displays the amount of RAM that tested good. Finally, PRINTF gives a page of information on how to use the monitor.

The code around label WARMS$ gives a clue to one powerful aspect of the ROM monitor. Notice that WARMS$ forms a small loop that sets up a register dump vector (via DUMP_REG$), calls the monitor subroutine (MONITOR$), prints a message telling the user that he can't EXIT the monitor, then branches back to WARMS$ and starts all over again.

This shows that the actual monitor is only a subroutine, available to any program via the vector at MONITOR$. Any application can branch to that address and have full, interactive, access to all monitor functions.

I find myself using this feature often. If I want to investigate the effects of a program I am developing, it's often easiest to invoke the monitor from within the operating system (I'm using SK*DOS). I can do my prowling around, then give the monitor an EXIT command, which returns me to SK*DOS.

Perhaps I ought to clarify something here. It isn't possible to EXIT from the monitor if you entered the monitor in the normal fashion; that is, following a hard reset or power-up. However, if you enter the monitor using a BSR to MONITOR$, typing an EXIT command at the monitor prompt executes an RTS instruction, returning you to the program that called the monitor.

Other Useful Entry Vectors

The PROCESS entry vector allows a program to execute a single monitor command. The calling program first fills a null-terminated string with the text for the command, puts the address of the string in register A0, then JSRs to the PROCESS entry vector. The command will be executed and, when complete, control returns to the calling program.

There are many other entry points providing access to routines for: character I/O, floppy disk support, and string manipulation. Two in particular get a big workout. PRINTF is a C-style text output routine, complete with control string and number formatting for decimal and hexadecimal values. CASE does simple control-switching similar to the Pascal CASE statement.

I lifted both modules, virtually intact, from *68000 Assembly Language*. All credit for this code goes to Krantz and Stanley. It was a real pleasure to borrow such well-crafted code (and not have to debug the stuff).

PRINTF fills an important need in this monitor. Earlier monitors, notably the 6809 monitors based on the ROM from Southwest Technical Products, offered character output and string output only. You could call routines inside these monitors to output numbers as well, but it was painfully difficult to display numbers inside lines of text.

After my last 6809 ROM monitor, I swore I would never write another monitor without adding some formatted text output routine. PRINTF does the job beautifully.

You use PRINTF just like a C compiler would use it. First, push all the arguments onto the stack, starting with the rightmost argument. Finally, push onto the stack the address of the control string.

PRINTF will process the control string, pulling each argument from the stack as required. When PRINTF returns

Figure 1 — Jump Table

```
* This is the old-fashioned jump table, stored at the
* start of the ROM. It provides BRAs (not BSRs) to the
* more important routines in the monitor ROM. Any user
* program should reference these locations only. Don't
* access locations inside the body of the ROM code.
        PUBLIC    COLDS,WARMS,WARMS1,MONITOR
        PUBLIC    PRINTF,CASE
        PUBLIC    GETNXTFLD,GETHEX,GETLINE,PROCESS
        PUBLIC    SINIT,ACTRAM
        PUBLIC    PROMPT,UPPER
        PUBLIC    GETC,GETC0,GETC1
        PUBLIC    PUTC,PUTC0,PUTC1
        PUBLIC    KEYPRSD,KEYPRSD0,KEYPRSD1
        PUBLIC    FDREAD,FDWRITE,FDHOME
        PUBLIC    FDMTRON,FDMTROFF
        PUBLIC    FDWRTRK,FDRDTRK
        PUBLIC    SKROMTBL

* The following branch references are to externally
* declared labels that preface the actual code. This
* code will be added at link time. By convention, all
* internally-accessed monitor routines end in '$'.
        EXTERNAL  COLDS$,WARMS$,WARMS1$,MONITOR$
        EXTERNAL  PRINTF$,CASE$
        EXTERNAL  GETNXTFLD$,GETHEX$,GETLINE$,PROCESS$
        EXTERNAL  SINIT$,ACTRAM$
        EXTERNAL  PROMPT$,UPPER$
        EXTERNAL  GETC$,GETC0$,GETC1$
        EXTERNAL  PUTC$,PUTC0$,PUTC1$
        EXTERNAL  KEYPRSD$,KEYPRSD0$,KEYPRSD1$
        EXTERNAL  FDREAD$,FDWRITE$,FDHOME$
        EXTERNAL  FDMTRON$,FDMTROFF$
        EXTERNAL  FDWRTRK$,FDRDTRK$
        EXTERNAL  SKROMTBL$

* This module should be assigned a link offset of
* $800000 at link time, to force RESET stack pointer
* and RESET PC values to occupy physical addresses 0-7
* at reset time. Note that there is no guarantee that
* the Mini-Frame will have RAM available at the initial
* stack pointer address. The RAM at that stack address
* must be activated by the monitor before it can be
* used by a BSR!
        LONG      $005F00     ADDRESS OF INITIAL SP
        LONG      $800008     ADDR OF BRNCH TO COLD STRT
COLDS:  JMP       COLDS$      COLD START ENTRY POINT
WARMS:  JMP       WARMS$      WARM START ENTRY POINT
WARMS1: JMP       WARMS1$     WARM START THAT RESETS SP
MONITOR: JMP      MONITOR$    ENTRY TO MONITOR (USE JSR)
PRINTF: JMP       PRINTF$     ENTRY TO PRINTF ROUTINE
CASE:   JMP       CASE$       ENTRY TO CASE ROUTINE
GETC:   JMP       GETC$       ENTRY TO GETC ROUTINE
GETC0:  JMP       GETC0$      ENTRY TO CHNL 0 GETC
GETC1:  JMP       GETC1$      ENTRY TO CHNL 1 GETC
PUTC:   JMP       PUTC$       ENTRY TO PUTC ROUTINE
PUTC0:  JMP       PUTC0$      ENTRY TO CHNL 0 PUTC
PUTC1:  JMP       PUTC1$      ENTRY TO CHNL 1 PUTC
KEYPRSD:   JMP    KEYPRSD$    ENTRY TO KEYPRSD ROUTINE
KEYPRSD0:  JMP    KEYPRSD0$   ENTRY TO CHNL 0 KEYPRSD
KEYPRSD1:  JMP    KEYPRSD1$   ENTRY TO CHNL 1 KEYPRSD
GETNXTFLD: JMP    GETNXTFLD$ ENTRY TO GETNXTFLD ROUTINE
GETHEX: JMP       GETHEX$     ENTRY TO GETHEX ROUTINE
GETLINE:   JMP    GETLINE$    ENTRY TO GETLINE ROUTINE
PROCESS:   JMP    PROCESS$    ENTRY TO MONITOR PROCESSOR
SINIT:  JMP       SINIT$      ENTRY TO SER INIT ROUTINE
ACTRAM: JMP       ACTRAM$     ENTRY TO RAM ACTVATN RTNE
PROMPT: JMP       PROMPT$     ENTRY TO MON PROMPT RTNE
UPPER:  JMP       UPPER$      CONVERT STRING TO UPCASE
FDREAD: JMP       FDREAD$     READ ONE SECTOR FROM FD
FDWRITE:   JMP    FDWRITE$    WRITE ONE SECTOR TO FD
FDHOME: JMP       FDHOME$     BRING FD HEADS TO TRACK 0
FDMTRON:   JMP    FDMTRON$    START FD MOTOR, WAIT A BIT
FDMTROFF:  JMP    FDMTROFF$   TURN FD MOTOR OFF
FDWRTRK:   JMP    FDWRTRK$    WRITE AN ENTIRE TRACK TO FD
FDRDTRK:   JMP    FDRDTRK$    READ ENTIRE TRACK FROM FD
SKROMTBL:  JMP    SKROMTBL$   RET ADDR SK*DOS CMD TABLE

* ROM addresses from $800000 to $8000FF are reserved
* for jump vectors. Any monitor program should start at
* $800100 and no monitor routine should reside below
* that address, as it may be overwritten in later
* versions of the monitor.
```

♦ ♦ ♦

Figure 2 — Main Monitor Routine

```
* MONITOR - Main routine in the 68010 EPROM.
* Note that there is no ORG statement in this block. It
* should be LINKed to appear at address $800100, as
* that is where the COLDS vector is aimed.
    EXTERNAL  PRINTF,SINIT,PUTC,GETC,GETLINE
    EXTERNAL  DUMPMEM,FILL,SWEEP
    EXTERNAL  PROMPT
    EXTERNAL  CASE,UPPER
    EXTERNAL  ACTRAM,TESTRAM
    EXTERNAL  CHANGE
    EXTERNAL  S19,GO
    EXTERNAL  MONITOR,WARMS1
    EXTERNAL  DUMP_REG$
    EXTERNAL  SELPORT
    EXTERNAL  FLOPPYRD,FLOPPYWR
    EXTERNAL  FDBOOT
    PUBLIC    WARMS$,COLDS$,PROCESS$,MONITOR$
    PUBLIC    WARMS1$

* Assignments of RAM locations for this monitor assume
* that the user will be running SK*DOS at some point.
* Thus, the RAM test location, I/O vectors and monitor
* stack reside in the block of RAM between $00C0 and
* $09FF reserved by SK*DOS for the ROM monitor.
* The RAM used by this monitor looks like this:
*
*    ----------------------------    $09FF
*    |  RAM test location (TLOC) |   $09FC
*    |---------------------------|
*    |  RAM vector to GETC       |   $09F6
*    |---------------------------|
*    |  RAM vector to PUTC       |   $09F0
*    |---------------------------|
*    |  RAM vector to KEYPRSD    |   $09EA
*    |---------------------------|
*    |  Future monitor use       |   $0980
*    |  Top of monitor stack     |   $0980 - $00C0
*    |---------------------------|
*
* To support SK*DOS, the user stack pointer is set to
* $0E00. This is actually only a precautionary measure,
* as the Mini-Frame crashes when put into user mode. (I
* think it is expecting user memory to be available off
* board; in any case, the USP should get a safe value.)
BUFFER   EQU  -256          STACK-BASED TEXT BUFFER
*                           STACK MUST ALLOW 256 CHARS
*                           SEE BELOW FOR USE
TLOC:    EQU  $9FC          RAM TEST LOC. USED TO SHOW
*                           THAT A PREVIOUS RESET
*                           ALREADY TESTED RAM.
MON_SP:  EQU  $980          STACK PTR USED BY MONITOR.
*                           THIS IS VAL WRITTEN TO A7
*                           BY BOTH COLDS AND WARMS1.
USER_SP: EQU  $E00          SET USER STACK PTR AS
*                           NEEDED TO SUPPORT SK*DOS.

COLDS$:  MOVE.W    #$3700,$450000 LEDS:  _ _ G _ R
    MOVE.L    #$400000,A0     POINT AT START OF MAP RAM
    MOVE.W    #$3FF,D0        SET A COUNTER
MAPTEST: MOVE.W    #$01A5,(A0)    WRITE A WORD
    MOVE.W    (A0)+,D1        READ IT BACK
    AND.L     #$01FF,D1       MASK OUT CONTROL BITS
    CMP.W     #$01A5,D1       NOW TEST IT
    BNE.S     RAMFAIL         BRANCH IF FAIL
    DBF       D0,MAPTEST      COUNT THIS WORD
    BRA.S     STARTUP         BRANCH IF ALL GOOD
RAMFAIL: BRA.S     RAMFAIL        DIE HERE WITH LEDS ON
STARTUP: MOVE.W    #$3E00,$450000  LEDS:  R _ G _ _
    MOVE.W    #$2000,$400000  SWITCH ON LOWEST 4K BLOCK
    MOVE.L    #MON_SP,A7      PUT STACK IN LOWEST RAM
    MOVE.L    #$A5A5A5A5,$0FFC  TEST A LOCATION
    CMP.L     #$A5A5A5A5,$0FFC
    BNE.S     RAMFAIL         BRANCH IF FAILURE
    MOVE.W    #$3B00,$450000  LEDS:  _ _ G Y _
    BSR       ACTRAM          ACTIVATE ALL SYSTEM RAM
    MOVE.W    #$3900,$450000  LEDS:  _ Y G Y _
    BSR       SINIT           INITIALIZE THE SERIAL PORT
    MOVE.W    #$3D00,$450000  LEDS:  _ Y G _ _
    MOVE.L    #HELLO,-(A7)    GET FIRST MESSAGE
    BSR       PRINTF          DISPLAY IT
    ADDQ.L    #4,A7           ADJUST STACK
    MOVE.W    #$3F00,$450000  LEDS:  _ _ G _ _
    CMP.L     #$A5A5A5A5,TLOC BEEN THROUGH THIS BEFORE?
```

```
    BEQ.S     WARMS1$          SKIP TEST IF SO
    BSR       TESTRAM          TEST ALL AVAILABLE RAM
    MOVE.L    D0,-(A7)         PUSH SIZE OF RAM
    MOVE.L    D0,-(A7)         PUSH IT AGAIN
    MOVE.L    #SIZE,-(A7)      PUSH MESSAGE
    BSR       PRINTF           DISPLAY IT
    ADD.L     #12,A7           REPAIR STACK
    MOVE.L    #$A5A5A5A5,TLOC WRITE FLAG TO SHOW WE'VE
*                              BEEN HERE ALREADY
HELP$:   MOVE.L    #FIRST,-(A7)    GET SIGN-ON MESSAGE
    BSR       PRINTF           PRINT IT
    ADDQ.L    #4,A7            FIX THE STACK
WARMS1$: MOVE.L    #MON_SP,A7      ENTRY POINT RESETS THE
SP
    MOVE.L    #USER_SP,A1      GET THE USER STACK PTR
    MOVE.L    A1,USP           AND SET UP THE POINTER
WARMS$:  BSR       DUMP_REG$       REINSTALL DUMP REGS
VECTOR
    BSR       MONITOR$         BRANCH TO MONITOR WITHOUT
*                              RESETTING STACK POINTER.
    MOVE.L    #EXITMSG,-(A7)  TELL USER EXIT WON'T WORK
    BSR       PRINTF           PRINT IT
    ADDQ.L    #4,A7            FIX THE STACK
    BRA.S     WARMS$           AN ENDLESS LOOP
EXITMSG: BYTE      $0D,$0A,$0A
    BYTE      'An EXIT back to the ROM Monitor is'
    BYTE      ' pretty pointless, you know.'
    BYTE      $0D,$0A
    BYTE      0
* Main routine for the 68010 ROM monitor. Normal entry
* is from a COLD start following reset. Alternate entry
* is from user program via the WARMS (or WARMS1) entry.
* User program may also enter via MONITOR jump vector
* in the ROM jump table. Entering through this point
* permits user program to activate monitor with special
* sequence from within the program, use the monitor as
* if it had been activated by a warm-start, then return
* to user program with an EXIT monitor command. Entry
* by this technique should be with a JSR. Leaving the
* monitor this way will return to the calling program
* with ALL registers preserved, though CCR isn't saved.
MONITOR$: LINK      A6,#BUFFER      TEXT BUFF IN STACK
    MOVEM.L   D0-D7/A0-A7,-(A7)    SAVE EVERYTHING
LOOP:    BSR       PROMPT
    LEA       BUFFER(A6),A0    POINT A0 AT TEXT STRING
    BSR       GETLINE
    BSR       PROCESS$
    BRA.S     LOOP

* NOTE: Exit from this routine is via the monitor EXIT
* command, whose code follows in case structure below
* and is labled EXIT$.
* PROCESS - This is the core of the ROM monitor. It
* executes the command found in the first two character
* positions of line pointed at by A0. Note that this
* routine is available externally via jump table. This
* permits user to load a string with a monitor command,
* put the string's address in A0 and JSR here so the
* monitor can process the command. As is customary, any
* string submitted to PROCESS must be terminated with a
* null byte. Also, any monitor command must be two
* characters long and must begin in column one.

PROCESS$: CMP.B     #0,(A0)          ANYTHING ON LINE?
    BEQ.S     PROCX            BRANCH IF NOT
    MOVE.L    A0,A1            YES, MOVE POINTER INTO A1
    BSR       UPPER            CONVERT TO UPPERCASE
    MOVE.W    (A1),D0          GET THE FIRST TWO CHARACTERS
    MOVE.L    #PROC_TBL,A0     GET ADDR OF PROCESS TABLE
    BRA       CASE             DO THE COMMAND
PROCX:   RTS                        RETURN TO MAIN LOOP
* JUMP TABLE FOR THE CASE SWITCH ABOVE. THIS TABLE IS
* NEEDED SO THE LINKER WILL PROPERLY RESOLVE ADDRESSES
* IN THE CASE SWITCH TABLE.
DUMPMEM$: BSR     DUMPMEM
    BRA.S   PROCX
SWEEP$:  BSR     SWEEP
    BRA.S   PROCX
FILL$:   BSR     FILL
    BRA.S   PROCX
CHANGE$: BSR     CHANGE
    BRA.S   PROCX
S19$:    BSR     S19
    BRA.S   PROCX
GO$:     BSR     GO
    BRA.S   PROCX
FDBOOT$: BSR     FDBOOT
    BRA.S   PROCX            RET ONLY OCCURS ON ERROR
EXIT$:   ADDQ.L  #4,A7        POP THE PROCESS RTN ADDR
    MOVEM.L    (A7)+,D0-D7/A0-A7   RESTORE EVERYTHING
    UNLK      A6             REMOVE THE FRAME
    RTS                      LEAVE THE MONITOR ROUTINE
SELPORT$: BSR     SELPORT
    BRA.S   PROCX
FLOPPYRD$: BSR     FLOPPYRD
    BRA.S   PROCX
FLOPPYWR$: BSR     FLOPPYWR
    BRA.S   PROCX
* WHAT - This is the default CASE arm. It just tells
* the user that the input was not too good.
WHAT:    MOVE.L    #WHAT_MSG,-(A7)
    BSR       PRINTF
    ADDQ.L    #4,A7
    BRA.S     PROCX
WHAT_MSG: BYTE      $0D,$0A
    BYTE      '  Beats me what you want.  Try again.'
    BYTE      $0D,$0A
    BYTE      '  (If you are really stuck, type HELP)'
    BYTE      0
SIZE:    BYTE      $0D,$0A
    BYTE      '  RAM: %D bytes. 1st non-RAM addr: %X.'
    BYTE      $0D,$0A
    BYTE      0
PROC_TBL: WORD      12
    BYTE      'D','U'          DUMP COMMAND
    LONG      DUMPMEM$
    BYTE      'S','W'          SWEEP COMMAND
    LONG      SWEEP$
    BYTE      'F','I'          FILL COMMAND
    LONG      FILL$
    BYTE      'C','H'          CHANGE COMMAND
    LONG      CHANGE$
    BYTE      'H','E'          HELP COMMAND
    LONG      HELP$
    BYTE      'R','L'          RAM LOAD (S19) COMMAND
    LONG      S19$
    BYTE      'G','O'          GO COMMAND
    LONG      GO$
    BYTE      'E','X'          EXIT COMMAND
    LONG      EXIT$
    BYTE      'S','P'          SELECT PORT COMMAND
    LONG      SELPORT$
    BYTE      'F','R'          READ FROM FLOPPY DISC
    LONG      FLOPPYRD$
    BYTE      'F','W'          WRITE TO FLOPPY DISC
    LONG      FLOPPYWR$
    BYTE      'F','B'          BOOT DOS FROM FLOPPY DISC
    LONG      FDBOOT$
    LONG      WHAT             DEFAULT CASE
HELLO:   BYTE      $0D,$0A,$0A
    BYTE      '  68010 ROM MONITOR     V2.1'
    BYTE      $0D,$0A
    BYTE      ' Written for Convergent Technologies'
    BYTE      ' Mini-Frame'
    BYTE      $0D,$0A
    BYTE      0
FIRST:   BYTE      $0D,$0A,$0A
    BYTE      'All commands are two chars, followed '
    BYTE      'by any arguments.'
    BYTE      $0D,$0A
    BYTE      'Separate all args (in hex) by at least '
    BYTE      'one space.  Available'
    BYTE      $0D,$0A
    BYTE      'commands are:'
    BYTE      $0D,$0A,$0A
    BYTE  'Dump memory    DU  <addr>'
    BYTE  $0D,$0A
    BYTE  'Sweep memory   SW  <start> <stop> [times]'
    BYTE  $0D,$0A
    BYTE  'Change memory  CH <addr> <data>...<data> or'
    BYTE  $0D,$0A
    BYTE  '               CH  <addr>'
    BYTE  $0D,$0A
    BYTE  'Fill memory    FI  <start> <stop> <data>'
    BYTE  $0D,$0A
    BYTE  'RAM Load (S1-S9)   RL  <offset addr>'
    BYTE  $0D,$0A
    BYTE  'Goto location  GO  <transfer addr>'
    BYTE  $0D,$0A
    BYTE  'Exit monitor   EX'
    BYTE  $0D,$0A
    BYTE  'Select user port   SP  <channel num>'
```

Continued on page 49

control to the calling program, that program must pop from the stack all values that it pushed before the call to PRINTF. For an example of using PRINTF, see the code just before the label HELP$ in the MONITOR.ASM module.

CASE provides a table-driven solution to the multiple branches found in so many programs, this monitor included. The switch value for CASE passes as a WORD in register D0; the address of the switch table passes as a LONG in register A0.

Transfer out of CASE is at the same level as the entry. If you do a JMP to CASE, your program will behave as if you had done a JMP to whatever location CASE switches you to. If you do a BSR to CASE, each routine in your switch table should end with an RTS, to return control to your program properly.

The Monitor Commands

The monitor processes commands using a large switch table and a call to CASE. Adding commands involves nothing more than linking the new code to the monitor and changing both the help screen and the switch table. Therefore, all monitor commands are written as finished, standalone subroutines.

Some of these routines are general purpose; almost every monitor written has some version of these modules in them. Others are specific to this monitor, as the Mini-Frame needed them to help get from the "Gee, what a neat board" stage to the "Gee, what a neat system" stage.

The SWEEP command does successive reads of all addresses in a given range. The number of reads can be specified on the command line, or the read cycle can be set to continue until you press a key.

I wrote SWEEP to help me track down the I/O devices and control locations in the Mini-Frame system. Generally, I held a logic probe on the chip select pin of whatever device interested me, then used SWEEP to check a range of addresses. When I hit the right range, I could see pulses on the logic probe as the chip select line toggled. Now that the Mini-Frame system is almost fully operational, I seldom use SWEEP.

FILL, CHANGE, DUMPMEM and GO provide typical monitor functions. DUMPMEM is kind of interesting — it's a rather extreme use of PRINTF. Without the PRINTF function, DUMPMEM would have been a pain; using PRINTF made this module much easier to write.

S19 lets the user import Motorola S1-S9 binary records directly through the serial port. Before I figured out how the floppy disk system worked on the Mini-Frame, this was the only means I had for loading executable code.

I declined to add any error detection to this loader, falling back on the standard excuse, "It's such a short RS-232 cable, and it's shielded; there probably won't be any problems!" I couldn't decide what to do if I did get an error anyway, so I just let it go.

About The LINK Instruction

The LINK instruction appears often in the monitor code. It permits a routine to reserve a large block of RAM within the stack for use as local storage. Upon exit, the routine can release that RAM, wiping out the local variables and resetting the stack pointer. The code at label MONITOR$ in module MONITOR.ASM provides a good example of using the LINK instruction and the benefits it provides.

Upon entering the MONITOR subroutine, the LINK instruction forms a secondary stack pointer in A6 by first pushing the value of A6 onto the machine stack (A7), then copying the new value of A7 into A6. Finally, it adds the value of the argument to A7, leaving a gap in the stack exactly (in this case) BUFFER bytes long. This leaves a block of bytes in the stack area that can be directly accessed by the subroutine (relative to A6).

Because the actual stack pointer A7 has moved below the reserved area, that area cannot be clobbered by interrupt service routines. The UNLK instruction at the end of the subroutine undoes all the above, leaving A6 with its original value.

I use the LINK instruction heavily to provide local variables. The monitor only uses about a dozen bytes of RAM that must occupy fixed addresses; with only a little additional effort, I could have eliminated those.

I/O Redirection

Given the two serial ports on the Mini-Frame board, I wanted to be able to run the monitor from either port. To do this, the monitor must be able to switch the character-in, character-out and key-pressed routines between channels. The technique I used involves RAM vectors at fixed locations. Refer to Figure 3 (KEYBOARD.ASM) for details.

The three RAM vectors reside in low RAM beginning at address $09EA. The module KEYBOARD.ASM just checks for a keypress and gets a character from the serial port.

When a program needs to input a character from the serial port, it calls the ROM vector GETC via a JSR instruction. Control passes to routine GETC$. This routine checks the contents of the RAM vector to make sure it contains a JMP instruction, originally left there by the monitor when it started up.

If no JMP instruction exists (usually because a program being tested has run away), GETC$ writes into the RAM vector a JMP to GETC0$, which handles character input for serial port 0.

GETC$ then JMPs to the RAM vector address, which in turn JMPs to the routine that will service the character input request. That service routine should return control to the calling program by executing an RTS instruction.

The RAM vectors in the I/O path gave me great flexibility for the monitor and for any programs that use the monitor's I/O routines. Using these vectors, I can run SK*DOS from either serial port.

So, supporting multiple users requires very little code. All you need is a small supervisor program to start up two versions of the monitor and to switch rapidly between the two serial ports (using the RAM vectors). It felt very strange the first time I fired up my monitor on two serial ports simultaneously; each port functioned perfectly, even when loading S1-S9 records at 9600 baud.

In Conclusion...

This has been a rough overview of the monitor; there is simply too much code here to go into great detail. I will gladly answer any questions readers might have; you can reach me via the Micro C BBS. The code modules are available for downloading and on the Micro C Issue #47 disk.

I place these routines in the public domain. The PRINTF and CASE modules were already public domain, thanks to authors Donald Krantz and James Stanley.

One final comment: This monitor represents my first programming effort in the 68000 assembly language. Therefore, some of the code will appear crude or inelegant to anyone skilled with the processor. As I looked over the modules while writing this article, I could already see some spots that could use a bit of smoothing.

But I didn't; these modules contain the code presently burned into my EPROMs. I invite any reader who wants to hammer on this stuff and clean it up or improve it to do so, and I hope he will share the results with us.

♦ ♦ ♦

Continued from page 47

```
    BYTE  $0D,$0A
    BYTE  'Read from flppy FR <start> <end> <log sctr>'
    BYTE  $0D,$0A
    BYTE  'Write to floppy FW <start> <end> <log sctr>'
    BYTE  $0D,$0A
    BYTE  'Boot from floppy  FB'
    BYTE  $0D,$0A
    BYTE  $0A
    BYTE  0
```

♦ ♦ ♦

Figure 3 — Keyboard Services

```
* Low-level routines for getting characters and lines
* from the keyboard. A line is (usually) a byte-array
* string of characters terminated by a null, suitable
* for use with the PRINTF routine.
    PUBLIC    GETC0$,GETC1$,KEYPRSD0$,KEYPRSD1$
    PUBLIC    GETC$,KEYPRSD$,GETLINE$
    EXTERNAL  PUTC,PRINTF,GETC
    EXTERNAL  GETCRAM,KEYPRSDRAM
    EXTERNAL  GETC0,KEYPRSD0
BASE    EQU   $C30000
* GETC - This routine simply checks the RAM link for a
* JMP.L instruction and, if found, jumps to it. The
* monitor normally writes a JMP to the ROM-based GETC0
* routine, to use RS-232 channel 0 for initial comm. If
* the JMP isn't found, the link is reset to channel 0
* before the JMP is made.
GETC$:   MOVEM.L   D0/A0,-(A7)    SAVE SOME REGS
    MOVE.L    #GETCRAM,A0     GET THE LINK ADDR
    CMP.W     #JUMP,(A0)      IS THERE A JMP THERE?
    BEQ       GETCX           BRANCH IF YES
    MOVE.W    #JUMP,(A0)+     NO, WRITE A JMP
    MOVE.L    #GETC0,(A0)     WRITE THE GETC0 ADDR
GETCX:   MOVEM.L   (A7)+,D0/A0    RESTORE THE REGS
    MOVE.L    #GETCRAM,-(A7) PUSH LINK ADDR
    RTS                       AND GO THERE
JUMP:   EQU   $4EF9           JMP.L INSTRUCTION
* GETCn - Primitive routines to get one character from
* the keyboard. Character is returned in the low byte
* of D0, with the remaining bits of D0 set to 0. All
* other registers are preserved. GETC0 accesses only
* RS-232 channel 0; GETC1 accesses channel 1. Actual
* routine used at execution time varies, depending on
* the link written into monitor RAM location GETCRAM.
GETC0$:  MOVEM.L   A0,-(A7)       SAVE A0
    MOVE.L    #BASE,A0        POINT AT CHNL 0 DATA PORT
    BRA       GETC_$          GO TO SERIAL HANDLER CODE
GETC1$:  MOVEM.L   A0,-(A7)       SAVE A0
    MOVE.L    #BASE,A0        POINT AT CHNL 0 DATA PORT
    ADDQ.L    #2,A0           POINT AT CHNL 1 DATA PORT
GETC_$:
LOOP0:   MOVE.W    4(A0),D0       NOW GET THE STATUS
    AND.L     #1,D0           CHECK FOR CHAR AVAILABLE
    BEQ       LOOP0           LOOP UNTIL CHAR IS THERE
    MOVE.W    #1,4(A0)        SELECT REG 1
    MOVE.W    4(A0),D0        GET THE RCV STATUS
    AND.L     #$70,D0         LEAVE ERROR BITS
    BEQ       GOOD0           BRANCH IF NO ERROR
    MOVE.W    (A0),D0         GET CHAR (TO CLEAR CHIP)
    MOVE.W    #$30,4(A0)      RESET THE CHIP
    CLR.L     D0              RETURN A NULL
    BRA       GETCX0
GOOD0:   MOVE.W    (A0),D0        GET THE CHAR
    AND.L     #$00000FF,D0    LEAVE ONLY LOW BYTE
GETCX0:  MOVEM.L   (A7)+,A0       RESTORE A0
    RTS
* GETLINE - This routine gets a line of characters
* (terminated by CR) from the keyboard. The characters
* are stored in a buffer whose address is passed at
* invocation in register A0. All registers are returned
* unchanged. The line returned in the buffer is always
* terminated by a null and the CR entered at the keybrd
* is NOT included in the line.
* Characters of significance:
* CR ($0D) never echoed, but appears in buffer as null
* CAN ($18 or ^X) deletes all previously entered chars.
* It causes a CRLF sequence to go to the terminal.
* HT ($08) and DEL ($5F) both act as delete keys,
* causing the the previous character to be erased.
GETLINE$: MOVEM.L   D0/D1,-(A7)
GETL0:    CLR.L     D1          D1 INDEXES INTO THE STRING
    CLR.B     0(A0,D1.W)      MARK BUFFER WITH NULL
GETL1:    BSR       GETC        GET A CHAR FROM KEYBOARD
    TST.B     D0              WAS IT NULL?
    BEQ       GETL1           IGNORE IF YES
    CMP.B     #$0D,D0         WAS IT CR?
    BEQ       GETL_EXIT       EXIT IF SO
    CMP.B     #$08,D0         WAS IT BACKSPACE?
    BEQ       BKSPC           BRANCH IF YES
    CMP.B     #$5F,D0         WAS IT DELETE?
    BEQ       BKSPC           BRANCH IF YES
    CMP.B     #$18,D0         WAS IT CANCEL (^X)?
    BEQ       CNTLX           BRANCH IF YES
    MOVE.W    D0,-(A7)        NONE OF ABOVE,PUT ON STACK
    BSR       PUTC            SEND BACK TO CRT
    ADDQ.L    #2,A7           TRASH THE STACK
    MOVE.B    D0,0(A0,D1.W)   MOVE CHAR INTO STRING
    ADDQ.W    #1,D1           COUNT THIS CHAR
    CLR.B     0(A0,D1.W)      MARK BUFFER WITH NULL
    CMP.W     #255,D1         USED ALL 255 CHARS YET?
    BEQ       GETL_EXIT       BRANCH IF YES
    BNE       GETL1           NO, KEEP GOING
BKSPC:   TST.L     D1          ANY CHARACTERS TO DELETE?
    BEQ       GETL1           BRANCH IF NOT
    SUBQ.L    #1,D1           DROP COUNT BY 1
    CLR.B     0(A0,D1.W)      ZERO OUT THIS CHAR IN BUFF
    MOVE.L    #BACKUP,-(A7)   GET THE BACKUP SEQUENCE
    BSR       PRINTF          ERASE THE CHARACTER ON CRT
    ADDQ.L    #4,A7           REMOVE THE MESSAGE ADDR
    BRA       GETL1           GET MORE CHARACTERS
CNTLX:   MOVE.L    #CANCEL,-(A7)  GET THE CANCEL TEXT
    BSR       PRINTF          SEND TO SCREEN
    ADDQ.L    #4,A7           RESET THE STACK
    BRA       GETL0           START OVER AGAIN
BACKUP:  BYTE      $08,$20,$08    BKSPC, SPACE, BKSPC
    BYTE      0
CANCEL:  BYTE      '^X'
    BYTE      $0D,$0A
    BYTE      0
GETL_EXIT: MOVEM.L   (A7)+,D0/D1
    RTS
* KEYPRSD-This routine checks RAM link for legal JMPs.
* If found, a transfer through the link to the
* keypressed routine. If a JMP isn't found, the link is
* assumed corrupted and transfer to KEYPRSD0 is set up.
KEYPRSD$: MOVEM.L   D0/A0,-(A7)    SAVE SOME REGS
    MOVE.L    #KEYPRSDRAM,A0 GET THE LINK ADDR
    CMP.W     #JUMP,(A0)      IS THERE A JMP THERE?
    BEQ       KEYPRSX         BRANCH IF YES
    MOVE.W    #JUMP,(A0)+     NO, WRITE A JMP
    MOVE.L    #KEYPRSD0,(A0) NOW WRITE THE GETC0 ADDR
KEYPRSX: MOVEM.L   (A7)+,D0/A0    RESTORE THE REGS
    MOVE.L    #KEYPRSDRAM,-(A7)  PUSH LINK ADDR
    RTS                       AND GO THERE
* KEYPRSDn - This routine glances at the serial port
* (channel A) and returns the state of the Rcv Char Av
* bit in the Carry bit of the CCR. All registers are
* saved. To use this routine, a program should first
* BSR to KEYPRSD, then BCC or BCS. The BCC will branch
* if no char is available, while the BCS will branch if
* a character is available.
KEYPRSD0$: MOVEM.L   A0/D0,-(A7)    SAVE SOME REGS
    MOVE.L    #BASE,A0        GET THE PORT'S ADDR
    MOVE.W    4(A0),D0        GET THE CHIP'S STATUS
    ROR.W     #1,D0           CHAR AVAIL BIT INTO CARRY
    MOVEM.L   (A7)+,A0/D0     RESTORE THE REGS
    RTS
* Same again, for channel 1.
KEYPRSD1$: MOVEM.L   A0/D0,-(A7)    SAVE SOME REGS
    MOVE.L    #BASE,A0        GET THE PORT'S ADDR
    MOVE.W    6(A0),D0        GET THE CHIP'S STATUS
    ROR.W     #1,D0           CHAR AVAIL BIT INTO CARRY
    MOVEM.L   (A7)+,A0/D0     RESTORE THE REGS
    RTS
```

♦ ♦ ♦

Of Awks And Addresses

By Scott Robert Ladd
302 N. 12th Street
Gunnison, CO 81230
(303) 641-6438

Scott moves to the coldest place on earth and then starts off this column Awk-wardly. I wonder if his frozen plumbing had anything to do with it.

Since I last put fingers to keyboard for this column, the universe has taken a pleasant turn for the better. As you can see from the address in the masthead, I have changed my domicile.

Gunnison is a town of 6,500 people (mostly) located deep in the Colorado Rockies. While listed as the "Coldest Place in the Nation," it's also in the middle of some of the most beautiful countryside. The Black Canyon of the Gunnison, Taylor and Blue Mesa Reservoirs, and several National Forests are within a few minutes walk or drive. The change from my former residence in the polluted, overpopulated Denver basin is amazing.

We may even hold a "Rocky Mountain SOG" early next summer here. But more on that later ...

This also gives me an opportunity to ask for feedback from the readers of this fine magazine. Fire off those letters with your opinions and suggestions regarding this column. At best, I'll read them; at worst, they'll get used as Vogon fire-starters.

Toolbox

The cold temperatures and snow made me think of the Arctic. The arctic, in turn, is full of interesting creatures, such as seal, walri (walruses?), polar bears, and odd-looking sea-birds (this is leading somewhere, I hope). The sea birds include penguins and auks. An Auk is a black-and-white diving seabird with a big beak. An Awk (with the "w") is a C-like programming language common to most UNIX systems. (See? I was right.)

Awk is named for its creators, Alfred Aho, Peter Weinberger, and Brian Kernighan (who put the "K" in K&R). Their book, *THE Awk PROGRAMMING LANGUAGE* (known as "the Awk book"), defines the language. Awk first appeared on AT&T UNIX systems in the late 1970s.

What is Awk useful for? Well, it can process text files as a series of strings and tokens. It can create utilities and report programs. Awk is also capable of more complex tasks — including much of what people use C for.

Awk is actually an interpreter. You can run simple one-line Awk programs on the command line, or longer programs from a text file. An Awk program consists of a series of one or more pattern/action statements. A pattern/action statement has the following form:

```
pattern { action(s) }
```

As in C, multiple action statements can be separated by semicolons. A pattern is a conditional statement that's compared to each line in the input file. If the pattern is true, the actions associated with that pattern get executed. Patterns can consist of simple comparisons or regular expressions (such as those used by the UNIX utility grep).

Awk breaks the line of input into text tokens based on a default (or user-definable) set of delimiters. Normally, the delimiters are space, tab, and newline. Let's see how Awk works on a simple text file. MICROC.DAT contains a list of the people who work for an obscure computer publication:

```
Dave Thompson      Ed      Bend    OR
Larry Fogg         Tech    Bend    OR
Gary Entsminger    Ed      Davis    CA
Scott Ladd         Col     Gunnison CO
Laine Stump        Col     Lake_City MN
```

Running this awk program:

```
$3 == "Ed" {print $1, $2, $5}
```

on MICROC.DAT will display:

```
Dave Thompson OR
Gary Entsminger CA
```

The program says: "For every input line in which the third field equals the text 'Ed,' print out fields one, two, and five." See how simple Awk is to use?

Awk programs can be both powerful and

short. The following (much shorter than its C equivalent) counts the number of lines, words, and characters in an input file:

```
BEGIN {print "Program: Line, word,
    and character counter" }
   {nowords += NF;
    nochars += length($0) + 2 }
END  {print NR,"lines,", nowords,
   "words, ", nochars,
   "characters"}
```

BEGIN and END are special patterns. All the actions specified for the BEGIN pattern take place before reading any program lines; actions associated with the END pattern take place after reading the entire input file.

The built-in variables NF and NR represent (respectively) the number of fields in the current input line, and the total number of lines (records) read from the input file. Length() is a function similar to C's strlen(); it returns the length of the given token. The special token $0 represents the entire, unbroken input line.

As you can see, Awk programs can have variables. Variables contain either numeric or string values, and Awk automatically initializes them to 0. Numeric values can be floating-point or integer data; conversions happen automatically. The language has many built-in standard C-library functions, such as printf(). And, you can also write your own procedures.

The Awk book contains hundreds of example programs. The scope of the examples indicates how useful Awk can be. Included in the book are a pair of calculators, a cornucopia of sorting programs and procedures, a simple make utility, and an assembler/interpreter! All this from a language which "merely" processes text files. This is a real language.

Space limitations prevent me from giving you a complete tutorial but if you want to learn more about it, check out one of the versions for MS-DOS listed below. The Awk book describes the language as it exists on most UNIX systems. I know of two commercial Awks, one freeware Awk and several subsets in the public domain.

MKS

Mortice Kern Systems (MKS) produces a series of UNIX-like tools for MS-DOS. Their MKS Toolkit contains dozens of UNIX commands, and can be used as an alternative to MS-DOS's COMMAND.COM.

Standard C provides a semi-portable function for handling some asynchronous events.

The Toolkit includes MKS Awk, or you can purchase it separately. The interface to MKS Awk is similar to that of the UNIX version. The documentation is probably this Awk's biggest weakness; while the manual tells you how to run the interpreter, it says very little about the language.

Polytron

Polytron, makers of several programmer's tools, puts out a very good Awk. This product comes with a copy of the Aho/Weinberger/Kernighan book as its primary documentation. A file on the distribution disk contains information on differences between PolyAwk and UNIX Awk. PolyAwk includes several extensions to UNIX Awk, and has increased the size of the built-in function library substantially.

Awk-Ware

Finally, Rob Duff of Vancouver (Canada) has produced a version of Awk released as Freeware. Freeware is copyrighted software released by the author for free distribution. The only restriction on its distribution is that the author forbids the selling of the product for profit.

This complete Awk comes with excellent documentation. It, like the Awks above, includes MS-DOS extensions. (Rob Duff also puts out PC versions of YACC and LEX, two programs designed for constructing compilers.)

I have posted the archive Awk210A.ARC, which contains Rob's latest Awk, on the Micro C BBS. I strongly urge that you take a look at it. Rob Duff can be reached at FidoNET address 1:153/713. His BBS phone number is (604) 251-1816.

Awk is a comprehensive and powerful language, easy and fun to work with. Experienced C programmers should have this utility in their toolboxes.

Figure 1 — Control-C Control

```
#include "signal.h"
#include "stdio.h"
#include "stdlib.h"

void main(void);
void handler(int sig);

void main()
    {
    int i;

    signal(SIGTERM,handler);

    for (i = 0; i < 100; ++i)
      puts("signal()test program");
    }

void handler(int sig)
    {
    puts("\aCTRL-C pressed!");
    exit(1);
    }
```

♦ ♦ ♦

C Explorations

Well... I was going to be talking about interrupt handlers in C. That was until I read the article by Sam Azer on that very subject in the March-April issue of this magazine! Sigh... I hate it when someone steals my thunder. Sam, of course, has left me with only two alternatives: to send a pack of ravenous auks after him, or come up with a new subject. Since I'm fresh out of auks, I guess I'll talk about something else.

Signaling

Well, maybe I *can* slip something in on interrupts. Standard C provides a semi-portable function for handling some asynchronous events. Called signal(), it's probably one of the more forgotten and misunderstood functions in the C library.

The prototype for signal looks like this:

```
void (*signal(int sig,
      void (*func)(int)))(int);
```

This is the most complicated prototype in the ANSI standard. In English, signal accepts two parameters: an integer and a function pointer. The function used in the parameter must accept one integer parameter and return nothing. If successful, signal() returns the value of the function pointer parameter. Otherwise, it returns the value defined by the constant SIG_ERR.

While this seems complex, it isn't. Figure 1 shows an example program which uses signal() to control the

CTRL+C interrupt. As you can see, the call to signal() is quite simple.

signal() sets up a handler function for a specific event. In our example, the call to signal() states that when it receives the SIGABRT (CTRL+C) signal, the function ctrl_c_handler() should be called. In effect, we've created a portable program that traps the CTRL+C interrupt.

There are flies in this ointment, however. The ANSI standard is somewhat nebulous about which events cause each standard signal. There are six standard signals:

- SIGABRT — abnormal program termination
- SIGFPE — an erroneous mathematical operation (e.g., divide by zero)
- SIGILL — illegal processor instruction (or your program is sick)
- SIGINT — received an interactive attention signal
- SIGSEGV — the program performed an illegal storage access
- SIGTERM — received a program termination message

Microsoft and Borland think the CTRL+C interrupt should be mapped to the SIGABRT signal. Zortech C and C++ think that it should be assigned to SIGTERM. I think Microsoft and Borland have it right, but the ANSI standard certainly leaves both options open. If you want portability, be aware of the differences.

Many of the standard signals do not occur on PCs, and some additional signals were added. There is no interrupt trap on Intel microprocessors for illegal opcodes, for instance. Microsoft and Borland have expanded the definition of the SIGFPE (floating-point error) signal to handle several advanced problems.

I should mention another function associated with signals. It is raise(), which has the prototype:

```
int raise(int sig);
```

Call raise() with one of the signal constants as an argument. The function associated with that signal is then called. I'm not sure I see a real purpose for this function, except possibly in multi-threaded programs such as those for OS/2.

Pointers On Pointers

C seems to have this sign attached, saying "Watch out — slippery pointers ahead!" Pointers are both a boon and a bane of C. While powerful, they're also dangerous.

If you don't need pointers on pointers, you can skip the next few paragraphs. They introduce the fundamentals of working with pointers.

Pointer Basics

A pointer is nothing more than a variable which contains a memory location. You have complete control over what gets put in that location. Pointers can be initialized in one of three ways: with the address of an existing variable, with an address to a block of storage on the heap, or with the location of a piece of hardware.

The first method is rather easy. In Figure 2, the pointer p is initialized to the value of the integer i.

Figure 2 — Pointer Initialization

```
#include "stdio.h"

main()
     {
     int i = 0;
     int * p;

     /* set p to the address of i*/
     p = &i;
     /*asgn 42 to int pntd to by p*/
     *p = 42;
     /* print the value of i */
     printf("i = %d",i);
     }

                ♦ ♦ ♦
```

This program prints: i = 42. The statement *p = 42 assigns 42 to the integer at the address pointed to by p. Since p was set to the address of i, 42 is assigned to i. Another term for this is indirection; the value of i is being modified indirectly, through the pointer p. The indirection operator, *, says that we are referring to the item located at the address stored in a pointer.

All C programs use a block of memory called a heap. There is nothing particularly special about the heap; the program uses this block of memory for data storage at run-time. The malloc() family of functions allocates and deallocates space on the heap. Items allocated on the heap are referenced via pointers. Figure 3 shows how this works.

p points to an integer. The call to malloc() creates space for an integer on the heap and returns the address of the space. That address is assigned to p. When we add the indirection operator "*" to p (i.e., *p), we tell the compiler we want to access the contents of the memory location addressed by p.

Since the heap contains a finite amount of space, you can make the space used by the integer available again by using the free() function.

Allocating and freeing memory on the heap is known as dynamic memory allocation. The compiler allocates static and automatic variables with a fixed size (in the data segment and stack segment, respectively). Dynamic memory allocation takes place at run-time.

As with most programming techniques, dynamic allocation has some tradeoffs. While it is slower than static allocation, dynamic allocation can really reduce memory requirements.

An excellent example of the uses of dynamic memory allocation can be seen in the program FXREF.C (see *Micro C*

Figure 3 — Use Of The Heap

```
#include "stdlib.h"
#include "stdio.h"

main()
   {
   int *p;

   /*allocate memory for an int */
   /*assign the address to p */
   p = malloc(sizeof(int));
   /*asgn 23 to int just created*/
   *p = 23;
   /* print out its value */
   printf("heap int = %d",*p);
   /* deallocate (free) the int */
   free(p);
   }

              ♦ ♦ ♦
```

#46 for source), which I use as a standard compiler benchmark. FXREF reads a file from disk and builds a binary tree containing text tokens and line-number references. There is no way to know how big the binary tree will need to be before reading the entire file.

While we could create a binary tree structure statically, the design would need to handle the largest possible tree. Not only does that waste memory (since almost all trees will be smaller than the maximum), but it's possible that our estimate of the largest possible tree could be too small.

Finally, pointers can be set to point to physical memory locations. Some compilers, like Turbo C and Zortech, have a macro (MK_FP) which can set a pointer to a specific memory address. This is tricky and *very* nonportable; it is highly compiler and hardware dependent.

A program can crash in many ways using pointers. As you become more familiar with C, you'll no doubt find

many new and subtle ways to send your system to never-never land. Here are a few to get you started.

Inconsistent Pointer Types

If you assign a float value to a location you've declared an integer, you'll tromp on something. Many compilers will warn you about this problem, but not all.

Uninitialized Pointers

You have to initialize the little beasties or you get potluck. An uninitialized pointer can point almost anywhere in memory. Storing values at these random locations can overwrite parts of the operating system, the program, or your mother's phone number.

For instance, in memory models which use 32-bit data pointers, the compiler automatically initializes the pointer to address 0:0. That's the location of the MS-DOS interrupt vector table and writing to that area will crash your PC faster than a power failure.

Overrunning

C does no bounds checking of any kind. If you want to access the 11th element of a 10 element array, C will quite happily let you do this. Unfortunately, this will also overwrite whatever's there.

This is a good place to mention one of C's most notorious (and subtle) evil functions: gets(). gets(char *buffer) reads data from standard input, and stores it in the area pointed to by its char pointer parameter, buffer. The problem is that it does not do any checking to see if the input is longer than the space pointed to by buffer. This is deadly.

The recent "computer virus" used this very feature of gets() to install its own code into a function, which then did its dirty work. Never use gets(). Always use fgets(), which lets you specify the maximum length of the input data.

Freeing Pointers

Of course, if you use memory dynamically you have to free() it when you're finished.

This is one of those benign oversights which does not crash the program until the heap runs out of space. A common cause of this comes when allocating a char pointer via the strdup() function, but forgetting to free() it later.

Of course you probably shouldn't free just anything. Using free() on a pointer to static data or a pointer argu-

CITIZEN MATE/12 286 SYSTEM

80286 With 12.5 MHz Clock Speed
has on the Mother Board:
ONE Meg RAM with 1 Wait State
Video Controller Supports EEGA, EGA
CGA, MGA, Hercules and
Plantronics Color Plus
Controller Provides Support for
Two Hard Drives and Two Floppy
Drives, 5.25 and 3.5 Capability
Mouse, Parallel and Two Serial Ports

1.2 Meg Floppy Installed
32k Hard Drive Cache Installed
101 Enhanced Keyboard
MS-DOS 3.3 With GWBASIC
Small Footprint
Standard 1MB Expandable to 4MB
Novelle Compatible
Nation Wide Service

**********$1595.00

XT CLONE SYSTEMS
PLEASE CALL FOR CURRENT PRICE

HARD DRIVES FOR XT AND AT	
ST-225 KIT FOR XT (20 MEG)	$ 299.00
ST-238 KIT FOR XT (RLL 30 MEG)	$ 319.00
ST-251 FOR AT (40 MEG)	$ 359.00
MONITORS	
Color Monitor RGB (CGA)	$ 255.00
Color Monitor RGB (EGA)	$ 375.00
Monochrome TTL (Green)	$ 85.00
Monochrome TTL (Amber)	$ 95.00
EGA Color Video Card	$ 159.00

CITIZEN PRINTERS

MODEL 120D	120 CPS 9"	$ 165.00
MODEL 180D	180 CPS 9"	$ 185.00
MODEL MSP-15E	160 CPS 15"	$ 359.00
MODEL MSP-40	240 CPS 9"	$ 339.00
MODEL MSP-45	240 CPS 15"	$ 449.00
MODEL MSP-50	300 CPS 9"	$ 399.00
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%
MN Add 6% Sales Tax Subject to change

Reader Service Number 15

ment can cause unpredictable results. It is also a no-no to free() or use a pointer which has its heap data already freed.

Remember that once you free the data associated with a pointer, the value pointed to isn't protected. Sure the pointer will probably point to the same location and the data may still be there. But the heap no longer protects that memory, and is likely to allocate the space for new data.

Aliasing: Pitfall For Optimizing Compilers

An alias occurs when more than one variable accesses a single memory location. Pointers often run into this problem, as can certain types of functions. Figure 4 contains a common aliasing problem.

Figure 4 — Aliasing

```
void alias_func(int *, int *);

main()
     {
     int a;

     alias_func(&a,&a);
     }

void alias_func(int * i1, int * i2)
     {
     /* do various things */
     }

          ♦ ♦ ♦
```

When calling alias_func(), both parameters are the address of the int a. An optimizing compiler will sometimes have trouble with situations such as this, since it has no way of knowing at compile time that both i1 and i2 point to the same value. Depending on what goes on in alias_func(), the compiler may generate some weird code.

Both Microsoft C and WATCOM C have an option to ignore aliases. This allows the compilers to generate faster code, since they can remove, delete, and combine variables without concern for whether two variables are the same. Unfortunately, in code which contains aliases, these compilers will generate strange programs.

News And Reviews

C Ware DeSmet Personal C Compiler

For several years, there have been MS-DOS C interpreters and compilers available in the public domain or as shareware. Many are based on Small-C, which means they support only a subset of the language and they output macro assembler files. None have been powerful enough for serious developers.

An alias occurs when more than one variable accesses a single memory location.

C Ware has just changed that. They released a version of their DeSmet C (reviewed last issue) as shareware. Called Personal C, it's not only a true compiler, but it's cheap. The shareware registration fee is only $30! Talk about a bargain.

Personal C follows the original K&R standard, with a few ANSI features thrown in. This is your basic fast compiler and documentation.

However, the object modules produced are not compatible with Microsoft's .OBJ format and the package includes only two header files: STDIO.H and MATH.H.

The archive contains the compiler, linker, ramdisk driver, and macro assembler. It also includes several example programs. When you register, C Ware will send you a diskette containing: a profiler, program lister, object module librarian, and extended memory-allocation functions.

For a little bit extra, you can purchase an editor, debugger, and other utilities. Limited telephone support costs $15.

I've posted Personal C on the *Micro C* BBS, under the file name PCC12B.ARC.

Editor's note: You can also find it on Micro C's Issue #47 disk for $6.

While I think Personal C is a wonderful bargain, Power C from MIX (reviewed last issue) sells for only $20. Power C is a more complete package, with an excellent tutorial manual. And for an additional $10, you get Power C's library source and an assembler. However, Personal C still provides a very nice package for a very few bucks.

Personal C $30 Reg. Fee
C Ware Corp.
P.O. Box 428
Paso Robles, CA 93447
(805) 239-4620

Power C $19.95
MIX Software
1132 Commerce Dr.
Richardson, TX 75081
(214) 783-6001

QuickC v2.0

Well, it finally got here in early January. QuickC version 2.0 is a major improvement over its predecessor. This is a real compiler, useful for creating real programs (written by real programmers!). They've corrected almost all the earlier version's problems. However, it still isn't perfect. Some pluses and minuses:

PLUS: Hypertext-like Help. I've mentioned the new help system before. It contains all the information normally found in reference manuals. Information in the help displays, such as example programs, can be cut and pasted into your source programs. An index and table of contents support browsing through the help information.

MINUS: Marginal printed documentation. Microsoft provides a start-up booklet, a reference manual for the command-line utilities, and a printed tutorial called *C For Yourself.*

C For Yourself is one of the best C tutorials I've seen. Unfortunately, it's incomplete. Microsoft figures that programmers will use on-line help to go beyond the tutorial. This is fine for experienced programmers.

PLUS: Fast, fast, fast. Raw compile speeds are the fastest of any MS-DOS C compiler. The linker, oddly enough, has slowed down slightly. However, if you want fast, you use incremental compiles and links, a technology only available in QuickC v2.

When rebuilding a program, only those functions which have changed get recompiled and linked. For large source files, this produces amazing compilation times. The intelligent system avoids rereading header files, too. Once read, a header stays in memory.

PLUS: Complete Memory Model Support. The environment can now handle programs compiled in any memory model — not just Medium model, as in QuickC 1.0x. They've added the Huge model to handle items larger than 64K. The only memory

model missing is Tiny, used to create .COM programs.

PLUS: A Great Debugger. I'm usually disappointed with integrated debuggers; they are often so wimpy they're useless. QuickC 2.0's debugger, though, is a practical tool. Besides the obligatory breakpoints, watch values, and line-by-line stepping modes, it incorporates some new ideas.

Turning on Debugging History allows you to trace what has already occurred. You can even set conditional breakpoints (called watchpoints). When the conditional statement becomes true, the program stops. This will show up in version (3.0) of CodeView.

MINUS: A Stupid Debugger Bug. You can't examine the value of a variable during debugging without placing it into a watch window! I'm told this was an oversight, but I think someone just dropped the ball. It makes no sense to use a dozen keystrokes to load a variable into a watch window (and then remove it), when you only want to see the value.

PLUS: In-line Assembler. Microsoft built a true in-line assembler into the QuickC compiler. Turbo C has in-line assembler, too, but you must own an external assembler in order for it to work.

QuickC does everything internally. QuickC's assembler syntax is also more intelligent than Turbo C's. It lets you define blocks of assembler code rather than prepending each line with "asm." And, the QuickC debugger can handle in-line assembler. It can even display the processor's registers and flags during debugging!

PLUS: Microsoft finally includes their "real" make utility with a product. The old MAKE program was too primitive. NMAKE is the excellent new utility. It's compatible with UNIX make, has conditional lines, and works recursively (let me repeat that). Why they haven't released NMAKE before now is a mystery, since I understand they've used it in-house for years.

Sorta MINUS: The upgrade price is $50! Heck, a new package sells for under $70 through many discount houses. However, this isn't as much of a minus when you realize that the upgrade *is* a completely new package, identical to those sold in the stores. And, you get a massive improvement in performance.

Overall, I rank QuickC as a big winner, with few significant problems. The lack of documentation hurts it the most, but the increased capabilities of the system make up for it.

WATCOM C v7.0...

...will be released in mid-February. The best-optimizing C compiler on the market just got better. I'm still not sure I can believe some of the benchmark results I'm seeing. In one case, WATCOM C's floating-point emulator is faster than some compilers' coprocessor code.

The only downer is that the compile speeds have not improved. WATCOM is still the slowest compiler on earth; it takes nearly an hour to compile a 10,000 line program on my 16 MHz 80386. However, it produces amazingly good code.

C++ Is Still A'coming...

C++ continues to gain momentum, albeit slowly. Microsoft has announced an agreement with Glockenspiel to market a C++ translator jointly. Object libraries are available so that Windows and PM programming can be done in an object-oriented manner. I still think C++ is the C of two to five years from now.

Resources

If you want to attend Micro Assembly, and you haven't registered, this is your last chance! Micro Assembly is a programmer's conference being held May 27-29 in Denver, Colorado, at the Clarion Hotel (Airport). Registration is $35 in advance, $50 at the door. Send your name, address and phone number (and a check or money order, of course!) to: Micro Assembly, 2560 South Hazel Court, Denver, CO 80219. We hope to see you there!

That's All, Folks!

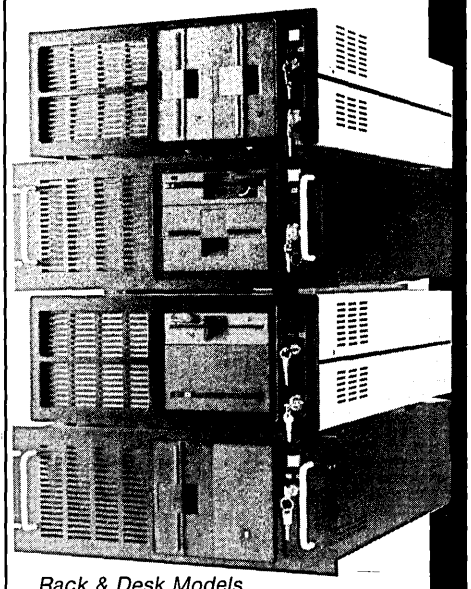
Well, that's all for this month. I won't be making any promises about what I'll cover next issue, because it never works out that way. Perhaps I'll be able to talk about Zortech's new C debugger. Lint may be a big subject next time around; after all, every C programmer needs lint. In the following issues I'll try to make clear the ongoing battle over grit, mud, and sludge: that is if Dave doesn't dive in first.

In the meantime, keep on programmin'...

♦ ♦ ♦

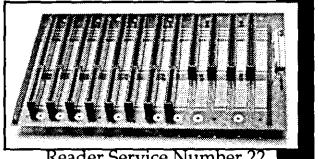
Rack & Desk PC/AT Chassis

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



Rack & Desk Models
Accepts PC, XT, AT Motherboards and Passive Backplanes
Doesn't Look Like IBM
Rugged, Modular Construction
Excellent Air Flow & Cooling
Optional Card Cage Fan
Designed to meet FCC
204 Watt Supply, UL Recognized
145W & 85W also available
Reasonably Priced

Now Available
Passive Backplanes

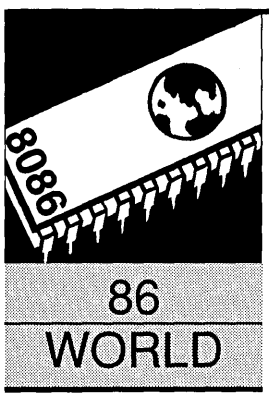


Reader Service Number 22

INTEGRAND RESEARCH CORP.

Call or write for descriptive brochure and prices:
8620 Roosevelt Ave. • Visalia, CA 93291
209/651-1203
TELEX 5106012830 (INTEGRAND UD)
FAX 209/651-1353
We accept Bank Americard/VISA and MasterCard

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



Another Sob Story

By Laine Stump
%Redhouse Press
Merkez PK 142
34432 Sirkeci
Istanbul, Turkey

Laine has an adventure with a hard drive. Seems it ate its DOS partition (causes serious indigestion). He recovers by creating a Rolaids routine in Pascal (or is it C?) to read and write sectors on the drive that "isn't there."

You've come to expect it from me, so there's no sense in trying to disguise it. I guess I'll just tell my story; get it over with. At least this sob story has a happy ending.

Friday Afternoon

It was 5:30 Friday. I had just finished clearing up a long list of problems in the software I'm working on for a small, hand held, 80188-based computer that plugs into a Bobcat front end loader. (PC Tech doesn't just make desktop PCs!) Time to make a backup and head home to catch Patty Duke on the tube.

But for some reason, my system (an X24, 12 MHz AT compatible) had locked up. With the software I write, I'm used to that kind of stuff (some would call it "bad" software; I prefer "daring"). I reached over and hit the reset button, just like I've done thousands of times since I got the first PFM monitor signon on my old Big Board all those years ago.

This time, something was different. I didn't see a "C>." I saw a "DISK BOOT ERROR. INSERT SYSTEM DISK AND PRESS ENTER." Yipes!!! Why couldn't this happen on Monday, instead?

Looking For The Beast

Don't panic yet. Take a deep breath. Think. What could cause this? Where do I start?

First I tried booting with a floppy disk and looking at the drive. "Invalid Drive Specification." That meant the problem wasn't with the system files themselves. Things were so screwed up that DOS didn't even recognize the hard drive.

So I got out Disk Manager and tried running some diagnostics. Not a single error on the entire drive. That only leaves one thing: the partition table must have gotten corrupted somehow. I started up FDISK and asked it to give me a list of current partitions. "No partitions defined." Well, that about sews it up.

Time Out

"Wait!," you scream, "What the heck is a Pertution Tiffle?" Not much really. It just tells DOS where all the data on your disk is. Every bit of it.

You see, DOS doesn't assume that your entire hard disk will be used for DOS. You might have part of it (a "partition") allocated for Xenix, or CP/M, or who knows what else. To keep the operating systems from stepping on each other's sectors, an area on every PC's hard disk is reserved to tell everybody "who" is in control of "what."

The partition table is stored on the very first sector of a hard disk (cylinder 0, head 0, sector 1). What it really tells DOS is where on the disk a "logical disk" starts, and how big this logical disk is. The standard allows for up to four partitions on a single disk.

In older versions of DOS, only one of those partitions could be a DOS partition, and that partition could be no larger than 32 megabytes. DOS 3.3 was the first to allow multiple 32 MB DOS partitions, and DOS 4.0 lets the partitions be larger than 32 MB (57 jigabytes or something).

But, as usual, I digress. Let's look back on our poor overworked hero, who's still struggling with his drive that has 0 megabytes. More about the structure of the partition table in a later episode. We don't care for now what it looks like. All we care about is that my drive doesn't have one.

What To Do?

I thought of just telling FDISK to repartition the drive as it was partitioned originally (a single 40 MB partition). I wasn't sure if FDISK rewrote the directory or the file allocation table, though (more about those in a sequel, too). I called Earl at home and dragged him away from the CD player long enough for him to tell me he wasn't sure, either.

Well, I was afraid to rewrite the partition table with FDISK. What if my data was still

there and I ruined it by running FDISK? That would be sad, wouldn't it? On the other hand, maybe the drive really was trashed. Maybe I was wasting time trembling about the possible effects of FDISK. Why don't I look at the disk with DEBUG'S "L" command and see if the directory is even there?

DEBUG

That was doomed to failure from the start, and I have to admit I knew it. But I thought I'd try anyway, just in case the good fairy intervened when DEBUG was run.

"Hold it, hold it," you say. "Why did you know DEBUG wouldn't work to look at the disk?" DEBUG uses DOS INT 25h and 26h to read and write to logical sectors on the disk. If DOS doesn't recognize the drive (and we already proved that), INT 25h and 26h won't work.

"logical" stuff. I'm not a Vulcan, you know).

I asked Bill, the only one left at the office by now. He didn't have anything, either. I think Norton may have done something like what I need, but I've

Don't panic yet. Take a deep breath. Think. What could cause this? Where do I start?

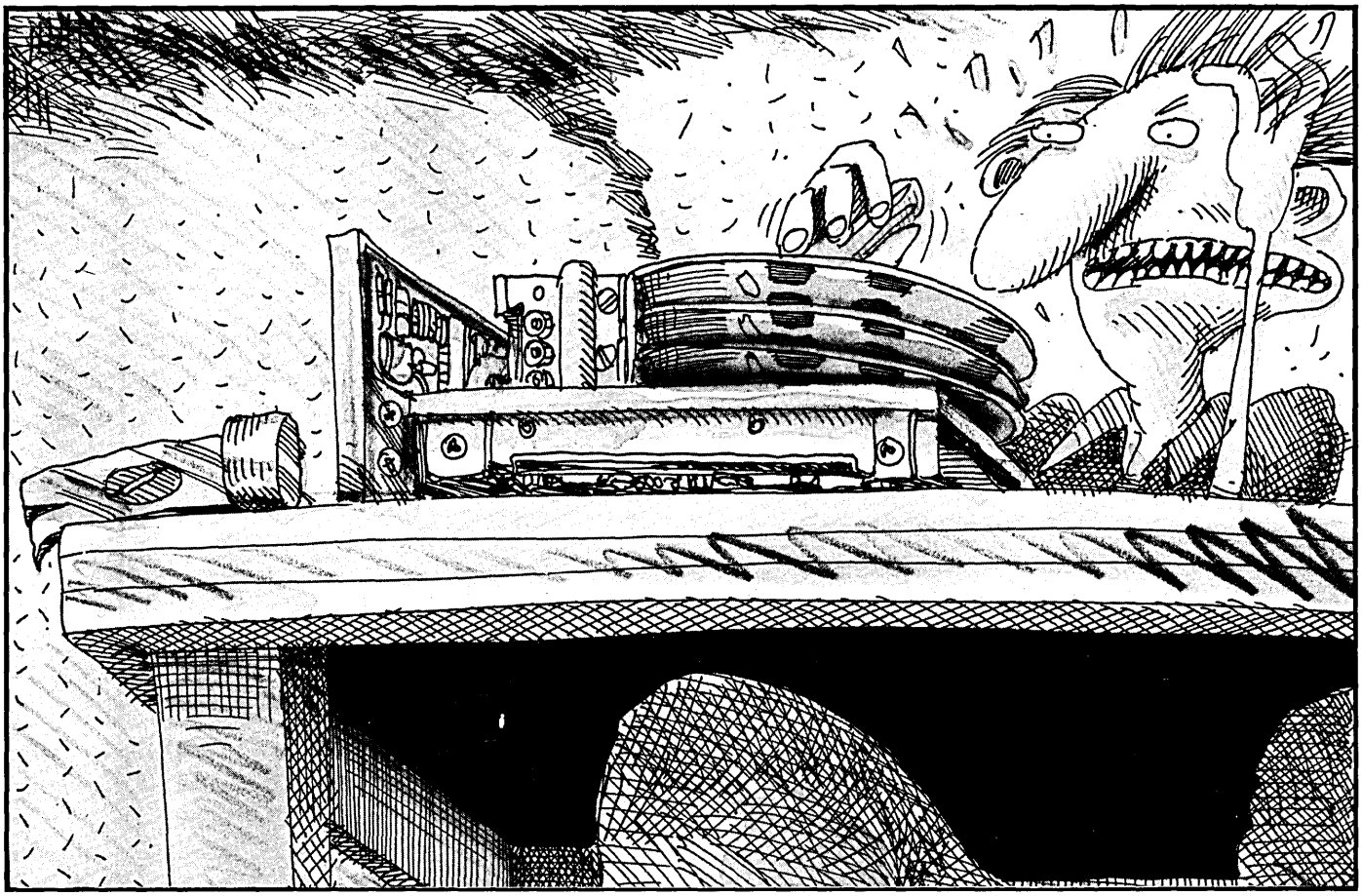
and a few books, and write the damn program myself. Maybe it wasn't such bad luck that this happened on Friday after all...

First Pass

Well, what routines do I need? Any program needs all kinds of cosmetic junk. The main needs of this program are 1) reading a disk sector using INT 13h; and 2) displaying a sector of data on the screen in hexadecimal.

I wrote most of my old programs in assembly or in Turbo Pascal. I don't like to admit it, but I still keep around a copy of Pascal 3.0. When I don't feel like assembly, and I'm moving from system to system and don't want to keep downloading a 500K library, I still fall back on my 40K Turbo 3.0.

I'm settling down more in my old age. I always seem to have a winchester or two around, so large libraries and



The good fairy wasn't doing me any favors that day, either. Sure enough, it didn't work. Next try: PC Tools.

The disk editor in PC Tools didn't work, either. It obviously was using DOS, too. What I needed was something to read sectors directly through the PC ROM BIOS INT 13h. It reads PHYSICAL sectors (none of this stupid

never had the guts to walk into a computer store and admit I needed anything from Pink Shirt, Inc. Kind of like a high school student going into the drug store where the cutest girl in town works and asking for Preparation H.

Anyway, I didn't have anything to do what I needed. The only solution was to go home, hole up with my X16

gigantic executables aren't a concern anymore. But my history has filled my backup disks with loads of useful little Pascal procedures.

So I sat down Friday night, and by Saturday afternoon I had a Pascal program that could read and write sectors. I took it over to the office on a floppy and started up my poor little X24. Sure

enough, the directory was still there. In fact, it was written right over the partition table!

Well! A fine tune I've played this time. I rushed back home, which is just a few blocks (anywhere is "just a few blocks" in Lake City, Minnesota), and ripped a couple short little procedures out of another program to give my disk viewing program the ability to save sectors to a file on another disk. And to put those same sectors back into another location on the same disk. Or into the same location on another disk. ("AHA!" you cry from the balcony.)

Into the rust bucket for a speedy trip back to the office. I hopped back to the stockroom, grabbed a spare 40 meg drive, and FDISKed it up just like my own. Then I loaded up my newly written program (dubbed "DISKEDIT") and saved the partition sector in a file on the floppy. A quick swap of drives back to the original, and I wrote the partition sector stolen from the spare drive onto mine.

Now for the moment of truth. I punched the floppy out of the drive, hit reset, and BINGO, my machine booted up just like it had on Friday morning. Elapsed time, 26 hours.

Epilogue To Part One

My partition table has been overwritten three more times in the last two weeks. Since the machine was running solid for several months until the day I plugged in my new Sunshine EPROM programmer, I have decided that the EPROM programmer is the cause of the problem. Either that, or it's all those static shocks I keep getting every time I grab for the texttool socket.

Since I have the partition sector saved on a floppy now, and a program to reload it, I haven't bothered to yank out the EPROM programmer or figure out why it gives my machine the DTs.

Rewrite

I gave myself Sunday morning off. (Actually I had taken Saturday morning off, too, but I don't want to admit it. You'd think I was lazy.) Sitting around listening to the soothing harmony and ingenious counterpoint of African kora music, I began thinking of just how dog-ugly my newly written program was. At first it just bothered me. Then it irritated me. Finally, I was obsessed with the idea of rewriting the program. And doing it right this time.

Sunday night I was back at it. Using Zortech C++ this time.

Figure 1 — Sample DISKEDIT output

```
                                        Laine's Mighty Disk Editor  v0.0 02/13/89

 CURRENT        |  0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F | 0123456789ABCDEF
Drive:  80      |                                                 |
  Cyl:    0     | 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
 Head:    3     | 01 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
  Sec:    7     | 02 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
                | 03 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
 MAXIMUM        | 04 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
  Cyl:  610     | 05 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
 Head:    3     | 06 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
  Sec:   17     | 07 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
                | 08 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
                | 09 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
Write Prot      | 0A 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
                | 0B 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
                | 0C 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
                | 0D 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
                | 0E 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................
                | 0F 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 | ................

   PgUp/PgDn - Prev/Next Sector,  F10 Updates Sector (alt+F10 enables update)
 G - Goto Sec, M - Modify byte, R/W - Read/Write Sec to/from File, ESC to quit
```

♦ ♦ ♦

Figure 2 — disks.c - Interface to IBM ROM BIOS disk routines

```
/* lrs 2/13/89   No Rights Reserved. Use it for what you like. */

#include <bios.h>
#include <dos.h>
#include "disks.h"

void InitDisks()
  { /* ROM BIOS "Reset Disks" command */
  union REGS reg;
  reg.x.ax = 0;
  int86(0x13,&reg,&reg);
  }    /* InitDisks */
/*-------------------------------------------*/
void GetDiskParams(int drive, int* maxcyl, int* maxhead, int* maxsec)
  { /* get size of disk */
    /* (note: this (and the rest) assumes 512 byte sectors */
  union REGS reg;
  reg.h.ah = 8;
  reg.h.dl = drive;
  int86(0x13,&reg,&reg);
  *maxcyl = reg.h.ch + ((reg.h.cl & 0xC0) << 2);
  *maxhead = reg.h.dh;
  *maxsec = reg.h.cl & 0x3F;
  }    /* GetDiskParameters */
/*-------------------------------------------*/
int ReadSector (int drive, int cyl, int head, int sec, void far* buf)
  { /* read a 512 byte sector from phys location on disk */
    /* return TRUE if success */
  union REGS reg;
  struct SREGS sreg;

  int retry = 3;;
  do {
    reg.x.ax = 0x0201;
    reg.x.bx = FP_OFF(buf);
    sreg.es = FP_SEG(buf);
    reg.h.ch = cyl & 0xFF;
    /*top two bits (8 & 9) of cyl # hidden in bits 6 & 7 of sector*/
    reg.h.cl = sec | ((cyl >> 2) & 0xC0);
    reg.h.dh = head;
    reg.h.dl = drive;
    int86x(0x13,&reg,&reg,&sreg);
    } while ((reg.x.cflag) && (retry-- > 0));
  return(reg.x.cflag == 0);
```

Continued on page 60

SHUGART 3 1/2" FLOPPY DRIVES

Brand new

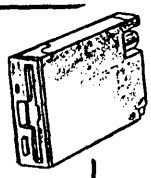
S.S.S.D

48 T.P.I

$39.95

MODEL # SA-300

3 1/2" FLOPPY DRIVES



720K Byte
Double sided
Mounting kit
Power & Data cable adapter

IBM COMPATIBLE

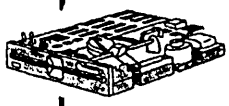
$119.95

TOSHIBA

Model # FDD 4210GOK

FULL 90 DAY WARRANTY

TOSHIBA 5 1/4" FLOPPY DRIVES



DSDD, 360 K
1/2 HEIGHT
BLACK FACE PLATE

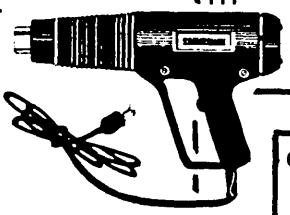
(90 DAY WARRANTY) $89.95

SELECTED VALUES

HEAT GUN

U.L. Approved

2 Speed, dual temp.



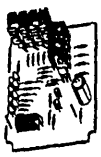
Easy Power#72101

With Stand **$33.95**

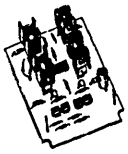
TECHNA-KIT

D-C Motor Controller

- Control 2 D-C motors with a computer or other logic source
- For motors rated 6-24 VDC
- Control forward/reverse/run/cw/ccw/stop
- Up to 6 Amp starting surge, 4 Amp cont.
- Dynamic breaking (capable)
- Will also run most 4-lead stepper motors



$29.95

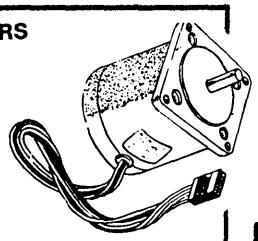


USMD-C

- Control standard 6-lead stepper motors with a computer or other logic source
- For motors rated 1.7 - 12.0 VDC
- Optical isolation
- Control: forward/reverse/step rate/stop
- Industry standard 22 pin edge card connector

$29.95

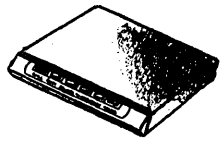
STEPPER MOTORS



6 lead Hybrid
1/4" SHAFT X 7/8"
35 OZ. In Torque
6VDC
200 Steps/Rev
O.D. 2 1/4" X 3"

SIGMA #20-2223D-23509 $9.95

COMPUTER POWER SUPERVISOR

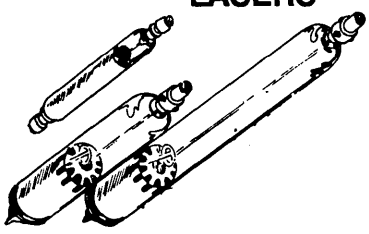


PROVIDES SURGE/SPIKE PROTECTION FOR YOUR COMPUTER!

- 5 OUTLETS, EACH WITH A LIGHTED SWITCH
- 1 MASTER ON/OFF SWITCH
- SYTLISH CABINET PROVIDES FULL SHEILDING
- RATED 15 AMP, 125 VAC, 1875 WATTS

$49.95

LASERS

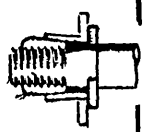


5 MW Laser Tube	$89.95
Power Supply Kit (115VAC)	$69.95
Power Supply (wired) (12VDC)	119.95
1 MW Laser Tube	119.95
Power Supply (Bat. Pwrd.)	79.95

(These lasers are brand new and guaranteed to have a cosmetic defect or not meet manufactures full specifications. All are tested in our lab to insure your satisfaction.)

INSTANT METAL THREADS

Complete starter kit



Includes: 5 Insert tools
4 Allen Keys
1 Hex wrench
8ea. Aluminum thread sets
6-32, 8-32, 10-32, 10-24 and 1/4-20

w/carrying case $19.95

M-F 9-6
SAT 9-5

united products corporation
DISTRIBUTORS OF ELECTRONICS SINCE 1968

1123 VALLEY STREET • SEATTLE, WA 98109-4425

VISA · AMERICAN EXPRESS

PHONE: **(206) 682-5025**
FAX: **(206) 682-5593**

Reader Service Number 101

DISKEDIT

The results of my labors are much nicer than the original, ugly Pascal program. DISKEDIT allows you to look at any sector on your hard disk, even if DOS doesn't recognize it. You can read and write sectors, save sectors to a file or recall them from a file, and modify specific bytes within a sector.

Display

The screen displays the bytes of a sector in a fashion similar to DEBUG, but in a more modern, screen-oriented fashion (i.e., silly little boxes around the display). Current cylinder, head, and sector are displayed at the left side of the screen, along with maximum cylinder, head, and sector. See Figure 1 for a sample display.

Big Display

Sectors on PC compatibles are all 512 bytes. At 16 bytes per line, that's 32 lines for a full sector. That's no problem for me, with my 66 lines of 34010 Monochrome display. Most poor unfortunates are still stuck with 25 lines. While writing DISKEDIT, I was stuck with the normal dilemma of deciding whether to make myself happy (write the program to use the full screen), or make everybody else happy (just use 25 lines).

In the end, I was able to make everyone happy. This was partly possible because I also wrote the video library I used for DISKEDIT. VIDEO.C includes a function called screenrows() which returns the number of lines on the display installed (it looks in the EGA standard location of 0040:0084h in BIOS RAM).

DISKEDIT uses that information to decide how many lines of the sector it can display at once, and whether it should move a full sector, or a half sector, when it sees PgUp or PgDn.

If I run DISKEDIT at home on my Hercules display, I'll see half a sector at a time. If I run it at the office on my PC Tech Mono 34010 board, I'll see the entire sector at once. An EGA in 43 line mode is big enough to display an entire sector, too.

All this is automatic. It was easy to do, too. Took about 20 minutes. See NextSector(), PrevSector(), and DisplaySector() for examples of using screenrows().

DISKS.C

This is the *real* topic of this column. Reading and writing sectors in C using INT 13h. As usual, the code (see Figure 2) is mostly self-explanatory, but a few things bear discussion.

```
  }   /* ReadSector */
/*-------------------------------------------*/
int WriteSector (int drive, int cyl, int head, int sec, void far* buf)
  { /* write memory block to specified physical sector on disk */
    /* return TRUE if success */
  union REGS reg;
  struct SREGS sreg;
  int retry = 3;

    do {
      reg.x.ax = 0x0301;
      reg.x.bx = FP_OFF(buf);     /* THIS MUST BE FIXED !!!! */
      sreg.es = FP_SEG(buf);
      reg.h.ch = cyl & 0xFF;
      /*top two bits (8 & 9) of cyl # hidden in bits 6 & 7 of sector*/
      reg.h.cl = sec | ((cyl >> 2) & 0xC0);
      reg.h.dh = head;
      reg.h.dl = drive;
      int86x(0x13,&reg,&reg,&sreg);
      } while ((reg.x.cflag) && (retry-- > 0));
    return(reg.x.cflag == 0);
  }   /* WriteSector */
```

♦ ♦ ♦

Figure 3 — Excerpt from VIDEO.C - Write16Bytes()

```
void Write16Bytes (void* buf)
  { /* write the 16 bytes at *buf as hex digits, followed by the
       ASCII equivalent of same, at the curent cursor location.
       This routine is a bit specialized to have in a general
       purpose library, but I wanted faster screen paints. That's
       one of the advantages of writing your own library - you get
       to change it! */
  int ct;
  BYTE* this;
  WORD far * scrptr;

  this = (BYTE *) buf;
  scrptr = (WORD far *) (screen + (currow * scrcols) + curcol);
  for(ct = 0; ct < 16; ct++)
    {
    *scrptr++ = nibs[(*(this)) >> 4]+curcolor;
    *scrptr++ = nibs[(*this++) & 0x0F]+curcolor;
    scrptr++;
    }
  this = buf;
  for(ct = 0;  ct < 16; ct++)
    *scrptr++ = curcolor
              + (*this > 0x1F ? *(this++) : (*(this++),'.'));
  at(currow,curcol+65);
  }    /* Write16Bytes */
```

♦ ♦ ♦

If you look in any reference for the PC ROM BIOS, you'll see that INT 13h disk functions use register CH for cylinder, DH for head, and CL for sector. You'll notice some bit shifts where I load these registers, though. That's because of a little detail they didn't know about back when some of those books were written.

In the old days, when all disks were floppy, they didn't have more than 40 tracks. A single byte was plenty of space to store the track number. Just about any winchester has at least 600 tracks, though. That means you need more than a byte to tell the ROM BIOS which track you want.

PC compatible disk BIOSes have solved this by adopting a standard that uses 10 bits for cylinder. Bits 0-7 are passed in their original manner in register CH; bits 8 & 9 are passed in bits 6 & 7 of CL (the bottom 6 bits of CL contain the sector number).

Within the C code, everything is integer (16 bits). But when I make the actual call to INT 13, I grab bits 8 & 9 of cylinder, shift them right two bits, and OR them in with the sector number. In GetDiskParams(), which returns the disk size, I do just the opposite: shift bits 6 & 7 of sector left two bits and OR them with the rest of the cylinder number.

This is part of the reason I didn't use the biosdisk() function I described a few issues ago. I couldn't trust it to do all this bit fiddling. Besides, it isn't in the Zortech library (it's just in Turbo C). int86x() (used to execute the INT 13h), on the other hand, is in both libraries. I feel much safer setting up the arguments and doing the INT myself.

FP_SEG()

You also may not recognize FP_SEG() and FP_OFF(). These are macros defined in DOS.H which extract the segment and offset from a far pointer. int86x() doesn't understand a pointer as such. It takes two integers, one of them the segment portion of the pointer, the other the offset of the pointer.

After looking at the definition of FP_??() in dos.h, I couldn't decide whether it would work properly with a near pointer, so I forced the pointer I was using to be a far pointer. To make it work properly, I had to force a type conversion of the pointer passed to ReadSector() and WriteSector(), like this:

```
(char far *) buf
```

If I didn't do this, I might end up with the call to ReadSector() sending only an offset for the pointer, while ReadSector itself expects a segment. Actually, I *did* do this the first time. Fortunately, I always use function prototypes (and Zortech C enforces them) so I was properly informed, and fixed it.

VIDEO.C

You've seen most of VIDEO.C before (see Figure 3 for excerpts from VIDEO.C). I used it a year or so ago when I talked about the keyboard hardware interrupt (INT 9). The only things that have changed are that I have added recognition of larger screens, and I put in a new procedure: Write16bytes(). In fact, Write16bytes() is the only part that is listed in the magazine. The rest is available on the Micro C bulletin board, or on disk from Micro C.

Write16Bytes() takes a pointer to a 16 byte array and writes it on the screen in "DEBUG" format. I haven't decided yet if it should really be in a general purpose library, like VIDEO. At first I was duplicating the task of Write16Bytes() with calls to cprintf() in a loop.

This worked just fine, except it was slooooowww. It took at least a quarter of a second to paint. The problem was the overhead of all those calls, formatting two character strings, building a screen pointer, updating the cursor ...

Write16Bytes() takes care of all that by building its own pointer. Once. Then it pokes everything right into screen memory, updating the cursor position just once, after the entire line is finished.

But, like always, I had to sacrifice some generality to get blazing speed. (It's instantaneous now. Watch this...)

Other Modules

When I wrote the original program in Pascal, I put everything in a single file. It was big, ugly, and big. When I did the hand translation to C (okay, first we do a global replace of "begin" with "{." No. That's not it. First we replace "{" with "/*," *then* replace "begin" with "{"), I left it all in the same file at first.

After I got it working, I separated it into reasonable pieces and made a .h file for each. Not only did this make it easier to think about conceptually, it also made it easier to look at on the screen (especially with all those Point Editor windows on my big screen). And having all those files in the \project\diskedit directory looks impressive.

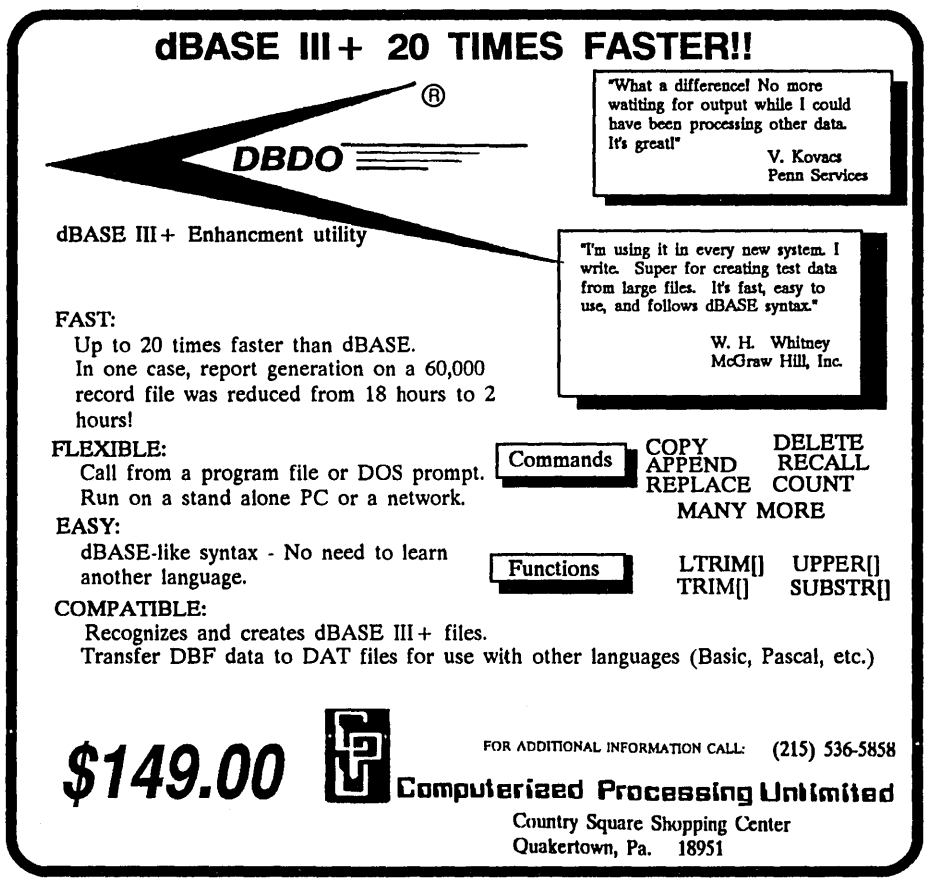
I used Zortech C to test all this code, but I used my Turbo C reference manual to find all the library routines (Earl had the Zortech book at home). I originally wrote VIDEO for Turbo C. Because of all this, I am fairly certain that all the modules of DISKEDIT will compile without change under Turbo C. If I ever get bored with Zortech, maybe I'll try it. Don't hold your breath.

A Sequel??

Well, I happen to have some procedures (Pascal, of course) lying around that convert sector number to cluster number (a cluster is a group of sectors), decipher file allocation tables, and a few other things, as well. I want to explain exactly the contents of the partition table and use it to locate logical disks. By the time I'm done, I'll be chaining clusters, following cluster chains, deciphering directories, and directing decipherers.

P.S. I have a MASM tip to add to Richard Lamb's (in the last issue). Not only are DWORD local variables weird, so are BYTEs. The size allocated on the stack for local BYTE variables is sometimes wrong. It is always at least two bytes, for one thing. You might as well declare them as WORDs.

♦ ♦ ♦



Reader Service Number 105

Conquering A New Dimension

Anthony Barcellos
P.O. Box 2249
Davis, CA 95617-2249
Voice: (916) 756-4866
Data: (916) 758-1002

Lotus' sweep of the PC spreadsheet market left early competitors dead in their tracks. Now one of those competitors is back, and the shareware market has a new, very powerful contender.

Most sheets are rectangular and thin. Spreadsheets, for example. Usually, we three-dimensional users make do with two when it comes to number crunching.

Suppose we plump up a spreadsheet into the forbidden third dimension. What do we call it now? "Spreadpillow" or "spreadmattress"? What's the right 3-D metaphor?

Power Sheets

Maybe we could just call it Power Sheets, a new incarnation of Datamension's Report Manager. *PC Magazine*'s Jared Taylor singled out Report Manager as one of the best products of 1984, back before Lotus 1-2-3 managed to stamp out virtually all its early competitors.

After a quiescent period, Report Manager has come back to life under the ministrations of its creator, Al Baker. Baker has revised the program and relaunched it into the shareware market as Power Sheets, version 2.25E.

So what does Power Sheets offer? Al Baker says, "This Shareware version of Report Manager is still, by far, the most powerful pure spreadsheet product on the market." Baker lists Ford, Amoco, and First Chicago among his corporate customers, citing their development of turnkey applications with the features of Power Sheets.

Bigger Can Be Better

That sounds impressive. The list of specifications bears out that initial impression. Power Sheets offers a work area that has 255 columns, 255 rows, and 255 pages. That's a total of over 16 million cells.

Columns and pages are each labeled from A through IU. Rows are numbered from 1 to 255. Absolute cell references look like "B3F" or "AB103CD," as you might expect. Relative references are fairly simple: "@(2,3,4,B3F)" means the cell 2 columns to the right, 3 rows down, and 4 pages behind B3F. If you use "&" in place of the cell name, the reference is taken relative to the current cell.

Ranges can be linear, rectangular, or solid blocks. Just specify the endpoints of a linear array, diagonally opposite corners of a rectangle, or diagonally opposite corners of a 3-D block.

Power Sheets doesn't stint on work space, but its "power" comes from its array of functions and the EXEC programming language. The standard functions include: ABS, AVG, DATE, MAX, MIN, RND, SQRT, STD, SUM, TIME...

Baker throws in a few twists like: variations on the IF function for branching; a lookup feature customized by row, column, or page; MODE to find the most frequently occurring value in a list; and SSQ to sum squares. His collection of logical functions is impressive, the list of trigonometric functions is complete, and the logarithmic function comes in both natural (base e) and common (base 10) forms.

Power Sheets has a CASE function that I have seen in no other spreadsheet. An example explains it best. CASE(A1A,"No match",0, "Zero",1,"One",2,"Two",3,"Three") takes the value of cell A1A and compares it to each of the following odd-numbered entries.

If it finds a match, the CASE function returns the immediately following entry as its value. If it finds no match, CASE returns the second value. The CASE example above would convert the number 0, 1, 2, or 3 to its alpha equivalent and returns "No match" if A1A has none of the specified values. Neat.

EXECute Your Own

If Power Sheets doesn't have the function you need, you can create your own with the EXEC programming language that Al Baker includes as part of his spreadsheet. While the EXEC documentation is terse, to say the least (six pages or so out of the 55-page manual), all is not lost.

You get a suite of EXEC examples that perform amortization, internal rate of return, linear regression, net present value, and loan pay-

ment. In addition to these and other financial calculations, the EXEC examples can issue a CD (change directory) command and perform iteration.

Parsimonious Power

By now you may be getting nervous. Will your poor old PC be able to handle such a powerful program? In truth, the system requirements are astonishing. Astonishingly small.

Power Sheets is perfectly happy on a dual-floppy 256K PC. The program grabs 70K for its own use and the rest for data storage. If you have more room, it will be glad to take advantage of memory up to 640K.

While you can't expect to fill all the cells of the 255 by 255 by 255 work area, Power Sheets uses a sparse matrix management scheme to minimize wasted data space. (Lotus is just now getting around to that, if they ever manage to ship release 3.0 of 1-2-3.)

The Power Sheets registration fee is as parsimonious as its sparse memory mapping. Baker offers full registration for only $19.95, an incredible bargain. (How could you go wrong?)

For an additional $10 he will mail you a two-disk set with the latest version of the program. Ten dollars more entitles you to a year of technical support. A disk of sample spreadsheets with on-disk narrative, EXEC programming language tutorial disk, and an advanced guide to power features are $5 each. Send Al Baker a check or money order for $50 and you receive all the above in one bundle.

If you create a wonderful turnkey system with Power Sheets that you would like to distribute, Baker offers unlimited distribution rights for a flat $100 fee. He'll provide a version of Power Sheets that will not permit your users to break out of the EXEC program (and thereby gum up your application).

Power Sheets is not just another pretty Lotus clone. Yes, it uses good old F1 for Help, but the Help screen is actually a menu system from which you can select operations.

While it's too bad that Power Sheets won't read Lotus worksheets directly, it supports DIF files for import or export. Power Sheets can read or write delimited ASCII files (your choice of delimiter), but Baker notes that it also works with non-delimited files. (I don't know how he does that.)

Baker has created a remarkable package that deserves a special niche of its own. Try it out and see if you agree.

Power Sheets
Registration: $19.95
Total package: $50
Al Baker
3936 Sunset Lane
Northbrook, IL 60062

Down And Out

Is this the end of the ARC wars? We can hope so, although it seems the users will never have a fully compelling resolution of the controversy.

The score stands at one and one. Software Enhancement Associates clobbered PKWare in round one, when Phil Katz folded like an accordion and yielded virtually all rights to PKARC and PKXARC.

Round two went to Katz, when SEA lost their contempt of court action over Katz's continued use of the terms "ARC" and "archive," which SEA claims to own. No further court action is anticipated, for which we may all be grateful.

SEA's archiving program will now presumably benefit from their acquisition of Phil Katz's code, so perhaps the original ARC will begin to approach the performance of the PKWare versions. Since SEA claimed that Katz stole his code from ARC in the first place, it might seem that SEA isn't getting that much back.

SEA deserves much credit for creating the ARC standard and popularizing the concept of combining and compressing several files into a single archive file. Phil Katz deserves credit for much of the concept's success, since his utilities were so much faster than SEA's. I well remember how sysops argued over the merits of ARC, the main sticking point being speed.

ARC was elegant, neat, and *painfully* slow. Many sysops preferred to keep on using SQZ to compress files and LU to combine them into LBR files. Even though it took two passes, it was much faster than ARC. PKARC and PKXARC swept those arguments away and archive files quickly replaced library files.

However, SEA is not responsible for the compression techniques used in creating archives. File compression algorithms have long been discussed and published in the computer press. Anyone using Huffman encoding, for example, would go to the same sources.

I'm not at all surprised that SEA's software expert found matching code in ARC and PKARC. It is just not clear that the similarity between the two products goes beyond simple incorporation of identical public-domain algorithms for compression and expansion of files.

Despite the disclaimer of any admission of wrongdoing, the court-approved settlement in *SEA v. PKWare* made Katz look bad. (No one has much faith in such disclaimers.) Either Katz swiped the code or he was forced into submission for some other reason.

As I noted a couple of columns ago, one argument was that PKWare was too small to stand up to SEA in court. I was promptly taken to task in our "Letters" column (*Micro C* Issue #46) for claiming that SEA is a "much larger" company, although I was merely reporting the most popular rumors among Katz supporters.

I also reported the rumor that Katz was caught red-handed with stolen code. Perhaps as one inclined toward Katz's side, I must automatically espouse the claims of all his supporters.

Nevertheless, it's interesting to be criticized for "idle, misinformed speculation" by someone who follows his own arguments with "All of this, of course, is speculation." I think that encapsulates the entire controversy, which will remain forever embroiled in speculation. Hard facts would be so much nicer. In their absence we can espouse the criterion "my speculation is (at least) as good as your speculation."

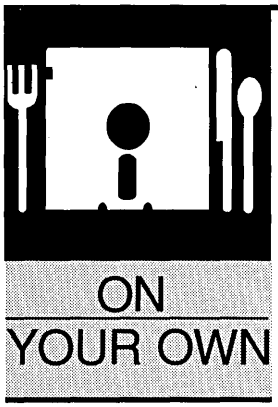
SEA has argued persuasively that it is a mom-and-pop operation on the same scale as PKWare. However, until recently, SEA pursued corporate accounts while PKWare worked the user community. One can reasonably question SEA's claim that its resources are much smaller than PKWare's. Again, no hard facts.

There is no definitive answer, whatever the partisans may say. Much of the user community will continue to regard SEA's lawsuit as substituting court action for competition. Even if we accept SEA's argument that Katz transgressed their proprietary rights, one has to marvel at the extent of SEA's claim to the very notion of archiving.

The term "archive" and the operation of "archiving" has a much longer history than SEA is willing to acknowledge. ARC is a recent chapter in a book, *not* the whole book.

The final round is now in the court of public opinion. SEA's brief to the user community will persuade some and soften the opposition of others. Yet one fact remains: The shareware community has been split into two factions. I hope the rift will narrow with time.

♦ ♦ ♦



Starting A Robotics Company

By Norbert Bukowski
Bukowski Robotics, Inc.
6125 South Maple, Ste. 3
Tempe, AZ 85283
(602) 838-2889

When it comes to business, there's nothing magic about robots. Like anything else in computers, you need a lot more than ideas and educations. You need guts, money, and maybe even a bit of luck.

Bukowski Robotics, Inc., a small systems integrator and robot manufacturer, provides a great example of what to expect and what to avoid when you go out on your own.

Robotics is one of the most rapidly growing and changing of the High Tech industries. In this field, you can look forward to difficult challenges and substantial rewards because there's an equal chance for great successes and incredible blunders.

Getting into business is often the hardest step. A technical degree and a great idea aren't the only requirements. You need startup capital and you need to understand how to manage it.

My $30,000 seed capital came from an antique car restoration business that I started in college; not much in today's world. Automotive restoration is so different from a High Tech start-up that few of my business management skills applied. As I soon discovered, I might as well have started at zero.

A start-up company has to run very lean (car clichés distributed at random). Since checks can take 90 days or more to show up, cash flow becomes the major headache.

When we started the company, we thought we had a hot product so the money would come rolling in. The product bombed and it wasn't long before we found ourselves in a sink or swim situation with no life boats in sight.

How I Got Started

My first exposure to computers came in high school, circa 1976. I can still see vividly a 300 baud teletype printing out an image of the U.S.S. Enterprise in asterisks. I was not impressed. The mainframes at the time appeared ominous.

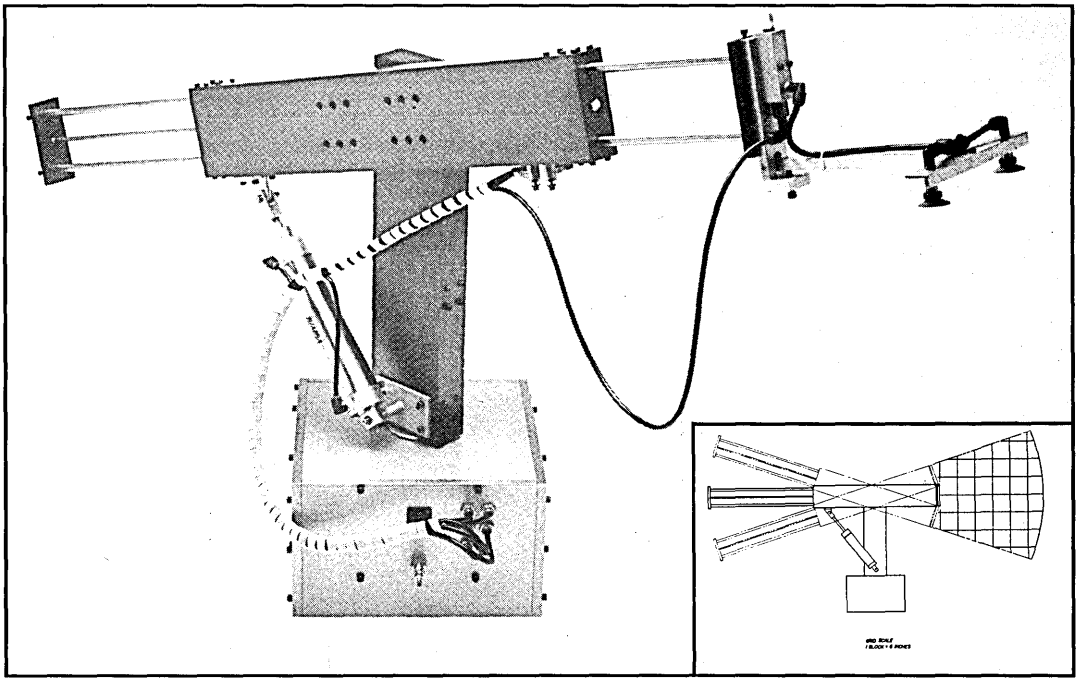


Figure 1 — BR 100 Series Pick and Place Robotic Arm.

CALL 1-800-888-8087 to order (VISA or MC)

	U.S.	CAN/MEX	FOREIGN		TOTAL
			Surface	Air Mail	
1 Year — 6 issues	☐ $18	☐ $26	☐ $36	☐ $50	
2 Years — 12 issues	☐ $34	☐ $50	☐ $68	☐ $100	
3 Years — 18 issues	☐ $48	☐ $72	☐ $99	☐ $150	

U.S. Subscribers SAVE 24% OFF NEWSSTAND PRICE*

*Savings based on one-year cover price of $23.70

YES! I want to subscribe!

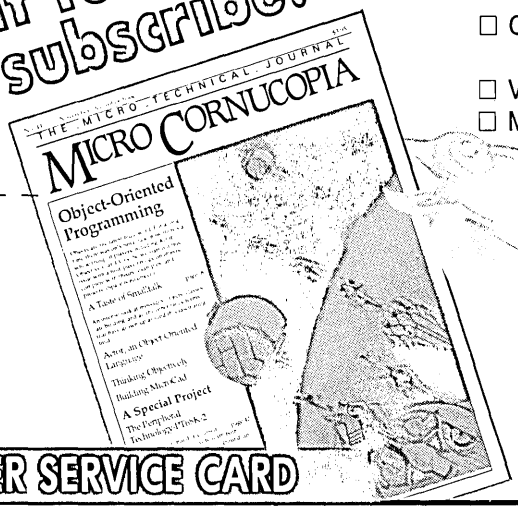
☐ New
☐ Renewal

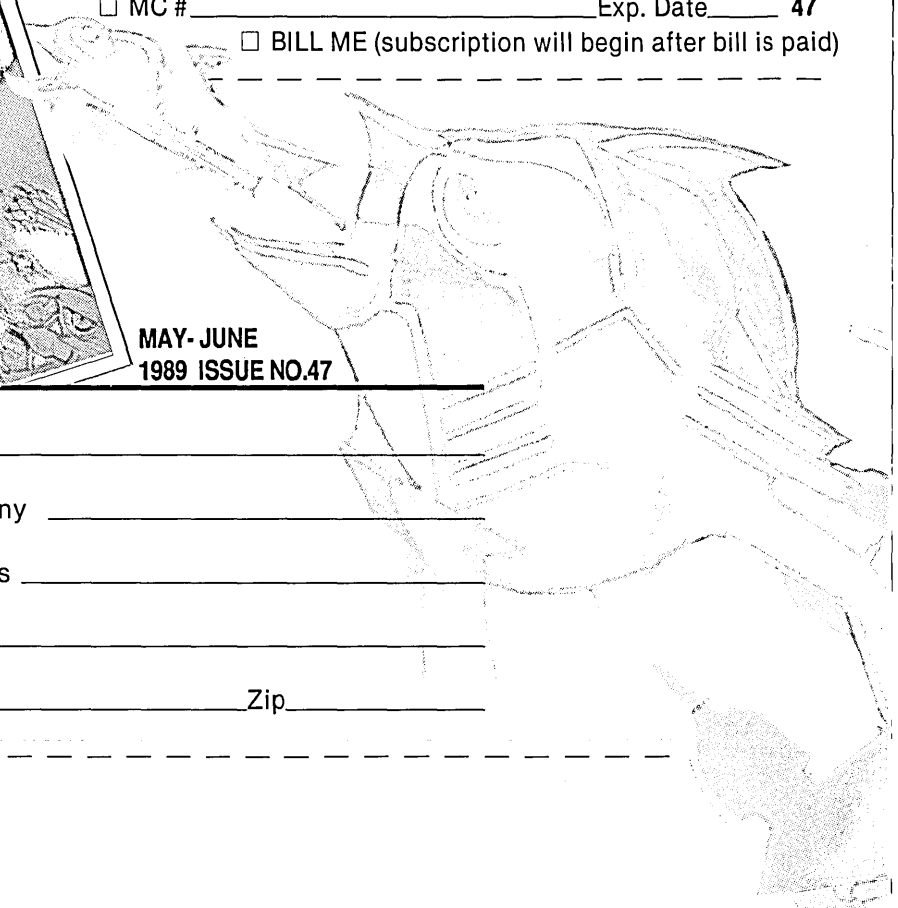
Name _____
Company _____
Address _____
City _____ State _____ Zip _____

☐ CHECK Payable to Micro Cornucopia, Inc. P.O. Box 223, Bend, Or 97709 (U.S. dollars drawn on a U.S. Bank)

☐ VISA # _____ Exp. Date _____
☐ MC # _____ Exp. Date _____ **47**

☐ BILL ME (subscription will begin after bill is paid)





READER SERVICE CARD

MAY-JUNE
1989 ISSUE NO.47

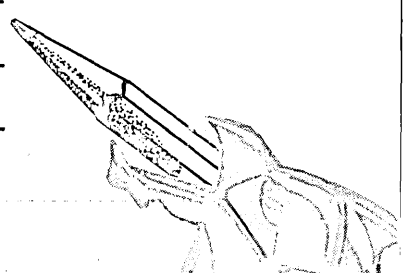
Write in the reader service numbers of any advertisers from whom you would like to receive free information.

Name _____
Company _____
Address _____
City _____
State _____ Zip _____

Yes!

Our Company would like to Advertise in Micro C

Name _____
Company _____
Address _____
City State Zip



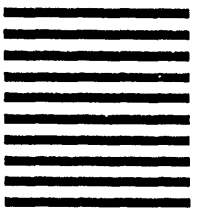
NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO.19 BEND, OR

POSTAGE WILL BE PAID BY ADDRESSEE

Micro Cornucopia
P.O. Box 223
Bend, Oregon 97709-9982

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES



BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO.19 BEND, OR

POSTAGE WILL BE PAID BY ADDRESSEE

Micro Cornucopia
P.O. Box 223
Bend, Oregon 97709-9982

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS MAIL PERMIT NO.19 BEND, OR

POSTAGE WILL BE PAID BY ADDRESSEE

Micro Cornucopia
P.O. Box 223
Bend, Oregon 97709-9982

A few years later I met the KIM I. That was impressive! The single-board KIM I contained a small affordable CPU and easy access digital I/O.

I had previously assumed that computers were out of touch with the real world. They were closed systems. I was eager to experiment with a computer that could touch my world. Before long I had built simple interfaces for the KIM I and used it as a controller.

My second computer was an Apple II. This computer was easy to use and very popular. The stock machine didn't have digital I/O so I installed a wire-wrapped 6522 VIA (interface chip). This gave me an easy way to expand my KIM I experiments.

A curious car customer spotted one of my projects, a stepper motor contraption. He happened to be the owner of a large hair care products factory and asked if I could build a machine for him. He wanted an automatic plastic tube bender which could make those squirt bottle nozzles that athletes use to dispense Gatorade.

I thought it would be a good challenge and agreed to give it a try. It was a simple machine and development went smoothly. Eight weeks later I had delivered my first automated machine.

While in school (I have a degree in Physics), I became friends with Lee Groves, a budding hardware engineer. Lee had dreamed up an auxiliary CPU card that would plug into one of the Apple II's slots. The board featured a 6502 CPU, 8K nonvolatile RAM, EPROM address decoding and, of course, the ubiquitous 6522. Obviously this was something the world needed.

I built the tube bender and then figured the market was ready for Lee's deluxe I/O card. I sold my car business and jumped in with all my savings and enthusiasm. In no time, I had a huge payroll, enormous rent, gigantic advertising bills, trade show expenses, and no sales. I hadn't considered marketing.

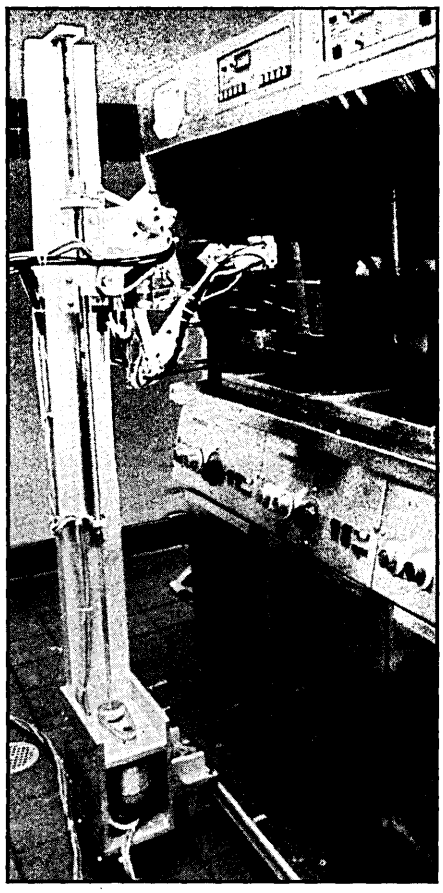
I had assumed that because I thought this product was great, and had many uses, people were just waiting to purchase it. Unfortunately, most Apple II users were still playing Star Trek.

My investment turned into a big flop, even though the product received good reviews from critics like Don Lancaster (*Computer Shopper*, May 1985).

Sink Or Swim

Around this time, IBM made its big thrust into the personal computer market and any hopes of a maturing Apple audience vanished. I had to decide on a new direction, but the option of continuing in the board market with an IBM version made me queezy.

> We eventually worked out enough bugs to show the prototype.



Robo-Fries anyone? Burger King Prototype at work.

In marketing the Apple card, I realized that board level products that were not complete solutions (i.e., printer cards, modems, memory expansions, etc.) would be difficult to sell. And, there were already people producing the standard I/O boards. So I decided to get out of the board market completely.

As it turned out, the Asian board producers wiped out almost every domestic board company. I'm glad to see a resurgence from American companies in special application boards such as transputers, data acquisition, and digital signal processing. If you work in these areas, don't let me discourage you. I feel that your audience is maturing and growing (although still small).

I decided to pursue custom automation for the industrial market. I had built the tube bender without any problems. Finding customers was relatively easy. Unfortunately, my salesman had a habit of boasting that we could supply anything to anyone. We lacked focus, to say the least.

Subsequent projects were much more challenging than the tube bender. We struggled for over a year before getting enough of a handle on the technology to be able to promise what we could deliver and be able to deliver what we promised. (If you're in the business of selling things that have not been done by yourself, or anyone else, you'd better tread carefully.)

Robotic technology is mostly proprietary. Not too many people are eager to share their technology with you. In fact, you'll probably find that most other computer-related industries work that way.

The Big Fish

Not long after we changed over to custom automation, we got a great break. At a small party, I met a Burger King franchiser who jokingly suggested that I make a robot to replace his employees. It seems that he had trouble staffing his six restaurants.

After a short discussion, I decided that the fry station would not be too difficult. We had successfully built a pneumatic pick and place robot which performed a similar task, so we approached Burger King through the franchisee.

Our proposal was short and sweet, and we said little about technology. These were corporate types, not engineers. Don't bury your idea under too many facts or figures and don't expect quick results. Corporate momentum is not a myth.

It took over a year of phone calls and correspondence to close the deal. I can still hear the franchiser saying, "Don't give up." When the deal finally came through, we were sure it wouldn't happen. It was a great feeling to know that the technology we had worked so long to develop had impressed a large company.

However, we were a very small company faced with a very significant task. We were like a fisherman who hooks a fish too big for his tackle. Hooking Burger King was easy, bringing it home was difficult.

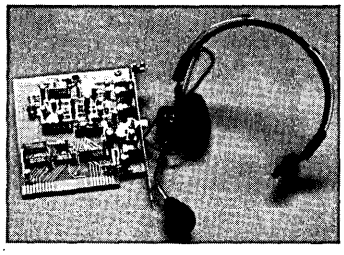
VOICE MASTER KEY™
VOICE RECOGNITION SYSTEM
FOR PC/COMPATIBLES & TANDY 1000 SL/TL

A FULL FEATURED VOICE I/O SYSTEM

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

SPEECH RECORDING SOFTWARE. . .Digitally record your own speech, sound, or music to put into your own software programs. Software provides sampling rate variations, graphics-based editing, and data compression utilities. Create software sound files or send voice memos through LANs. **A complete, superior speech and sound development tool.**

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



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

ONLY $149.95 COMPLETE

ONLY $89.95 FOR TANDY 1000 SL/TL MODELS—SOFTWARE PACKAGE ONLY.
Requires Tandy Brand Electret microphone.

ORDER HOTLINE: (503) 342-1271
Monday-Friday, 8AM to 5PM Pacific Time

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

CALL OR WRITE FOR **FREE** PRODUCT CATALOG

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

Reader Service Number 143

The idea which we initially sold Burger King required air to operate. The corporate types had not fully considered the technology and canned the pneumatics. At this point I should have squawked, but I chose to gloss over the resulting technology problems.

We were forced to try an all-electric design. Had we been a larger company, this wouldn't have been much of a problem. But we were small and a design change midstream always costs time and money. We eventually worked out enough bugs to show the prototype. We were at the end of our development money and needed to show well.

By this time the Burger King executives had changed three times. We had revised project specifications so many times they didn't know what to expect.

T3 300 Series Industrial Robot Line Card.

We had adhered to the general principles very well. These were to build a prototype, not a finished product. The machine performed the task it was meant to, albeit slowly. The corporate types were not impressed. I recommended that they look at it as a step toward solving a very difficult problem. We'll see what they do.

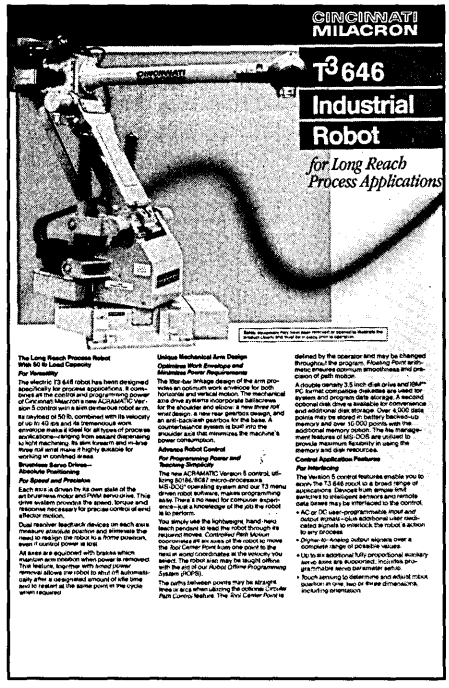
In The Meantime...

We're working on some other, very interesting projects. All can be characterized as innovative, and some as quite unusual. Our custom automation line has proven to be very reliable and steady.

In a few weeks, we'll release a new modular controller based on the G64/96 bus (Gespac Eurocard). It will be called UMAC, for Universal Multi-Axis Control. The 68000 CPU will run SK*DOS. We designed the motion control board around the new HP motion control chips. All other boards will be added as required from the G64 suppliers.

Agriculture control has been a long-term interest. We have a product in development that turns the growing environment (greenhouses, nurseries, farms) into environment responding servo systems. We call it the Agrobot.

Our unusual projects include Disney-fication and animation projects. Our Bukowski Robots have played minor roles in several Hollywood productions. They are also featured in various amusement parks.



T3 646 Industrial Robot Line.

By far, our most ambitious project is one that we hope NASA will use on its long-term space voyages. We're still in the proposal stage, so I can't say too much. I can say that it will be an autonomous robot with a very important function.

The lessons we've learned have been, at times, brutal. However, we're surviving, growing and having an interesting time. Whatever you choose to do, keep in mind that nothing comes easy, but that dreams can come true.

♦ ♦ ♦

ERAC CO.

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

AT/BABY AT

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

$1581

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

XT/TURBO

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

★ ★

$999

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

ELGAR

UNINTERRUPTIBLE POWER SUPPLIES

400 Watt MODEL IPS400+ $650

Power distribution center and sine-wave UPS. Only 2" high.

560 Watt MODEL IPS560 $350

Sinewave, 560W complete with batteries.

400 Watt MODEL SPR401 $180

Supplies may have minor cosmetic damage, but are electrically sound. Squarewave output. Run on internal or external 24VDC battery when line goes down. Typical transfer time = 12MS. Battery supplied. For AT, XT & Kaypro.

★

NEW 24V INTERNAL BATTERY $75

NiCds

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

GEL CELLS

12V 20ah	$25.00
12V 15ah	15.00
12V 2.5ah	8.50
12V 6ah	10.00
D Cell 2.5ah	2.00

ROBOTICS

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

KAYPRO EQUIPMENT BARGAINS

9" Green Monitor	83		$50
84, 1600			60
9" Amber CRT	$45	Keyboard	75
PRO-8 Mod. to your board			149
Host Interface Board			15

Replacement Power Supply	$50
Drivetek 2.2M FD	75

CPM COMPUTERS

K4-83	$350	K2-84	$400
K4-84	425	K4X	425

IC'S

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

TEST EQUIPMENT

OSCILLOSCOPES

TEK 7403N/7A18N/7B50A 60 MHz	$650
TEK 465 Dual Trace 100 MHz	1000
Scope Probe x1, x10 100MHz	35
USM338 50MHz Dual Trace Delayed Sweep	300

ANALYZERS

TEK 491 10MHz-40 GHz	$4500
Nicolet 500A 1 Hz 0-100 KHz	1800
Biomation 805 Waveform Rcrdr	259
Biomation 8100 2-Channel Waveform Recorder	795
HP1600A Logic Analyzer	600
HP1600A/1607A Logic Anlyzr	1000

MISC.

Optronics 550 MHz Freq Cntr	$100
Data Royal Function Gen F210A	200

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

CPU & SUPPORT CHIPS

MC68000-8 CPU			$8.00
Z80 CPU	.75	Z80A CPU	1.50
Z80 CTC			1.50
Z80A PIO	2.00	Z80A SIO	5.00
8089-3			6.50
80C85A	4.50	8088	6.50
8212			2.00
8251	1.50	8253-5	1.50
8255-2	3.50	8255-5	2.50
D8284A			2.50
D8749			7.00
MK48Z02B-20			10.00
Dallas D1220Y			10.00
SIP DRAM 256-12			7.00
41256-15			6.50
4164-10	2.50	4164-12	2.10
4164-15	2.00	4164-20	1.25
6845			5.00
1793	6.00	1797	7.00
VC3524 Switching Regulators			5.00
1458 Dual Op-AMP			.70
LM2877P 4W Stereo Amp Dual			2.50
MB81464-15			2.75
2716	3.50	2732	3.75
2764	4.00	27128	6.50
27256	5.25	27512	7.00
AM2901, 2903, 2904, 2910			Call

SWITCHERS

5V/9.5A, 12V/3.8A, -12V/.8A	$39.00
5V/3A, 12V/2A, -12V/.4A	19.50
5V/6A, 12V/2A, -12V/1A	29.00
5V/6A, 24V/1¼A, 12V/.6A, -12V/.6A	29.00
5V/10A	19.00
5V/20A	24.00
5V/30A	39.00
5V 100A	100.00
5V 120A	110.00

POWER SUPPLIES

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

★ ★ ★

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

Reader Service Number 93

Letters

Continued from page 6

when I found my favorite magazine in the mail, I noticed how fresh and clean it was. It's the plastic wrapper, I think. Thanks so much for the improvement.

In the past (since issue #10 or #11), I have had to endure significantly damaged copies of your loving effort. The mailman simply did not appreciate what he was bringing. Maybe it started an issue or two ago, but it was only this time that I noticed the great quality of the cover and the lack of a label hiding some precious bit of it.

Whatever inspired you to do the improvement, I commend you for having done it. Now I can leisurely peruse *Micro C* without grumbling half the time about the mailman wrecking the cover. Keep up the good work.

On another topic, I just got a nice letter from someone who got a Kaypro at a garage sale, discovered *Micro C*, and did all the speedups and Kaypro 8 disk improvements. He bought most of the available back issues, read my "On Your Own" article, and inquired about starting his own business doing the same thing I do (California Title-24 energy use documentation for new residential buildings, using energy use software).

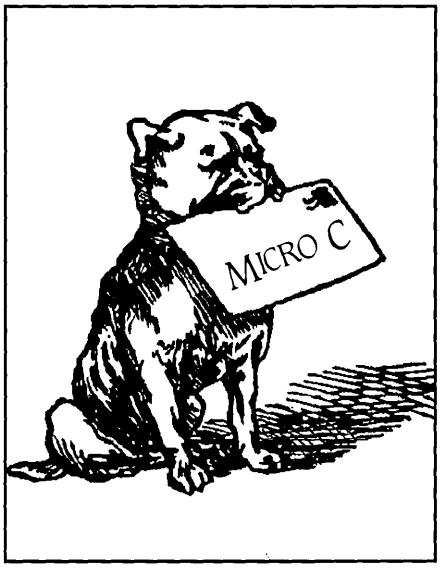
Of course I've responded to him with a letter encouraging him to seriously consider the effort. As always, I would have it no other way. Thanks again for the great opportunity to share my experience with others by publishing my "On Your Own" story way back in 1984.

Eric J. Torney
207 Kent Ave. #1
Kentfield, CA 94904

Editor's note: Guess what? Issue #46 didn't get plastic wrapped. The printer forgot to do it. (The mailing manager was sick, it wasn't on the work order, they were in a rush, no one thought of calling us to ask...)

A Taste Of Frustration

Wanting to expand my computer horizons, I began to read "A Taste of Smalltalk," *Micro Cornucopia* Nov-Dec 1988, by Steinman and Yates. Presumably this article was meant to be introductory. I absorbed the idea that messages are sent to objects and patiently indulged the authors' digression into classes.



But in the section on messages, I came unglued. "Messages are represented symbolically by selectors." What is a selector? The rest of the section brought in a new fuzzy notion at each sentence without clarifying anything that had gone before.

So I purchased the book *An Introduction to OOP and Smalltalk* by Pinson and Wiener. Within three pages I understood the basic concepts of objects and messages, and comprehended a sample program that reports the number of vowels in a file. Only after I felt at home with OOP did the authors lead me to deeper areas of abstraction, encapsulation, classes, inheritance, and polymorphism.

Paul G. Hershall
3308 Ferndale St.
Kensington, MD 20895

SEA Speaks

Now that the dust has settled in the first shareware copyright case, it is time for SEA to make public the facts that many members of the shareware community deserve to know.

As the creators, publishers, and defenders of the industry standard ARC file compression format, we have always maintained a strong belief in a fundamental concept of shareware — that shareware be distributed for free for all non-commercial use. To this end, we have never, and will never, charge for the use of ARC in a non-commercial environment.

We also believe that full program sources should be available and we have always made the full ARC sources available to all users. We have also licensed a great many people to use the ARC sources in their own programs.

We discovered that PKWARE had obtained our source code without obtaining a license. He [Katz] modified that code so that the program ran faster and provided several other enhancements. However, the nuts and bolts of the program were done by SEA. That is called PIRACY, plain and simple.

We tried to politely ask PKWARE to obtain a license. He ungraciously told us where to go.

We asked our lawyer what we should do. He said we were bound by law to protect our rights to the trademark and copyrights on ARC. If we did not, then anyone could use the ARC trademark and copyrights.

We didn't want to go to court. We couldn't afford the lawyers fees. We also couldn't afford to create ill will among users. But PKWARE left us no choice.

Anyway, the case didn't get very far, thanks to the testimony of an expert witness, John Navas. He looked at the source code of both programs and found, lo and behold, that the PKWARE program was indeed a blatant copy of the SEA code.

When Katz heard this, he called us directly — bypassing the attorneys — and said he wanted to settle.

We were only too happy to put a quick end to this. We wanted the facts to come out. Unfortunately, Katz demanded that part of the settlement terms be kept under court seal.

Some of the terms are public: PKWARE cannot distribute the program after January 1, 1989; they cannot substantially change the program (though they can make bug fixes); and if they receive inquiries for the product, they must send out SEA literature. Also, PKWARE is prohibited from creating a new program that is compatible with ARC or PKARC.

If those terms sound one-sided, then it only goes to prove the extent to which PKWARE felt that it had no legitimate right to its program. After all, why would he give up everything if he was right? He obviously was not aboveboard in this case, even though the

settlement terms said he was not admitting fault in any way (a standard legalese ploy).

Now we are faced with several problems.

The bulletin board community has heard many comments by people who did not possess the facts of this case, and therefore made ill-informed opinions.

You are well aware that no party in a legal action can really speak his mind while the action is occurring. Because we didn't respond, people assumed that we were wrong.

Well, we weren't wrong and we won't be silent anymore. We realize that people came to the only conclusion possible, given the kind of information they received. We will respond to any and every comment about this case. We welcome questions and urge people to call us at our office.

We'd also like to clear up a few basic misconceptions that have appeared on the boards:

•SEA waited too long to take action: The legal world moves slowly. First we had to be aware of the situation, determine that a violation of our copyright existed, try to settle amicably, and then take legal action.

•SEA is a Goliath pursuing a David called Phil Katz: Phil Katz is not just a person. He is a company, and a big one at that. We calculate that PKWARE currently grosses almost $2,000 a day, or about five times what we do. We challenge him to make his audited figures public.

•PKWARE must be a small company because we hear there are only three employees, including his mother: We, too, are a family run company. Andy Foray and Thom Henderson are brothers-in-law and Irene Henderson serves as secretary/treasurer. We have hired a programmer and a license manager. We didn't do this because we had a windfall profit; we did this because we needed to stay competitive and to serve new markets.

•SEA should have pursued the case to a jury trial so a precedent could be set for the industry: We wish we'd had the money to support our lawyer to take this case to a jury trial conclusion. However, we needed to end it before the legal fees devoured us. Besides, we weren't out to crucify the guy — we just wanted him to stop stealing our work.

We have also issued a new policy statement regarding the licensing of ARC. It has been uploaded to the IBMSW forum on CompuServe, the utilities/archivers conference on BIX, and sent to other BBSes. The terms probably are the most liberal for any licensing policy for any software company. And if that isn't enough, give us a call and we'll see what we can work out.

Thom Henderson, President
System Enhancement Associates
voice: (201) 473-5153
data: (201) 473-1991

Greetings From Holland

I've been reading *Micro C* since Issue #39. Before that time, I had never heard about or seen your magazine in Holland. The magazine is a surprise compared with all the glossy issues normally exported from the U.S.A. to the rest of the world. Thank you very much for providing interesting articles without all the advertisements which make up 70% of the usual computer magazine.

Not only do you provide technical and software information, but the articles from Laine Stump show there's more in this world than U.S.-oriented computers and programming.

Take the Borland products for instance. If I want assistance, they tell me to join Compuserve. If I login to Compuserve from Holland, they charge me $100 per hour just for the phone. But recently things got better; Borland opened an information center in Paris. For an American, that's close to Holland. For me it means crossing two borders.

The articles about fractals are interesting. They let me make fine fractal pictures and made me learn a lot about fast algorithms, too. While your approach seems more hardware oriented (faster CPU, floating point units, and integer only algorithms), my friends and I are looking for smarter algorithms which will recognize points lying outside of the Mandelbrot set.

Peitgen, the German, had a book released a few months ago in which he suggests such an algorithm, claiming that it should reduce calculations by 40%! Unfortunately, we have not been able to make a running program.

In Issue #43 you wondered whether someone was really doing useful things with fractals instead of just looking at them. Since I am a student in geology and geophysics, I made some inquiries at my department. The result was as you would suspect. Nobody was really using them, though the fractal principles lead to larger understanding in meteorology and low temperature geochemical reactions.

Wim de Wilde
Turkooislaan 80
3523 GN Utrecht
The Netherlands

Editor's note: I checked with a known expert and reasonably famous person (Larry) and he said there definitely are practical applications for fractals. One of those applications is the search for practical applications. (He's been getting a lot more practical and a lot more creative since he began dividing his time between fractals and running.)

Fond Memories

OK, OK! I did it. Issue after issue of multi-megabytes of RAM and storage have finally driven me to purchase a clone. Will a McTek 286A really do anything more than my old faithful CP/M single board?

You sure make it tough on us old guys who at one time thought 8K of memory was the penultimate. There was a time, if I just had 4K more of memory I could make my program work. Gee, if I didn't ask what you wanted to do next I could squeeze it in. Only problem was I was the only person who could run the program. Oh well. I wrote it for myself, so who cares.

So let's see if pull-down menus, windows, zero wait states, and such are really better. So what if they aren't. It will be fun just seeing if I can get to know the system, like my old friend CP/M.

I read every page of *Micro C* even though I think Object-Oriented Programming is silly. But there was a time when I thought Pascal, C, and all other compilers were silly. Maybe someday they'll bring back delay-line memory.

James A. Shaffer
Allied Signal Aerospace
Navcamsmed Box 1115
FPO New York, NY 09554-7000

♦ ♦ ♦

Continued from page 4

blissful things like 20 MHz 386s or chocolate sodas.) But, if you're following your bliss then your work is your play.

The immediate question from the peanut gallery was:

"What's our bliss?"

"You already know, you've always known, you just have to listen." (Disappointed moment of heavy listening.)

During the discussion, I flashed back to the early days of *Micro C*. I had thought long and hard before deciding to start the magazine. Couldn't sleep nights. During the day I couldn't concentrate on my "real" work because my head was full of magazine.

As I've mentioned before, experts had told me the idea wouldn't work. The audience was too small and I didn't have time to add a magazine to my work and class load.

But I made time. (And thinking back, I had to make time because *Micro C* was my bliss — though I was probably the last to see it.)

Of course I was very practical. For me, the key was getting Digital Research Computers of Texas (shipper of the original Big Board) to give us their list of purchasers. After they agreed, Sandy and I got to work. Sandy produced a flyer about *Micro C* and I hauled thousands down to the West Coast Computer Faire. After the Faire I called DRC to find out why we hadn't received the labels.

They'd changed their minds. No labels. (And we'd only received two subscriptions from the Faire flyers.) Until now, I haven't understood why, at that point, I didn't return the two checks and find something else to occupy midnight to 5 a.m.

However, we continued and shortly thereafter Sandy and I got two huge breaks (breaks always come when you're doing what you should be doing). DRC called and, in lieu of labels, offered to send out our flyers with the Big Board orders. Plus, *BYTE* ran my review of the Big Board. Boy did things get crazy.

Our mailbox soon overflowed with subscription orders (I distinctly remember receiving eight in one day), requests for more information, and articles.

That was eight years ago. I strongly suspect, now, that *Micro C* happened because *Micro C* was my bliss, which also explains why I survived that first year without sleep and without pay.

Let Me Stress Something

So *Micro C* grew and prospered, not linearly, but in fits and starts. I was new to such subtle details as: financial management, personnel relations, marketing, and advertising sales. Every day, phone calls generated new questions forcing me to come up with new answers:

"No, you can't purchase our mailing list, we don't sell it."

"I don't know why we don't sell it, we just don't."

"Yes, I will speak to the management about that."

Now that's stressful. We still don't sell our list, though many times I've promised to talk to myself about it.

So, even in my bliss I felt really stressed. Of course, there's the obvious kind of stress. You know, the kind that chills the hands and rattles the heart. But there's the even more common, more insidious kind, noticeable only when it changes suddenly or someone points it out. It's the: sour stomach, uncomfortable but unsure why, irritated by the slow driver ahead, kind of stress.

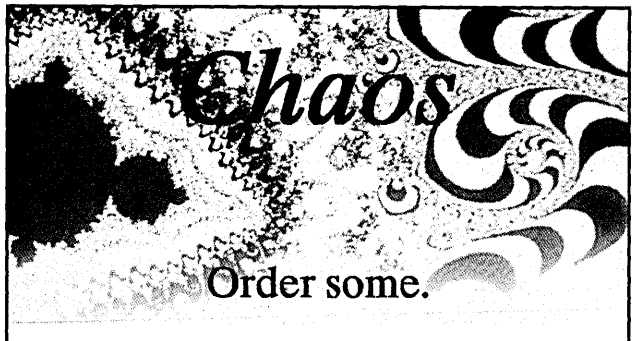
I suspect all of us drag the sneaky kind around whether we work in benches or in Bend.

Back To The Search

Anyway, I've been one of the lucky ones. I've found my bliss (or, it found me) and I've managed to follow it through some of the most interesting times in the computer industry. It began in computers, continued with writing, took off with airplanes, and, now, has settled into metaphysics/new age. (You know, I'm looking for the answer to life, the universe and everything, and despite what Larry says, I'm not sure it's 43.)

This has become a most interesting and wide-ranging search, much wider and deeper than I would have guessed. It's a trip (not unlike the antique air tour) that's created a lot of new ideas and feelings. I've been reading and experiencing things ranging from hard-edged dissertations on warps in the time/space continuum to the warm and fuzzy feel of psychic healing.

Over the past year, I've become very curious about all this — a neighbor worked up my astrological chart, predicting I'd write a book soon (the stars didn't know where I'd find the time) — a Tarot card reader predicted I'd write a book (the cards didn't come up with any free time either) — a channeling friend predicted I'd be talking about metaphysics in *Micro C* (nah, it'll never happen) — and, just recently, a friend of a friend gave me a psychic healing.



Mandelbrot Explorer 3.0

Fantastic fractal grapnics on 16-color EGA/VGA to 800x600. Magnifies up to 16.5 trillion times. Stop and start at will, save and retrieve, collage, full control over color boundaries, "zoom box," display of periodic orbits, auto-backup, all optimizations for speed including pixel interpolation and 386 integer support. Comes with seven ready-made pictures for immediate gratification.

$30

Peter Garrison
1613 Altivo Way
Los Angeles, CA 90026
213 665 1397

When ordering please specify EGA/VGA and disk format
Overseas orders please add $4

Reader Service Number 112

It Doesn't Feel Scientific

For a long time, I've had a problem understanding this strange new world. (There has to be some explanation, right?) I've also had a problem accepting something that's not repeatable. (Look, if it's real it's repeatable. That's the whole premise in any science lab.)

But there I was shuffling Tarot cards and handing them over to the reader. She then read my past, present, and future based on the order of the cards. If, at the end of the reading I'd shuffled them again, chances are very, very good that my past, present, and future would have read differently. (The repeatability problem.)

So it's chance, right? Two years ago, that reader (who didn't know me from Adam) told me some amazing things:

"The cards tell me that you're some kind of teacher, you explain things to lots of people."

"I edit a magazine."

"And it looks like you write quite a bit, too. You have a very dedicated, very loyal audience."

"Yes, it's a personal kind of computer journal in a plain brown wrapper."

"But I see it growing and getting fancier, looks like color. Do you have color now?"

"No, color goes against the grain."

"But it'll come, and soon. And it'll be a good change, there'll be a lot of excitement. You won't lose the personal touch so the change will be very positive.

"You'll also write a book. Sort of fiction, sort of real-life. It'll be a very successful book published by a large publisher. Don't worry about that yet, it will be many years before you begin."

It wasn't very many months before color started showing up in *Micro C*. However, the book still awaits its inspiration. (You and I both know that with a minor extension, this editorial would be about the right length.)

Of all her comments on my past, present, and future, 90% have been right on the mark. So far.

Support

I've found a network very similar to the *Micro C* network — people sharing ideas and energy. I've found folks very involved in their searching and researching. They're folks who have trouble explaining (at least to the uninitiated) what they're doing, but they're excited about it. They're breaking new ground and applying new technology.

Yep, new technology. Look at the similarities between the latest ideas in physics and some earlier ideas from metaphysics. You'll understand why theories about tunnels in the space/time continuum are bringing the two groups together.

Back To Earth

Meanwhile, I'd like more energy and joy in my life. That means serious work on my physical condition, my mental condition, and my diet. I'm not overweight, nor do I have any chronic medical problems, but my body is still due a little tender loving care. (After all, it's born the brunt of that stress.)

I'll be working with professionals on my diet, exercise, stress reduction, learning to be more flexible, more open, learning how to take risks without fear, and learning how to hear and trust my inner voice. In other words, I'm learning again how to live.

If you're interested in hearing about this trip, or if you have some suggestions about where I should look or with whom I should speak, let me know by mail (P.O. Box 223, Bend, Oregon 97709) or leave a message on the Micro C BBS (503-382-7643, 3-12-2400, 24 hrs.) or call 503-382-5060. If you're suggesting books, please include the publisher's name, address, and phone number.

Who knows, *Micro C* might be about people, too.

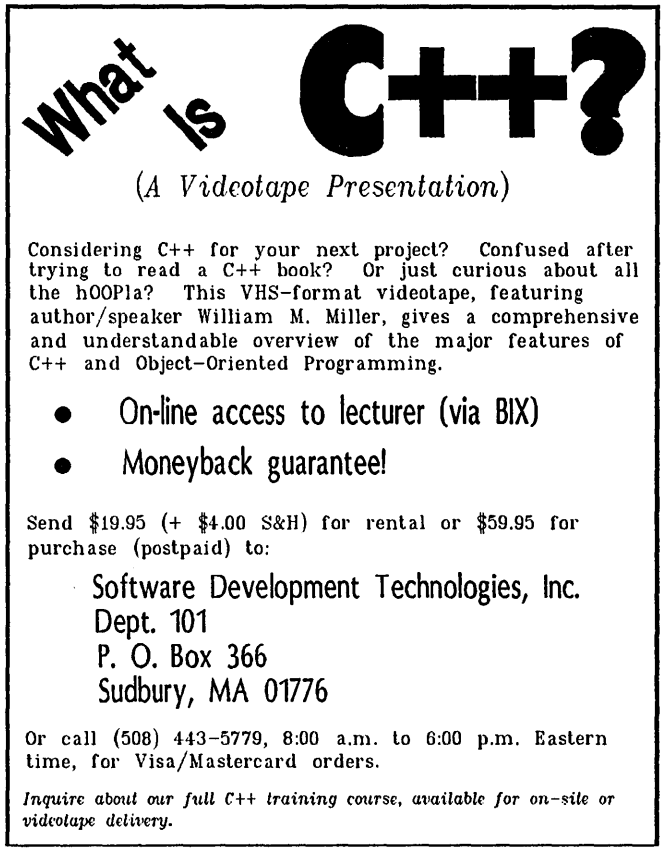
RAM Surcharges

You might be surprised when you pick up the phone to order the latest, greatest system at the latest, greatest price out of *Computer Shopper* or your local computer paper. It turns out that some shops are adding RAM surcharges because:

"We've been surprised how high RAM prices have (jumped, remained, gone) since we wrote the ad (one month, two months, a long time) ago. Not knowing exactly what to charge, we thought it best to give you our (rock bottom, very lowest, most competitive) price. No doubt the price will be (rising, going through the roof, doubling) when my (manager, boss, wife, girlfriend) finds out I'm giving these systems away."

But RAM prices have been pretty stable for nearly a year. In fact, they've declined a bit lately, and most dealers have figured out how to sell computers without jacking up the price at the end of the deal.

"Fine car, fine car. Best on the lot. Of course it'll be a little extra for the tires. Took 'em off my wife's car, myself. Broke



Reader Service Number 150

her heart. Just broke her heart. Her ridin' around on the drums like that."

Flash

If you've been perusing *Micro C*'s micro ads lately, you might have noticed "FLASH, The Disk Accelerator." Well, they sent along a copy and I stuck it into the system in my office. This package buffers disk reads (and writes, if you wish) in memory — any kind of memory. (Even the kind you paid extra for.)

Since mine is a barebones 640K system, I found that I couldn't allocate much for the buffer before my free space disappeared. Oh well, disk reads aren't much of a problem when you're writing short novels like this.

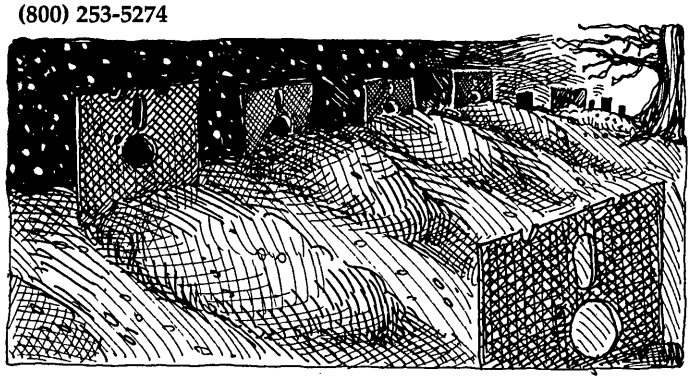
However, they threw in two other programs. One program is MAPIT, a small utility which tells you where TSRs lie mouldering in your machine and displays the interrupt table. Very nice.

But the other addition, FLASHKEY, has become indispensable. None of my keyboards repeats fast enough. Some do pretty well, others force me into manual mode (finger-tapping).

FLASHKEY fixes that. On callup you can set the speed, and the initial delay; it works with everything, and it only uses about 3K. (Let's see what MAPIT says.) Anyway, I can't imagine being without it.

Finally, the price is right. FLASHKEY sells for $19.95 (you only get the $19.95 price if you ask for it) including manual, MAPIT, and FLASH (the disk buffer).

FLASH $19.95
Software Masters
6352 North Guilford Ave.
Indianapolis, IN 46220
(800) 253-5274



The End Of User Groups

Today I received the latest copy of *Push & Pop*, the newsletter for the Sacramento Microcomputer Users Group (SMUG). It turns out it's Number 1 of Volume 13. More important, it's probably the final number of the final volume.

Apparently SMUG will be shutting down or merging with another group. After filling huge halls at meetings, after having the industry giants at their beck and call, after 13 years of leadership in this incredible technology, this venerable club has only 50 paid members.

One of SMUG's problems is that it stayed with CP/M, S100, and other (more and more) unique systems. But, I suspect there may be another problem. It's one facing all groups. There may be a diminishing need for user groups.

"Hey, hold on fella, there's more and more need for user groups, for the popular systems. Look at all these people getting their first Apples and PCs."

You have a point. But these aren't groupies. Outside of the kids, these are the last ones in. (Anyone really interested in computers would have gotten involved before now.) Sure, they need help, but they're getting that help from co-workers and family.

Computer clubs may soon go the way of automobile clubs. We learn to drive in high school. And, we're perfectly comfortable taking 50-mile trips without dragging along food, bedrolls, friends, mechanics, tools, and spare parts. The car clubs that remain support old Porsches, Chevies, and T-Birds. (And, of course, they travel around in groups and they carry tools...) So car clubs have been reduced to supporting the weird and irreplaceable.

Who knows, maybe SMUG's just a little ahead of its time.

Gary Entsminger Manipulates World Destiny

Every once in a while I get a letter that really opens my eyes, gives me a whole new perspective, helps me see the obvious. One such letter showed up yesterday.

Its author, who asked to remain anonymous, mentioned that I seemed apathetic, that *Micro C* was struggling, that the magazine's course had floundered:

"My subjective viewpoint is that this struggle started about the same time as Gary Entsminger... Not having met Gary, I

HANDS-ON PC REPAIR TRAINING

NAC offers the highest quality advanced 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. Network maintenance training is also available.

- Diagnose more quickly and accurately, complete repairs faster, and reduce costs.
- Discover better diagnostics, parts sources, and new trouble-shooting tools and test units.
- Toll-free technical help and parts-locator line, free technical and vendor update service.
- Used by GTE, ITT, Rockwell, Xerox, SoCal Edison, regional Bell companies, the U.S. Dept. of Commerce, and many others.

Our five-day PC Maintenance Course is offered in California, New York, Washington, Atlanta, Chicago, St. Louis, Dallas, Seattle, and other major cities. Call without obligation to get a course outline and dates for the class locations most convenient to you.

National Advancement Corporation
2730-J South Harbor
Santa Ana, CA 92704
(714) 754-7110
Computer Hardware Training Specialists

Reader Service Number 59

can only speculate from his writing that he is a strong personality that has pushed *Micro C* off its original course with brute force... I can only hope you'll leave the ibbem camp for something I'll look forward to."

Well, sure. Our first clone article (the do-it-yourself XT in issue #27) showed up just a year after Gary arrived. At that point CP/M was rolling, Kaypro was solvent, folks were excited about the 64180 and the Z800, Ciarcia was writing for *BYTE*, and RAM was cheap.

Now all that's changed.

Of course, if Gary hadn't infiltrated *Micro C*, things would undoubtedly have turned out very different. Digital Research would have announced CP/M 4.0 (a multi-user, multitasking, graphics and UNIX compatible operating system for $49.95), Kaypro Corp would be buying out a troubled IBM, Zilog would have announced its latest Z800 (a $5, 60 MHz, 64-bit, chip with on-board: 25 Mega-flop coprocessor, MMU, DMA, and clock with sweep second hand), *BYTE* would still be *BYTE*, and we'd be using 256K dynamics for packing material.

Amazing, the things I learn when I read my mail.

Miniscribe Takes A Dive

After hearing distributors and dealers wax eloquently about Miniscribes, I'm hearing another tune. Miniscribe is announcing a significant loss for the latest quarter and rumor has it the financial problem has been caused by one of their 30 meg drives. (They're taking a major write-down so I'm guessing they have a batch in the warehouse.)

I've also heard some disgruntled comments about their turnaround time on warrantee repairs. More particulars as they come in.

Meanwhile, I talked recently to Microscience. Great talk. The guy spoke about the problems they were having with the mix of binder and lubricant for coating their platters. The supplier was adding too much lubricant so the surface was getting sticky.

And that was after they'd gotten warrantee returns down to 0.5% for the unit. (That's probably one-tenth the industry standard.)

I'm glad I'm not building hard drives for a living.

Mr Mox And The Timer

I know you'd love to have your system auto-connect with compuserve at 3 a.m. to download your messages, but you're not sure you want to leave your system on all night (or over the weekend).

Well, Mr Mox, the nifty little power strip that turns itself on when someone calls your modem, can also turn itself on at a preset time. Also, your system can tell whether it was turned on at the preset time or by a call. So you can have your system turn itself on and answer a call from another modem, or turn itself on at a preset time and day and access someone's BBS, or compuserve, or a Mac, or whatever.

In fact, there's no reason you couldn't preprogram the

ASMFLOW $99.95

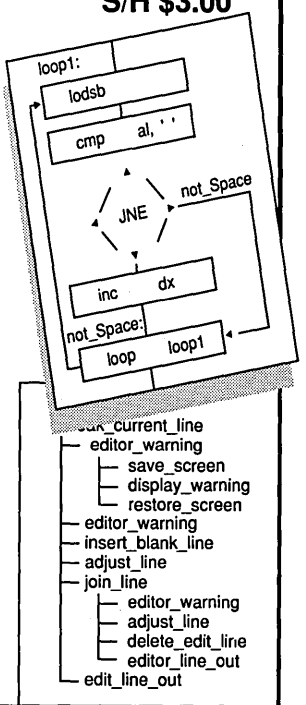
S/H $3.00

AT LAST!

YOU CAN STEP BACK AND TAKE A LOOK AT YOUR CODE

FLOW CHART AND ANALYZE YOUR ASSEMBLY LANGUAGE SOURCE CODE

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



QUANTUM SOFTWARE
19855 Stevens Creek Blvd, Suite 154
Cupertino, CA 95014
(408) 244-6826 – Free Demo

VISA • MC 30 - Day Money Back Guarantee

Reader Service Number 139

Logic Gem $198

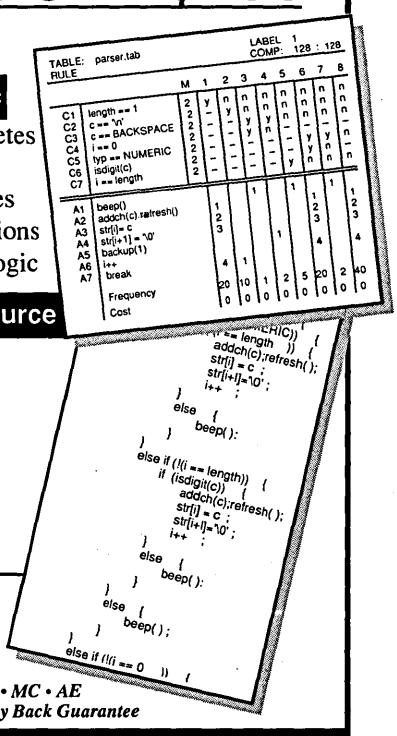
AT LAST!

Develop perfect logic

- Automatically completes incomplete logic
- Removes redundancies
- Eliminates contradictions
- Sorts and optimizes logic

Generate bug-free source

- C
- Pascal
- dBASE
- Fortran
- English
- Structured Basic
- Interpretive Basic



Sterling Castle
(800)722-7853
(800)323-6406 CA

VISA • MC • AE
90 Day Money Back Guarantee

Download demo • (213) 453-7705 • 3/12/24,8,N,1

Reader Service Number 141

power strip to turn on at a certain time and day, or even right away, and have it control the power to some device other than the computer. (Greenhouse lights, ventilating fan...)

Let's say you set on and off times. These get translated into the number of ticks of the power strip's clock and then get sent to the strip. The power strip simply counts down to the "on" time, turns on its outlets, and starts counting down to the "off" time (if you've set one).

The only thing that'll keep the power strip from carrying out its task is a power failure. If the power goes away, even for a short while, the strip loses its programming.

For $99.95, you get the plain Mr Mox (turns on the power when it sees a carrier detect from your modem). For $139.95, Mr Mox comes with the clock and software to set it. The software keeps track of which timer event triggered the power-up so the computer'll do the right thing (like call your mother at 3 a.m. and whistle in her ear).

If you want to try out the auto answer modem, call (619) 481-1753 and you'll be talking to Jay Bowden's system. (You'll get a carrier right away, but it'll take about 60 seconds for the computer to come up, load the BBS, and respond.)

Reformatting XTs

A lot of folks are ordering SpinRite (or the latest version of Disk Technician) to optimize the interleaves on their XTs. (I did.) Both programs check to see if there's a difference between the optimum interleave and the current one. If there is, they'll offer to do a nondestructive low level format using the optimum interleave.

However, there's a problem built into the XT's controller. When either program does a nondestructive format, it reads the original track, then tells the drive controller to format that track. When an XT controller receives a single-track format command it homes the head, then steps it back out to the track and writes the format information. (Following the format, the program writes the data back onto the track.)

So, formatting an XT drive this way is a slow process and those long seeks cause head position errors on most drives. (Drives with an odd number of heads have an additional head and disk surface dedicated to detecting head position. They don't have the problem.)

That means you will probably have more errors after a nondestructive low level format than before. So, if you need to completely reformat your hard drive, take the data off and do a destructive low level format (followed by FDISK and the DOS FORMAT).

AT controllers, on the other hand, don't have the problem. They simply step the drive to the track that'll be formatted — no homing and no problem. A nondestructive format is as good as the destructive version.

So these programs work wonderfully on ATs. On XTs, however, I'd limit them to checking for optimum interleave and reporting data errors, and maybe reformatting a bad track or two. There's no way I'd let either program reformat an XT's whole drive.

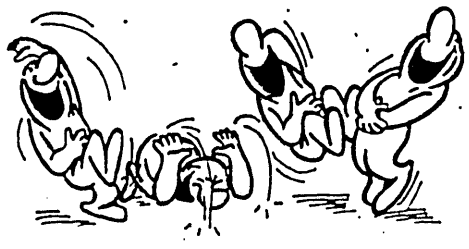
Haunted By My Indiscretions

It's fun reading about the dalliances of our esteemed leaders, but it's not so funny when it's closer to home. For instance, I logged onto the Micro C bulletin board last evening, and there it was again:

"Beep!"
Please enter your registration code: _____
"Beep!"
Please enter your registration code: _____
"Beep!"
Please enter your registration code: _____

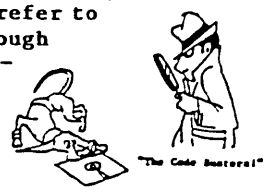
Boy, when you start a "Personal Relationship," you don't know where it's going to stop.

David J. Thompson
Unregistered Shareware User



YOU WANT THE SOURCE?!

WELL NOW YOU CAN HAVE IT! The MASTERFUL DISASSEMBLER (MD86) will create MASM compatible source code from program files (EXE or COM). And the files are labeled and commented so they become USEABLE. MD86 is an interactive disassembler with an easy to use, word processor like interface (this is crucial for the REAL programs you want to disassemble). With its built-in help screens you won't have to constantly refer to the manual either (although there are valuable discussions on the ins and outs of disassembling which you won't want to miss).



MD86 is a professionally supported product and yet costs no more than "shareware". And of course, it's not copy protected. VERSION 2 NOW AVAILABLE!

MD86 V2 is ONLY $67.50 ($1.50 s&h) + tax

C.C. Software, 1907 Alvarado Ave., Walnut Creek, CA 94596, (415) 939-8153

Reader Service Number 31

Pick A SOG VIII, 1989

The Regionals

No more excuses — you've got to attend a SOG. This year we have four SOGs on tap. One on the west coast, another on the east coast, one in the Lone Star State, and yet another in Colorado (wherever that is).

You have your choice of the BC SOG, all green, green, green, with saltwater, fishing and logging, on Vancouver Island. Or the Rocky Mountain SOG, in 7,700 foot elevation Gunnison, Colorado. This recreation community is surrounded on three sides by serious snowcapped mountains. Or how about SOG East? Mid-September is a beautiful time in southeastern Pennsylvania. Finally, there's the Long Horn SOG; October is a fine time of year for dodging cow pies in North Texas.

COLORADO

Rocky Mountain SOG

What: *Rocky Mountain SOG*
When: *July 28-30, 1989*
Where: *Gunnison, Colorado*
(Aspinall-Wilson Conf. Ctr)

Rocky Mountain SOG (RM SOG) kicks off on Thursday the 28th with a traditional SOGgy favorite: white-water rafting. Friday and Saturday will be filled with workshops and presentations, followed by a banquet Saturday night. (Scott's arranged an all-night room for the Jolt SIG.)

Gunnison, population 6,500, lies nestled in the Rockies along the Gunnison River. Besides white-water rafting, you'll find hiking, camping, fishing, and general exploring. Gunnison County has some of the most beautiful wilderness in the country.

RM SOG will be held during the same weekend as nearby Crested Butte's Aerial Weekend, which features parachuting, hang-gliding, and hot-air balloon rides and races. This will be a family vacation as well as an educational experience.

When you register, they'll send you a packet. In that packet you'll find information on activities ranging from tours of the Black Canyon to ghost town excursions. You'll also receive hotel and airport information. Both Continental and United Airlines service Gunnison's airport.

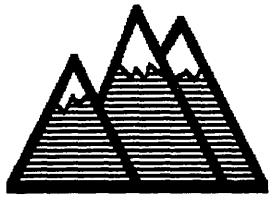
SPEAKERS NEEDED! If you have something you would like to present, feel free to contact them! Speakers will get free admission and a free tee-shirt!

The Keynote speaker for the Saturday Night Banquet will be Walter Bright, author of Zortech's C++.

Prices:
Conference registration: $25 in advance, $30 at the door
RM SOG Tee-Shirt $10 (specify size)
1/2 day raft trip $35
Full-day raft trip $50
Saturday night Banquet $15
Thursday night Barbeque $10
Cafeteria Lunches (Fri/Sat) $10 ($5 per lunch)

Registration Info: Please register as soon as possible. If you preregister, please do so before July 1. Make checks payable to "Rocky Mountain SOG." Include the names, addresses, and phone numbers of everyone you're paying for, and list what each is ordering.

Contact: *Rocky Mountain SOG*
302 N. 12th Street
Gunnison, CO 81230
(303) 641-6438



CANADA

BC SOG

What: BC SOG
When: July 7 & 8, 1989
Where: Port Alberni, British Columbia

If SOG's your attempt to get away from it all, then the BC SOG is your spot. Vancouver Island's Port Alberni is about as far from anywhere as you're going to get by car, plane, or boat. (195 highway miles northwest of Victoria, British Columbia.)

This lumbering and fishing community of 18,500 lies at the end of a fjord-like saltwater inlet. Famous for its salmon fishing, its sheltered islands and inlets, this is a popular area for kayaking, canoe camping, boating, and just about anything else connected with saltwater, freshwater, or coastal mountains. (If you aren't planning to take your whole family, don't let them see the pamphlets you'll receive in your registration packet.)

The community sports 12 motels, 6 bed and breakfasts, and a mess of campsites and trailer spaces. (It's all in the packet.)

They're scheduling talks on Friday and Saturday, July 7 and 8, at North Island College. The facility will open Thursday evening, the 6th, for early arrivals. They'll also have a space for the traditional all-night technical forums (The Jolt SIG).

Technical events include: a planned mini-maze competition put on by the Seattle Robotics Society (this is tentative as we go to press) and a fresh salmon barbeque scheduled for Friday evening.

Sponsors David Stern and Randy Young ask that you call them or drop them a card by June 15. (It's not mandatory that you preregister, it just makes their planning a lot easier.)

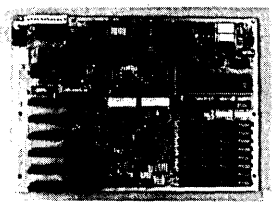
If you'd like to speak, definitely get in touch. They're actively looking for you.

If you're a vendor, swapper, or hacker, bring your wares and spares. The display and meeting area will be open from Thursday evening through Saturday evening. 8' tables are only $10.00 each.

Prices: *Registration $10 (per family)*
BC SOG T-Shirt $10 (Specify size)
Salmon Barbeque $10 ($5.00 12 and under)
Display Table $10
(All prices in Canadian dollars.)

Contact: *David Stern*
4403 8th Ave.
Port Alberni, BC
Canada V9Y 4S6
(604) 723-5917
or:
(604) 724-2019 (Randy Young)

68000 SINGLE BOARD COMPUTER



- 512/1024K DRAM
- 4 RS-232 SERIAL PORTS
- FLOPPY DISK CONTROLLER
- REAL TIME CLOCK

BASIC KIT(8 MHZ) - BOARD, MICROPROCESSOR, HUMBUG MONITOR/ BASIC IN ROM, 4K SRAM, 2 SERIAL PORTS $200

PERIPHERAL TECHNOLOGY PROVIDES ACCESSORIES TO BUILD COMPLETE SYSTEMS!

PACKAGE DEAL - COMPLETE KIT WITH 10 MHZ MICROPROCESSOR, SK*DOS OPERATING SYSTEM, 512K DRAM $575

SYSTEM BOARD(12MHZ) - ASSEMBLED/TESTED, 1MEG RAM, 6 PC/XT PERIPHERAL PORTS, SK*DOS $899

COMPLETE INFORMATION AVAILABLE UPON REQUEST

PERIPHERAL TECHNOLOGY
1710 CUMBERLAND POINT RD., #8
MARIETTA, GA 30067 **404/984-0742**

COD/MASTER CARD/VISA/CHECK
SK*DOS IS A TRADEMARK OF STAR-K SOFTWARE SYSTEMS CORP.

Reader Service Number 119

TEXAS

LONGHORN SOG

What: Long Horn SOG
When: Oct 13 - 15, 1989
Where: Dallas, Texas (INFO MART)

More information next issue, but this will be a big one. This computer club has over 4,000 members in some 15 user groups (over 90 sessions at each meeting). Plus, Dallas weather should be marvelous this time of year. So plan to come and enjoy the southern sun and the Texas hospitality. If you'd like to speak, you should contact Stuart immediately.

Contact: *Stuart Yarus*
The Computer Council Of Dallas
1950 Stemmons Fwy., Box 277
Dallas, TX 75207
(214) 867-8012 (evenings)

PENNSYLVANIA

SOG East

What: *SOGeast*
When: *September 7 & 8, 1989*
Where: *York, Pennsylvania*

Introducing SOGeast — the Eastest SOG yet! Nestled in York, Pennsylvania, this SOG will bring the best of the West to the East.

York is in the heart of the Pennsylvania outlet district. Those who love bargains will have their hands (sacks, buckets, wheelbarrows) full. Also you'll find: antiques, down-home Penn Dutch cooking, crafts, and much more. Historic Gettysburg is only 30 minutes away.

York is 25 minutes from Harrisburg airport, and about 60 miles from Baltimore/Washington International.

Tentative schedule:

Wednesday Night - (4 p.m. til about 9 p.m.) Twilighter Railroad Excursion - Steam train ride and dinner trip, $25. (Special trip just for SOG attendees and families.)

Thursday Daytime - Meetings, talks, etc.

Thursday Supper - Volleyball, frisbee, barbeque (Penn Dutch style) picnic (outdoor). $15 per person.

Thursday Evening - Rooms available all night for talking, speaking, Jolt SIG, boxing, etc.

Friday Daytime - More of the same.

Also On Friday - York County Fair begins (runs nine days). Largest livestock and agricultural show on the East Coast. Lots of food, free show entertainment, big name entertainment on the grandstand, and much more.

Saturday Morning - For those who can't make the Wednesday train ride, there'll be another on Saturday morning. (The Saturday trip is open to the public, so you'll need reservations).

They now have three conference rooms (hold 300 total when all are connected, or about 100 a piece). They will have room for exhibit tables, so bring your stuff. More on other activities will follow. They've scheduled these rooms at a local hotel, but they're still jawing with the local colleges. Either way, we'll have a place!

They'd like verbal registrations as soon as possible so they can reserve the train, rooms, etc. Please get your money to them by August 1. Also, speakers and exhibitors (tables $35 each), please call right away. Free admission for speakers (call early so they can schedule you).

Registrants will receive lodging and transportation information by return mail. Additional local information will be awaiting your arrival. Make checks payable to "The CDS Group."

Prices:
SOGeast Registration $25
SOGeast Tee-Shirt $10 (specify size)
WednesdayTrain Trip & Dinner $25
Thursday Night Picnic $15

Contact: *John Ribar*
The CDS Group
3161 Honey Run Dr.
York, PA 17404
(717) 792-5108 (8-12 p.m. Eastern time)
(717) 854-3861 (Weekdays 8-9 a.m., 4:30-5:30 p.m. eastern time)
(you can also talk to Carol Park at this number)
Compuserve 73577,1652

Last Minute Addition!!!
North Cal SOG

Terry Sherb called right before magazine deadline to say he is working on a North Cal SOG. Here's the tentative information:

What: North Cal SOG
When: August 25, 26 ,27
Where: UC Davis Campus, Davis, California
Anyone wanting to speak or help, contact:
Terry Sherb (916) 927-8745 (evenings)
A lot more information next issue.

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN for MARCH 5, 1989

OUTSIDE OKLAHOMA: NO SALES TAX

DYNAMIC RAM				
SIMM (1)	256Kx36	80 ns	$600.00	(3) STATIC COLUMN DRAM
SIMM (2)	1Mx9	80 ns	370.00	
SIMM (2)	256Kx1	60 ns	140.00	
1Mbit	1Mx1	100 ns	23.50	
41256	256Kx1	60 ns	11.25	
41256	256Kx1	80 ns	11.00	(4) 2-PORT VIDEO RAM
41256	256Kx1	100 ns	10.85	
51258 (3)	256Kx1	100 ns	11.95	
41256	256Kx1	120 ns	8.75	
41264 (4)	64Kx4	120 ns	14.50	
EPROM				
27C1000	128Kx8	200 ns	$28.50	80387-16 80387-20
27C512	64Kx8	200 ns	13.95	
27256	32Kx8	150 ns	8.15	80287-8 $230.00
27128	16Kx8	250 ns	4.95	
STATIC RAM				
62256P-10	32Kx8	70 ns	$24.50	80C87-2 $145.00
6264P-12	8Kx8	120 ns	9.75	
6116AP-12	2Kx8	120 ns	5.50	

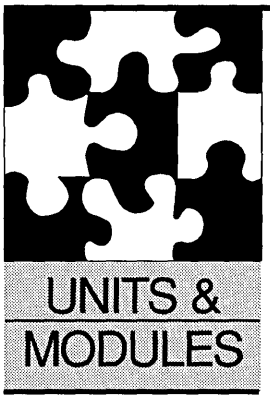
(1) PS2/70 OR PS2/80 1Mbyte
(2) FOR: AST AT&T COMPAQ DELL MACS MYLEX PS/2 Tandy WYSE WD

OPEN 6½ DAYS, 7:30 AM-10 PM: SHIP VIA FED-EX ON SAT.
WE EXPORT ONLY TO CANADA, GUAM, PUERTO RICO & VIRGIN ISLANDS

SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY:
Thr Std Air $6/3 lb
Fr: P-1 $12.25/1 lb

MasterCard/VISA or UPS CASH COD
MICROPROCESSORS UNLIMITED, INC.
24,000 S. Peoria Ave., BEGGS, OK. 74421 (918) 267-4961
No minimum order. Please note: prices subject to change! Shipping, insurance extra, up to $1 for packing materials.

Reader Service Number 37



Seven Compilers And A Book

By Michael S. Hunt
845 E. Wyeth
Pocatello, ID 83201
(208) 233-7539

Michael takes on Scott and finds producing compiler reviews isn't all that fast. (It takes time to compile and assemble the data.) So, in the spirit of pure research (and long lunches), the Micro C staff has begun linking together data on a whole new field. The object is to benchmark the reviewers.

This issue I'm reviewing seven compilers and a book. Writing the column has been the easiest part. If you take seven compilers times seven benchmarks times five compiles and five links and five runs, you get... well, it's not sanity. I do want to thank my good friend Lyle Carson for his help with, and use of, his Amiga computer.

First, The Book

I own one Modula-2 book and have read five others, but *Modula-2: A Complete Guide* is the best one yet. The book can be used as a text for classroom, self-study, or as a reference to the language. It contains 656 pages of examples, explanations, and reference material.

Dr. King's book is very thorough. It covers everything from the basics to advanced concepts. D.C. Heath published the book and Karen Ellison, of Jensen & Partners International, edited it. JPI uses an extremely condensed version of the book as the language tutorial for their TopSpeed Modula-2 package.

If you buy the JPI compiler, you can get the book for only $21. Or, you can order it through most bookstores or directly from D.C. Heath for $33.33. Dr. King will sell you a disk of source for $10.

Second, The Compilers

Before I get into the good stuff, I probably should mention a source of troubles I had. They aren't complaints, they're just the kinds of problems you might encounter when working with a new compiler.

For instance, I had a problem getting the Logitech programs to run. The programs just locked up the computer. No error messages because I had removed checks to optimize execution speed. After much fussing around, I remembered that Logitech's default stack is just 8K. I relinked the problem programs with a larger stack and they worked.

The Benchmarks

Benchmarks, if nothing else, are controversial. I have selected seven benchmarks (Figure 1) that I hope test compiler performance. Each benchmark time is an average of five runs, links, or compiles. I used a calling program that reads the system clock before and after the test. You'll find all the benchmark code on the Micro C BBS or Issue #47 disk.

I ran the MS-DOS non-8087 benchmarks, compiles, and links on a PC Tech X16B with an 8 MHz 80186, 1 MB no-wait-state memory, and a 61 ms 1 to 1 interleave SCSI 33 MB hard disk. The execution times for the MS-DOS 8087 version of the Whetston were run on a Leading Edge Model D with a 7.16 MHz 8088 & 8087 and two floppy drives. The Avant Garde

Figure 1 — Benchmarks to Test Compiler Performance

```
Sieve      Array indexing and integer arithmetic test.
MemTest    Memory allocation, deallocation & pointer handling test.
FileIO     File handling efficiency test.
Whetston   A floating point, integer and transcendental test.
Dhryston   Contains a mix of operations that emulates a typical program.
QuikSort   QuickSort algorithm, tests recursion and array indexing.
CQuikSrt   Same as QuikSort but uses coroutines to partition the array.
Empty      Empty program.
```

♦ ♦ ♦

compiler was run on an Amiga 1000 with 512K and two floppy drives. The CP/M code ran on the X16B with a Micro Interfaces 5 MHz Z80 coprocessor board.

Stony Brook Modula-2 Compiler

This package includes a command-line compiler, text editor, source level debugger, make utility, library source, and PMI's Repertoire toolkit (in object form). If you call the compiler from the editor, compiler errors will bump you back into the editor with the cursor on the error and the screen displaying an appropriate (we hope) error message.

You can also execute DOS commands from the editor (including your linker) and it lets you work on 15 files at a time (but it displays only two). Stony Brook refers to this package as the development system (in lower case), so I guess that could be used to distinguish it from QuickMod. The development system will soon be upgraded to include QuickMod's library and environment.

I found three major reasons someone would buy this package over their QuickMod.

Figure 2 — Average Compile & Link Times

	LogiTech	JPI	StBrook Mod-2	StBrook QuickMod	FTL LMM	FTL CP/M	TPascal 3.0	Avant Garde
Sieve								
Compile	6.01	4.45	3.82	0.83	2.29	4.17	2.19	3.6
Link	7.20	3.89	4.47	1.26	4.27	5.60	0.00	4.8
Total	13.21	8.34	8.29	2.09	6.56	9.77	2.19	8.4
MemTest								
Compile	7.37	5.37	4.20	1.15	2.75	4.91	2.15	2.8
Link	12.72	4.78	5.11	1.66	7.38	9.01	0.00	12.1
Total	20.09	10.15	9.31	2.81	10.13	13.92	2.15	14.9
FileIO								
Compile	10.14	6.92	4.45	1.33	3.14	5.63	2.23	5.1
Link	16.86	5.88	5.77	1.92	10.66	17.57	0.00	8.2
Total	27.00	12.80	10.22	3.25	13.80	23.20	2.23	13.3
Whetston								
Compile	19.83	11.04	9.45	4.84	3.85	7.73	4.77	9.9
Link	15.27	5.16	6.57	2.27	26.08	25.43	0.00	13.9
Total	35.10	16.20	16.02	7.11	29.93	33.16	4.77	23.8
Dhryston								
Compile	34.25	22.15	17.93	5.13	12.51	18.21	4.53	32.4
Link	16.43	6.37	5.88	1.06	9.77	11.63	0.00	22.3
Total	50.68	28.52	23.81	6.19	22.28	29.84	4.53	54.7
QuikSort								
Compile	7.26	5.19	4.23	1.00	2.47	4.63	2.52	3.8
Link	7.48	3.90	4.56	1.52	4.36	5.73	0.00	8.1
Total	14.74	9.09	8.79	2.52	6.83	10.36	2.52	11.9
CQuikSrt								
Compile	8.41	5.69	4.46	1.12	3.48	7.16	NA	6.7
Link	8.07	4.23	4.80	1.54	11.25	11.77	NA	11.4
Total	16.48	9.92	9.26	2.66	14.73	18.93	NA	18.1
Empty								
Compile	5.52	4.26	3.04	0.48	2.14	2.95	1.62	1.2
Link	7.36	3.93	4.21	1.34	4.18	4.83	0.00	1.1
Total	12.88	8.19	7.25	1.82	6.32	7.78	1.62	2.3

♦ ♦ ♦

Figure 3 — Code Size

	LogiTech	JPI	StBrook Mod-2	StBrook QuickMod	FTL LMM	FTL CP/M	TPascal 3.0	Avant Garde
Sieve	3646	1717	856	882	3584	2048	8448	1372
MemTest	5636	2879	3170	4629	7680	3072	8448	1976
FileIO	14773	4901	6770	9334	10752	10240	8576	2200
Whetston	10123	13759	11242	14074	34304	13312	11648	4816
Dhryston	4428	3821	5590	7642	9728	5120	10112	10400
QuikSort	3839	1895	1044	1090	3584	2058	8832	5600
CQuikSrt	4428	2901	1712	1699	11776	5120	N/A	21916
Empty	3501	1611	750	770	3584	2048	8192	?

♦ ♦ ♦

First, this compiler produces object modules that comply with the Microsoft subset of the INTEL object module format. The calling conventions are identical to Microsoft Pascal (also the standard call interface for Microsoft Windows and OS/2).

Second, the objects produced are not DOS specific and with the necessary modifications to the runtime system (you get source), programs can run on any 80x86 based hardware.

Third, though the package doesn't have a slick user interface like QuickMod and TopSpeed, it does have a powerful and flexible set of keywords that lets you combine Modula-2 code with object output from almost any other language.

Stony Brook's development system is lean and mean. This flagship version stole the show in compile and total compile/link times (see Figure 2). It also garnered half the awards for fastest execution time. (TopSpeed took the other half.)

Unfortunately, the package does not include a linker (you must supply your own). And, though the manual is very thorough, it doesn't attempt to teach you Modula-2.

The code sizes are among the smallest (see Figure 3) and compare favorably with JPI's TopSpeed. The code sizes could have benefited from an intelligent linker. I used the Microsoft Object Linker 3.05.

Stony Brook's Other Compiler

QuickMod has a slick integrated environment comparable to TopSpeed's. The QuickMod Editor (basically the same as the Modula-2 editor) lets you reassign the editor's function keys to suit your taste. It also gives you ALT-key access to the debugger, compiler, linker, environment and compiler options, program execution, and library maintenance. This environment also includes a make so that any compiling or linking is done automatically.

QuickMod stores all object code in libraries. Libraries can be linked and modules added and deleted whenever you wish. Unfortunately, QuickMod's object code is stored in a proprietary format.

You can, however, import Microsoft object modules. The module is declared Foreign and is assumed to have been written for the Microsoft Assembler. (The manual contains a 15-page example on interfacing with Assembly language.)

The QuickMod library contains 30

modules and almost 200 procedures. This includes support for graphics, communications, mouse, and time-slice process control.

QuickMod has a very good context-sensitive help system. The debugger is powerful, though not as easy to use as the ones supplied by Logitech and TopSpeed. For instance, you operate it via command strings instead of control-key sequences.

The Stony Brook compilers are the most Wirth compatible of the compilers reviewed. In fact, the library included with the cheaper QuickMod library is a superset of the development system's. Stony Brook will soon be offering a full money-back guarantee. If you don't like their package, you just return it.

I timed QuickMod's compile and link times from within the environment. (QuickMod is the only compiler tested which does not support command-line execution.) To get a fair idea of the true load/compile/link times, add six seconds to the total compile and link times to cover program loading.

Stony Brook Miscellaneous

Both Stony Brook compilers include good manuals which contain a language reference (not a tutorial) and a runtime library reference. Each compiler comes on two disks and each will run on a dual floppy system. Installation is easy.

Stony Brook has just moved to California and will release version 2.0 of both compilers in May-June 1989. Version 2.0 will bring more optimization and the QuickMod environment will be added to the Modula-2 compiler.

QuickMod will be available in DOS and OS/2 versions for $95. Professional Mod will include DOS and OS/2 versions of the Modula-2 compiler and both QuickMod compilers will sell for $299. The runtime source will be unbundled and available for $150.

JPI TopSpeed Modula-2

This is the compiler everybody's raving about. I'm not going to rave, but TopSpeed is a first class product. It has fast compile and link times, an excellent environment, a great optional toolkit and debugger, almost everything you could want.

TopSpeed comes with a nice windowing editor, compiler, linker, library source, automatic make, and smart linking. The definition files are always recompiled so the implementation files can be compiled in any order. (This is a neat feature much like QuickMod's automake.)

Figure 4 — Modula-2 Compiler Features

	LogiTech	JPI	StBrook Mod-2	StBrook QuickMod	FTL LMM	FTL CP/M	TPascal 3.0	Avant Garde
Compiler								
186/286	Yes	No	No	No	Yes	N/A	N/A	N/A
8087	Yes	Yes	Yes	Yes	Yes	N/A	N/A	N/A
Linker	Smart	Integrated	DOS	Integrated	Incl.	Incl.	Automatic	Incl.
Debugger	Yes	Yes/opt	Yes	Yes	Opt.%	Opt.%	No	No
Librarian	No	Integrated	No	Integrated	Yes	Yes	No	No
Make	Yes	Integrated	Yes	Integrated	Opt.%	Opt.%	N/A	No
Editor	Point	Wordstar	Brief	Brief	Wordstar	Wordstar	Wordstar	Emacs
System								
Version	3.03	1.12	1.14	1.00	1.17	1.31	3.0	1.00
Memory	512K$	384K	&	512K	256K	58K	?	512K
Floppies	2	2	2	2	2	2	1	2
Harddisk	Recom.	Recom.	Recom.	Recom.	N/R	N/R	N/R	Recom.

$ 290K for overlay version not tested here.
% Included in the Editor Toolkit.
& No minimum memory requirement given, but the compiler is 151K in size.

♦ ♦ ♦

Figure 5 — Average Run Times

	LogiTech	JPI	StBrook Mod-2	StBrook QuickMod	FTL LMM	FTL CP/M	TPascal 3.0	Avant Garde
Sieve	78.18	34.87	46.52	47.46	97.75	416.71	381.93	102.8
MemTest	81.40	55.53	49.93	54.65	253.93	103.19	70.25	79.2
FileIO	20.44@	72.23	72.08	73.86	72.56	283.36	111.87	141.8
Whetston								
wo/8087	44.00	71.84#	25.05	24.90	151.27	381.36	172.87	13.6
w/8087	10.72	7.01	7.67	8.26	17.52	N/A	N/A	N/A
Dhryston	56.23	44.30	46.09	48.92	52.29	215.23	176.91	82.8
QuikSort	32.79	21.29	30.30	33.10	52.29	163.09	201.41	43.1
CQuikSrt	38.34	36.72	33.01	41.63	36.91	108.95	N/A	41.8

@ Logitech uses buffered IO for block reads and writes.
# JPI's Mathlib0 uses LONGREALs instead of REALs.

♦ ♦ ♦

Figure 6 — List of Vendors and Products

The following is a list of the vendors and products reviewed in this issue.

Alpha Systems
711 Chatsworth Place
San Jose, CA 95128
(408) 297-5594
Contact: Joe Wright

Products:	Turbo Pascal 3.0 (Borland)	CP/M-80	$60.00
	NZ-COM: The Automatic Z-System	CP/M-80	$69.95

Workman & Associates
1925 East Mountain Street
Pasadena, CA 91104
Voice (818) 791-7979
BBS (818) 791-1013

Products:	FTL Modula-2 Compiler	ver. 1.30	CP/M-80	$49.95
	Advanced Programmers Kit		CP/M-80	$39.95
	Editor/Toolkit	ver. 1.30	CP/M-80	$49.95
	CP/M Package (all three)			$99.95
	FTL LMM Modula-2 Compiler	ver. 1.18	MSDOS	$79.95
	Advanced Programmers Kit	ver. 1.16	MSDOS	$39.95
	Editor/Toolkit	ver. 1.12	MSDOS	$49.95

```
          MS-DOS (all three)                              $129.95
          FTL SMM Modula-2 Compiler ver. 1.31  MSDOS   $49.95

Jensen & Partners International
1101 San Antonio Road, Suite 301
Mountain View, CA 94043
(415) 967-3200 Fax: (415) 967-3288
Contact: Karin Ellison, VP Marketing  (800) 543-5202

Products:  TopSpeed Modula-2           ver. 1.12  MSDOS   $99.95
           Visual Interactive Debugger ver. 1.01  MSDOS   $59.95
           Techkit                     ver. 1.12  MSDOS   $59.95
           All three above                                $199.95

Stony Brook Software, Inc.
187 E. Wilbur, Suite 9
Thousand Oaks, CA 91360
(805) 496-5837 (in California)
(800) 624-7487 (outside California)

Products:  QuickMod          ver. 1.00  MSDOS   $95.00
           Modula-2 Compiler ver. 1.14  MSDOS  $345.00

Logitech, Inc.
6505 Kaiser Drive
Fremont, CA 94555
(415) 795-8500
Contact: Debbie (415) 795-0427

Product: Modula-2 Development System ver. 3.03  MSDOS  $249.00
         Compiler Pack               ver. 3.03  MSDOS   $99.00

Avant-Garde Software
2213 Woodburn
Plano, TX 75075
(214) 964-0260

Product:  Benchmark Modula-2 Compiler           ver. 1.00  AMIGA  $199.95
          Simplified Amiga Libraries            ver. 1.00  AMIGA   $99.95
          'C' Language Libraries                ver. 1.00  AMIGA   $99.95
          IFF Libraries, Image Resource Utility ver. 1.00  AMIGA   $99.95

Dr. K. N. King
Dept. of Mathematics & Computer Science
Georgia State University
University Plaza
Atlanta, GA 30303
(404) 651-2245
D.C. Heath order line  (800) 334-3284

Product: Modula-2: A Complete Guide   $33.33 (ship. incl.)
         Source Disk for book         $10.00
```

♦ ♦ ♦

The editor supports up to four open files in four windows, and you can add your own commands to its environment menus. The editor starts out like WordStar but can be customized.

The library contains 12 modules and over 250 procedures. Although the primary library is very complete, it's less Wirth compatible than any of the other packages.

JPI does include a set of standard modules on the Library Source disk. Just compile them and you're compatible. On the other hand, I wouldn't mind if JPI's fancier library became the standard. But either way, I'd like to see all vendors support a standard.

The default library includes time-slice process control, graphics, and a powerful windowing module.

The VID (Visual Interactive Debugger) is easier to use than Stony Brook's but does not have Logitech's windows. The debugger option includes a symbolic disassembler, an execution profiler, and a utility which lists the procedure sizes for each module and each module's percentage of total code.

Their TechKit includes an assembler, ROM support, TSR support, communications support, EMS support, startup source, and overlay support. TopSpeed comes on three disks, the debugger and toolkit on one each.

Logitech Modula-2

I started programming in Modula-2 when my programs became too long and my patience too short for Turbo Pascal 2.0. (Seven minutes to recompile the entire source!) I plunked down my $99 and got Logitech's Compiler Pak 2.0. It was love at first compile. I soon had my monster program broken down into manageable modules.

It wasn't long before I overran the limits of the compiler's identifier table. I had to wait six months for Logitech version 3.0 to hit the market. I upgraded to the Development System and away I went again. The number of utilities that came with the development system amazed me. Logitech still has the largest goody bag of any of the systems.

Logitech has been around for some time, but new kids on the block like Stony Brook and TopSpeed challenge Logitech's seniority. Logitech still puts in a respectable performance. DOS and OS/2 versions of the compiler are available and the standard library is large. The great execution times for the FileIO benchmark resulted from the disk buffering. (I couldn't disable the feature.) I tested all the other compilers without disk buffering.

The development system includes the compiler, linker, library source code, editor, post-mortem debugger, Pascal to Modula-2 translator, source code formatter, symbolic runtime debugger, make, xref, code checker, object file decoder, and program version maintenance. The translator works very well and includes extra modules to support Turbo Pascal. I liked the editor in the 2.0 version; the 3.0 editor needs a mouse. You can compile, link, and run your programs from the editor.

When I refer to Logitech, it is the Development System because the Compiler Pak is being phased out.

Logitech provides the best debugging tools. This includes windowing runtime and post-mortem debuggers.

The large library supports graphics, mouse, communications (RS-232), overlays, and decimal numbers.

They've also included the Point windowing editor which is difficult for me (perhaps some additional use will change my mind) because I don't have a mouse. Point is a configurable editor that acts much like, but not as nice as, the TopSpeed and QuickMod versions.

Logitech also includes a Pascal to Modula-2 translator. It's ready to translate Turbo Pascal 3.0 to Modula-2 and comes with a support library for Turbo Pascal. The translator does a very good job of conversion and seems to flag just about all potential problems. I translated a 73,852 byte Turbo Pascal 2.0 pro-

gram, hand editing only nine lines!

Logitech plans to release version 3.5 in June, 1989. New features include an enhanced debugger, 386 support, 30-40% faster compiler, 15-20% faster code, and the linker will support all Microsoft extension object file formats.

Currently you can buy just the compiler for $99, but when they release 3.5, you'll have to get the full Development System. The 3.03 Developer System runs $249, but 3.5 will be $299. You'll be able to upgrade from 3.03 Dev. System to 3.5 Dev. System for $49.

FTL Modula-2 MS-DOS

FTL includes a compiler, linker, librarian, editor, assembler, profiler, and a simple debugger. The editor provides an integrated environment for calling the compiler, linker, running a DOS Shell, and correcting errors in the source. The small memory model (SMM) supports 64K of code and 64K of data. The SMM FTL compiler is probably the best buy in compilers I know of — perfect for anyone who wants to try the language.

The large memory model (LMM) supports 64K of code and 64K of data per module (not including the stack or heap).

FTL generally had the slowest times and largest EXE files, but only the MemTest and Whetston times were awful. Workman claims the floating point emulation times are slow because they designed the emulation for accuracy, not speed. As with all their compilers, they've included the library source so you can change it.

All the other times were respectable. It beat Logitech in the Dhryston and held its own in the FileIO benchmark. FTL LMM 2.0 is expected out in March, 1989, and will cost $99.95. It's supposed to be a major upgrade to the compiler.

FTL Modula-2 Z80 CP/M

For $49.95, this is a bargain. The package includes a compiler, linker, editor, librarian, assembler, and support for ROMable code. FTL can't compete with Turbo Pascal 3.0 file sizes or execution times, but it is a good development system for Modula-2.

If you don't like the way a module is implemented you can change it; they've thrown in the library source. For example, one reason their Whetston times are so slow compared to Turbo Pascal 3.0 is that FTL uses 8 byte reals. Turbo uses 6 byte reals. If you want 6 byte reals, you can change the source.

FTL Miscellaneous

Workman markets Modula-2 compilers for Z80 CP/M, MS-DOS small and large memory models, and Atari. The libraries are not fully Wirth compatible but offer a fair degree of compatibility from compiler to compiler. One primary difference is the UNIX-style stream, file, and device I/O.

The manual for all versions is one spiral-bound book. The book doesn't list the definition files and they must be printed from disk.

Workman includes the library source in the package. FTL's packages include a librarian (maintains program and code libraries), library source, compiler, and linker.

Workman also sells an Advanced Programmer's Kit that includes a smart linker, a multitasking kernel, an overlay utility, and source code ($39.95). For an additional $49.95, you get the Editor Toolkit which includes a make utility, symbolic debugger, and editor source code.

Turbo Pascal 3.0

I included Turbo Pascal 3.0 as an alternative to the FTL CP/M Modula compiler. Borland's compiler and code are generally faster than FTL's. It does not support co-routines or separate compilation (notice the absence of information for the benchmark CQuikSrt in the tables).

This is a good implementation of Pascal. One of the key features for me is the ability to create and use overlays painlessly. Overlays allow you to shoehorn large programs into CP/M's small memory space.

Benchmark Modula-2

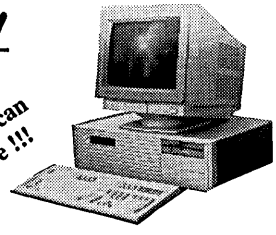
Avant Garde calls their Benchmark Modula-2 a "software construction set." Benchmark is complete, lacking only a debugger. Avant Garde plans to release a source/assembly level debugger in the second quarter of '89 for $100-$150.

Benchmark has an integrated environment that fully supports the Amiga interface. The compiler worked well with 512K and two floppy drives. That is about the minimum configuration for accomplishing anything on the Amiga.

Modules included in the package cover most of the standard library very well. The file handling is only a little different because of the Amiga's multitasking environment. Co-routine handling is not standard because Amiga's operating system handles it.

I did receive a module that covered

High performance AND low prices?
You must be *DREAMING!*



The Dream-286

AT 6-10 MHz 0 wait state motherboard
640 K of on board RAM (100 nS)
2 Parallel ports and 2 serial ports
1 .2 Megabyte 5.25" floppy drive
40 Megabyte hard drive
Hard/floppy disk controller card
12" amber Samsung monitor (tilt/swivel)
Hercules compatible mono graphics card
101 key enhanced keyboard

$ 1495

For the 286, in place of mono:
o RGB Color monitor and card.... $1695
o EGA Color monitor and card..... $1955
For the 386, in place of EGA:
o NEC Multisync $3145

The Dream-386

80386 20 MHz motherboard
1 Megabyte 80 ns on board RAM
2 Parallel ports and 2 serial ports
640 x 480 on board EGA/VGA card
1.2 Megabyte 5.25" floppy drive
40 Megabyte hard drive
Hard/floppy disk controller card (1:1 IL)
High resolution EGA monitor
Professional enhanced (101 key) keyboard
Case (UL/FCC) holds 5 half-height drives

$ 2995

DreamTech
5175 Moorpark Ave • San Jose, CA 95129 • (408) 996-2373

Prices are subject to change without notice. All orders are shipped UPS FOB San Jose unless otherwise specified at time of order. California residents add 7% sales tax. The following trademarks are recognized: IBM PC/XT,AT to IBM Corp., Hercules to Hercules Computer Technology, Inc. Other product names are trademarks of their manufacturers.

Reader Service Number 16

the NEWPROCESS and TRANSFER procedures. The current release doesn't include it, but the next version will. The included modules also provide an interface to all the Amiga hardware and software services, including the ROM Kernel, AmigaDos, and Intuition.

Benchmark comes configured to run on the 512K Amiga. The only thing time-consuming about installation was unarchiving the incredible number of sample programs. The sample programs are important because the Amiga is a complex machine and some programming tasks are more involved than MS-DOS or CP/M.

The Benchmark package includes a compiler, linker, editor, program profiler, cross reference utility, assembly language interface, and a decent manual. Unfortunately they don't supply an integrated make.

Avant Garde offers three additional libraries. They include: C Language Library for porting from C to Modula-2; Simplified Amiga Library which allows easy access to speech, graphics, window, and screens; and the IFF Library and Image Resource Utility that provide tools for handling Amiga video images. It's a reasonably priced "software construction set." I like the package and recommend it.

Bottom Line

Well, here come what I hope are objective personal recommendations.

TopSpeed's lightning performance, excellent integrated environment, and optional debugger and Techkit make it easy to recommend. All these features and performance make TopSpeed the winner. If you don't intend to use TopSpeed's library and choose the compatibility library instead, performance will suffer.

Despite TopSpeed's qualities, I've chosen QuickMod as my system of choice. I simply prefer its environment over TopSpeed's. QuickMod has fine performance and a large library.

I really like performance, and I'm eagerly awaiting the arrival of Stony Brook's Professional Modula-2 system. Then Stony Brook's Modula-2 Compiler will have the QuickMod integrated environment.

Because of the Modula-2 compiler's flexibility and control, I would recommend it to anyone developing an operating system, interfacing with foreign libraries, or developing 80x86 software to run on non-MS-DOS platforms.

If you like tools, the Logitech Development System is the package for you. It may not be the fastest, but Logitech has decent performance. I've watched them steadily increase the performance and usability of their product over the last several years.

Execution times hurt the FTL compilers. They are, though, good compilers at a great price. They would be a perfect introduction to Modula-2. I'm expecting FTL LMM 2.0 in March, and I'll report on the changes in this package and any others as I receive new versions.

Avant Garde has developed a fine development system for the Amiga. It was nice to be able to program the Amiga in a language other than BASIC or C, the only options until recently.

But you and I are the real winners. We have an excellent set of Modula-2 development tools. Whff, pftt, pop, bang, wobble wobble, crash. Am I done yet?

CP/M Note:

Doing the review reminded me that I like CP/M; I really do. But I find the lack of utilities and the spartan user interface frustrating. ZCPR3 has been around for quite a while, but replacing the command processor with ZCPR3 is a chore, even for an expert. We've needed an easy way to install and customize ZCPR3.

That is precisely where NZ-COM shines. NZ-COM lets you alter the operating system on the fly (without rebooting). NZ-COM is not just a face-lift for CP/M, it's an overhaul.

I tried NZ-COM on a Kaypro 2X with CP/M 2.2H. I installed and modified various flavors of the Z-System without a hitch. A full-up system left 51.5K TPA. If you use CP/M on a regular basis, I recommend NZ-COM.

Next Time

Micro C seems to be full of graphics articles lately so I've put mine on the back burner for a while. Sorry if you were waiting anxiously, and you're welcome if you are relieved.

Usually by this time I'm well into a project and know what the topic is for the next issue. The compiler review took so much time that I haven't decided which of the several projects brewing will be soup by issue #48. Hope you enjoyed the review. Maybe again next year. Until next time, hasta luego.

♦ ♦ ♦

You Don't Have To Be A Programmer To Be An Expert.

TINY EINSTEIN 2.0

The Expert System Shell

- Create expert systems and dynamic databases 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
- For Diagnosing... Simulating... Predicting... Classifying... Training... Monitoring... and Organizing systems.

Only $99.95! (Plus $5 S/H)
Demo $10.00 • The AI Newsletter $1

Reader Service Number 72

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

CP/M NOTES

FORTH Talk

Your PAD Or Mine?

A while back I sent you a letter regarding some solutions to my Floundering FORTH problem as printed in CP/M Notes (Issue #44 November/December 1988). I received a few letters, including one from Kevin Appert, who, with Bob Bumala, wrote the enhancements to the figFORTH you have for the Kaypro. (Micro C user disks K12 and K13.)

I had devised a rather long-winded solution to ID. overwriting PAD, but Frank Snively (Fallbrook, California) found a way so simple it should be called elegant. Here it is:

```
: PAD ( -- addr ) PAD 100 + ;
```

Ignore the error message stating that PAD is not unique.

FORTH lets you define two or more words with the same name. Since the compiler scans the dictionary beginning with the most recent Words, it will use the most recent definition. As a result, ID. will use the old PAD, while the rest of the world can use the new PAD in peace.

Walter J. Rottenkolber
P.O. Box 936
Visalia, CA 93279

Further FORTH

I am responding to Walter Rottenkolber who is having two problems with KFORTH:

1. The TYPE statement does not do what he thinks it should — it types too many characters and sometimes types garbage.

2. Something is overwriting the PAD buffer when he doesn't want it to.

The first problem is easy to solve. After loading KFORTH.COM, use the decompiler GOESINTO to see what goes into the word TYPE. Like this:

```
GOESINTO TYPE
```

Hit the space bar after each component of the word is displayed. It is evident that it displays a character string of "n2" bytes starting at address "n1." No comparison is made — TYPE does not look for a null byte or a carriage return. So if you say:

```
PAD 20 EXPECT PAD 20 TYPE <cr>
```

but only type in 10 characters before a carriage return, TYPE will output 20 characters anyway, and the last 10 may be garbage. Note that EXPECT terminates after 20 characters or a carriage return, whichever occurs first.

The second problem is a bit more problematic. It turns out that if you put:

```
PAD 20 EXPECT PAD 20 TYPE
```

on the same line, it works. But if you put:

```
PAD 20 EXPECT <cr>
```

followed by your input string and then:

```
PAD 20 TYPE <cr>
```

you get garbage. The culprit is the word STATUS, which is defined on screen 16 of FORTH.SCR. Screen 10 redirects QUIT to the new main loop, SQUIT, which uses STATUS to print out the stack contents, base, and current vocabulary after each FORTH line is executed. The problem is that VOC. assembles a string in PAD before displaying it. This trashes whatever was in the PAD buffer.

This should not present a problem since it does not affect use of PAD from a single command line. Just remember that PAD is for temporary storage, and the system is as free to use it as you are.

To reserve an area for permanent storage, try this:

```
0 VARIABLE TXT-BFR 19 ALLOT <cr>
TXT-BFR 20 EXPECT <cr>
TXT-BFR 20 TYPE <cr>
```

I like the public domain implementation of F83 (Laxen and Perry). This version of FORTH, also available for the Kaypro, is much more complete than KFORTH, includes the source and many more tools, and has much better file handling. The screen editor in KFORTH is smaller and simpler, and probably easier to get started with.

The best documentation for F83 is *Inside FORTH-83*, by Ting, available through the

FORTH Interest Group. For figFORTH, I like *FORTH Fundamentals*, by McCabe, published by Dilithium Press.

Peter Henry
ACTEK Inc.
12740 28th Ave. NE
Seattle, WA 98125

FORTH Forever!

FORTH is better thought of as a concept rather than a specific language. The very flexibility and power of FORTH leads to almost as many varieties as there are programmers. FORTH is not something to be dabbled in. One might try BASIC or Pascal and get some results. However, to become proficient in FORTH, considerable understanding of the particular FORTH system is required.

Unfortunately, many very poorly written systems labeled FORTH have been distributed. The original figFORTH was very clean. All too many of the extensions leave much to be desired. In recent years it has been popular to do dynamic patching into the body of the system. Such patching is a mixed blessing. Imagine the power and confusion of a system that changes as you use it!

For the student of FORTH (aren't we all?) I would recommend using a good, mature FORTH system consisting of metacompilers, an advanced editor, an assembler, and all the source code and documentation for the system. I would never recommend trying to debug an obsolete system. Life is too short to get into endless problem chasing.

F83 by Henry Laxen and Michael Perry is a real whistle and bells system and it is in public domain. It has some resemblance to FORTH. It will run with CP/M. Incidentally, running F83 or any good FORTH with CP/M or any usual operating system is like tying an Indy car to the back of a horse and buggy.

My suggestion then is to join the figFORTH group (these people provide all the help you could ever need or want) and obtain the newest and best system for your machine (probably F83).

There are various levels of FORTH students. First grade might be interpretive communication with the system. Second grade could be the use of the editor and assembler. Third grade would be to write your own system using the compilers. The next step might be to write your own compilers. Graduate work would involve the contribution of a new approach in FORTH to the comunity.

Those students who finish the third grade are not likely to want to return to other languages.

Darrell D. McKibben
1087 Remington Dr.
Sunnyvale, CA 94087

Editor's note: Harumph!

Kaypro Clock

I recently had to have my Kaypro 4-84 operated on. Turns out the power supply was sending occasional power spikes. With that replaced, all is fine again. I recommend Xerox Service (at least the Rochester, New York, branch still knows CP/M machines — they were discharging an Osborne when the Kaypro and I arrived).

The only problem was after servicing the machine when I used QP/M and got the message "Bad Clock?" It seemed to keep time okay after I reset it. I played with it for a day, but the message came up with each QP/M cold boot.

I don't know what made me think of it but I tried a program supplied with the machine — the graphic clock display in MBASIC. I have always thought it was pretty, but slow and not very useful. Haven't used it in over a year.

When I ran the Kaypro clock program, it asked if I wanted to set the clock. I said no because the clock was displaying the correct time with other programs. Then it said, "Clock not initialized..., Enter new value:" and I said YES. It paused a moment and then asked for the time to set the clock. When I next cold booted with QP/M, all was fine.

I suggest that after any internal hardware work on a Kaypro with a clock, the Kaypro clock program should be run. That may solve other people's clock problems. My other suggestion is to buy QP/M. After getting QP/M the clock keeps good time — before, it lost a few minutes a week.

QP/M by MICROCode Consulting
P.O. Box 9001
Torrance, CA 90508
(213) 212-5877

George W. Richards
154 Hurstbourne Rd.
Rochester, NY 14609

Turbo Pascal News

I recently discovered that Borland will allow upgrades from the CP/M version of Turbo Pascal to the new Turbo 5.0 Professional Pak for only $125 plus $5 postage, prepaid, which is a pretty good price. In case you have any other readers who haven't bothered to switch, they might want to take advantage of this. Borland also told me that Alpha Systems still supports the CP/M Turbo.

Alpha Systems
711 Chatsworth Place
San Jose, CA 95128
(408) 297-5583

Andy Morriss
1220 N. Hwy 75
Corsicana, TX 75110

♦ ♦ ♦

011
100
100
TIDBITS

Desktop Publishing —

Report From The Davis Workstation

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Well, desktop publishing has certainly filtered down to the farthest reaches (California, yet). This time Gary looks at screen grabbers and Word Perfect 5.0.

It's easy, right? You write it; you print it. If you use a typewriter, you save an entire step. Everything else, *they* say, is icing.

Well, not quite. If you hope to improve your status in the publishing business, attract more customers (with stimulating brochures and newsletters), or just make a good impression, you might want to add a little something. Chances are a typewriter alone won't be good enough, unless you write like Philip Roth or Alice Walker.

You have two choices —

- hire someone to ice;
- ice it yourself.

If you only need to make your impression once (and aren't compelled to know everything yourself), the choice is obvious — hire someone. You'll avoid a mountainside of frustration.

If you do want to make your own mistakes, have fun in the process, and perhaps even make a business of helping others, then get into desktop.

We'll try to help by sharing our *Micro C* discoveries and (shudder) frustrations. Desktop publishing is the hottest, most complex, and most confusing application you can tackle with your PC. Databases, spreadsheets, communications, networking, you name it, are faster to learn (with fewer hair-pulling scenes).

If you've read Dave's articles on desktop publishing, you know we've gone through a trial or two learning to ice *Micro C*. Dave decided we'd take the plunge in Issue #35 (the 1987 April Fool issue), and for two years we've been learning what will and won't work. Obviously, we're still learning.

Down here at Micro C Davis (my remote office at the civilized edge of California), I'm trying to improve the images I upload to the main office. Since I don't "publish" an entire magazine, I don't need everything Micro C Bend needs, so I can cut a few corners and spend less.

I've discovered that I don't have to have Ventura Publisher to produce high-quality documents, newsletters, and images which themselves can be understood by Ventura.

Several word processors costing half that of Ventura now incorporate desktop publishing features, such as fonts and graphics. These might be all you need.

Drawing Some Lines

Desktop publishing confuses us, in part because there are so many things we can do, and so many names we can call the operations. Even the term "word processing" carries many shades of meaning. Where, for instance, does "word processing" stop and "desktop publishing" begin?

I choose to think along three levels —

- text processing
- special effects
- document processing

Text processing is anything you do to words that you can see on a monochrome display. Rough draft level, I call it.

Special effects are the fonts (bolds, underlines, italics, etc.), the sizings, anything that you can do to change the appearance of words.

Document processing dresses up the pages of special-effected words. Here we draw, add shapes, create columns, import graphics, and generally prepare a document for printing or typesetting. The graphic element distinguishes it from "word processing."

All *Micro C* articles, for example, spend several weeks (or more) at level one. We read each article, check it for technical accuracy, revise it, check the revision for grammatical and spelling accuracy, before we move on to the fancier stuff.

At level one, all four of us use a programmer's (or "just text") editor. (Dave insists on having a passel of macros. Larry can only use his personalized keyboard configuration.) Once an article survives level one, Cary uses Webster's New World Spelling Checker and Grammatik III to tighten up what's left.

For levels two and three, Micro C Bend

moves into Ventura. Micro C Davis doesn't, relying on several less-expensive, yet powerful-enough tools to produce newsletters, brochures, and output understandable by Ventura. If you're on a budget, and aren't ready to make the full leap into Ventura (or PageMaker), you might consider some of the tools I used this issue.

Micro C, #47

For example, I wrote a program in Turbo Prolog using the BGI (Borland Graphics Interface) to display strange attractors for "The Last Page." Since I wanted to publish them in *Micro C*, I needed to capture the display and upload it to the Micro C BBS.

I didn't have the time or inclination to write a screen capture program which would save the display to a Ventura-readable format (.PCX). So I looked around for a good commercial screen capture utility.

I found several. In fact, many desktop publishing and drawing programs include screen capture utilities. The problem I found with the built-in capture routines was a lack of universality and versatility.

All screen capture programs generate pixel-based graphics files. (After all, the screen is made up of pixels and it's next to impossible to go from pixels to vectors.)

Bit-mapped images are relatively simple to generate, capture, manipulate, and display. Video cameras and scanners, for example, scan rows of bits. And rows of bits make up displays. So the translation is easy. Most desktop publishing (i.e., document processing) software tools use bit-mapped images (.PCX and .TIF files, for example, are the most common bit-mapped graphics formats).

Vector-mapped images are made up of straight lines and curves along with attributes like width and color. Vector files include: .PLT, .CGM, .PIC, .WMF, and some CAD and spreadsheet generated images (see Figure 1 for a Who's Who of graphics formats).

Each type of graphics mapping has its advantages and disadvantages. Bitmaps are —

- easy to capture and generate;
- easy to manipulate (many graphics editors let you turn single bits on or off);
- easy to combine with text;
- easy to invert (dark pixels become light; light pixels, dark);
- easy to rotate & scale;
- prone to jagged edges when printing (so a smoothing algorithm may be used to enhance the printed image).

Vector-mapped images are —

- harder to invert;
- easier to rotate and scale;
- harder to manipulate when used with text.

Editor's note: The only way to print or display a vector image is to plot it. Otherwise you have to translate it into a pixel image. You'd translate it into a roughly 100 by 100 dpi pixel image for an EGA video monitor, or a 300 by 300 dpi image for a laser printer. Either way, it's a pixel image that's being displayed or printed.

Vectors give three very important benefits: 1) They have infinite resolution so the sharpness of the produced image depends entirely on the output device. 2) Vector images can be resized almost infinitely. 3) Vector files can be significantly smaller than the equivalent pixel files. The type for this magazine, for instance, was produced from a vector file. The resolution of the type and its size depends on the output device, a laser typesetter.

I've found two excellent screen capture programs —

- Pizazz Plus (from Application Techniques);
- Catch (a PaintShow Plus utility, from Logitech).

Something else I should mention. Screen capture utilities are TSR (Terminate and Stay Resident) programs and they use the printscreen interrupt (#5).

If you run programs which compete for this interrupt, or if you run more than one TSR, or if you just need to reclaim all available memory for another application, you'll want to be able to unload your TSR. You can unload both Pizazz Plus and Catch.

Catch and Pizazz Plus capture displays which can be embellished, rotated, printed, and then exported to compressed or uncompressed bit-mapped files. The files can be read by Word Perfect 5.0, PageMaker, Ventura, and many other drawing and document processing programs.

I captured Figure 2 (a wild attractor) and massaged it before outputting as a Ventura-readable file.

Catch exports fewer formats and supports fewer printers (30 or so) than Pizazz Plus (which supports many formats and over 200 printers).

Word Perfect 5.0

You've noticed that I've mentioned Word Perfect several times in the same breath with Ventura and PageMaker. The reason — Word Perfect 5.0 offers a less expensive alternative. You can produce many small projects, including newsletters and brochures, easily with WP5.

In particular, WP5 lets you combine

Figure 1 — Partial List of Vector- & Bit-mapped Formats

```
vector-mapped

.CGM            .EPS
.DHP            .GDI
.DXF            .HPG
.HPGL           .DRW
.PIC
.WMF
.PLT

bit-mapped

.TIF
.PCX
.IMG

        ♦ ♦ ♦
```

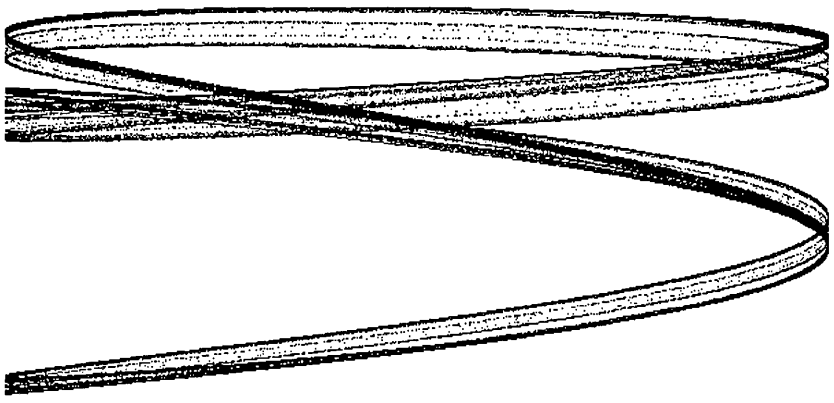


Figure 2 —A Wild Attractor.

Can't Afford A PostScript® Printer?

for only $195

NOW YOU CAN Print PostScript Language Text and Graphics On almost* ANY Printer

GoScript™

PostScript® Language Interpreter for the IBM PC

GoScript Software, the Low-Cost PostScript Language Printing Solution

Choose the one that fits YOUR needs:
GoScript - $195 - With 13 Fonts!
GoScript Plus - $395 - With 35 Fonts!
Both include our High Quality, Scalable Outline Fonts

*Includes Drivers for: NEC Pinwriter; HP LaserJet, DeskJet, PaintJet; Canon LBP-8II, BubbleJet BJ130; Epson LQ and FX; Toshiba 24-Pin; Fujitsu DL 24-Pin; Panasonic KX 24-Pin; IBM ProPrinter, Quickwriter, Quietwriter

ORDER TODAY!
Contact Your Local Dealer
or call the LaserGo Order Line
(800) 451-0088 (outside Calif.)

In Canada, Contact: CDP Communications, Toronto, ON, TEL: (416) 323-9666 FAX: (416) 323-3878

Exclusive European Distributors: Graphic Sciences Ltd., Surrey, England TEL: (01) 940-9480; FAX: (01) 948-2851

LaserGo, Inc
TEL: (619) 530-2400
FAX: (619) 530-0099
9235 Trade Place, Suite A, San Diego, CA 92126

LaserGo, GoScript are trademarks of LaserGo, Inc. PostScript® is a registered trademark of Adobe Systems, Inc. All other product names are trademarks of their manufacturers.

Reader Service Number 144

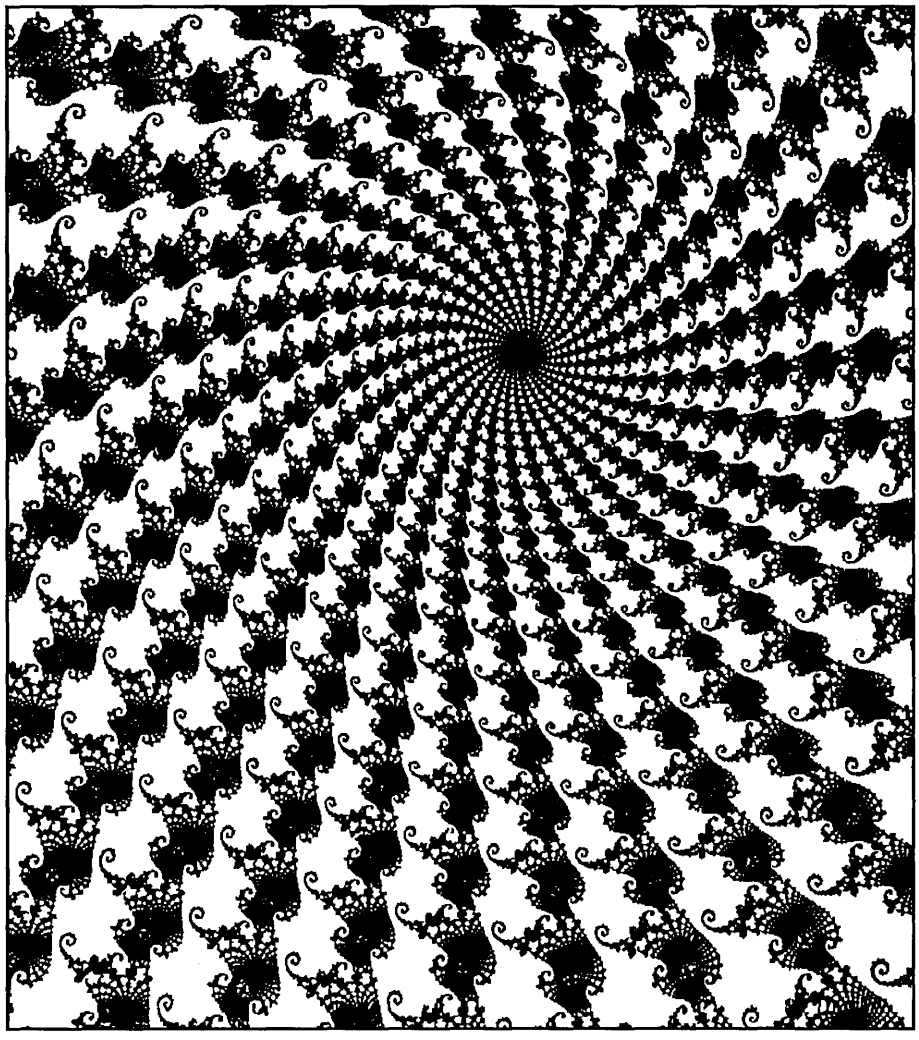


Figure 3 — Image from Scan Man

word and document processing in a medium-sized program. You edit in a familiar text mode (add fonts, create columns, measure in points, centimeters, or inches), then switch (via a function key) to graphics mode where you can import and massage images and view the document before you print it.

I found a lot to like about WP5. In particular, an excellent auxiliary program, "PTR," lets you create and modify printer drivers. If you have an older printer (I do), you're probably discovering that these fancy, new programs often ignore you. With PTR and your printer control codes, you can quickly bring up a new driver.

If you don't have a listing of your printer control codes, or if you just need control code information for virtually any printer, contact Cardinal Point Press (in Ellettsville, Indiana).

Cardinal Point publishes a three-volume set entitled, *Programmers' Handbook of Computer Printer Commands*. It's a gold mine of printer information.

If you're planning on using WP5 for desktop publishing, I urge you to check out, *WordPerfect 5, Desktop Publishing In Style*, by Daniel Will-Harris. It's swarming with useful examples, how to, and how not to information.

Danny (and wife Toni) wrote, edited, and formatted the entire book with Word Perfect, then printed it on an HP LaserJet II. It's a good example of what you can accomplish. Both text and design are informally attractive.

Scan Man

I won't get into peripheral hardware or recommend printers, but I do want to mention Scan Man (from Logitech).

Scan Man, an inexpensive hand held scanner (between $200-$300 depending on dealers' prices), will quickly scan a 4" by 6" image (at 200 dpi) into PaintShow Plus (the Scan Man software). There you can rotate, magnify, modify, and generally massage the image, before exporting it (if you need to) to document processors. Figure 3 is an image I scanned and exported to Ventura.

As far as I know, Scan Man (including the PaintShow Plus software) is the best-deal scanner on the market. It works, it's fast, and you probably don't need a second job to afford it. It's only 200 dpi, though; good laser printers are 300 dpi, so Scan Man's input doesn't take advantage of the laser's potential output. For many low-end desktop publications, though, 200 is adequate.

For more information —

Pizazz Plus
Application Techniques, Inc.
10 Lomar Park Dr.
Pepperell, MA 01463
(800) 433-5201

Word Perfect 5.0
Word Perfect Corp.
1555 North Technology Way
Orem, UT 84057
(800) 227-4000

Scan Man & PaintShow Plus
Logitech, Inc.
6505 Kaiser Dr.
Fremont, CA 94555
(415) 795-8500

BGI
Borland International
1800 Green Hills Road
P.O. Box 660001
Scotts Valley, CA 95066-0001
(800) 345-2888

Grammatik III
Reference Software
330 Townsend, Suite 123
San Francisco, CA 94107-9883

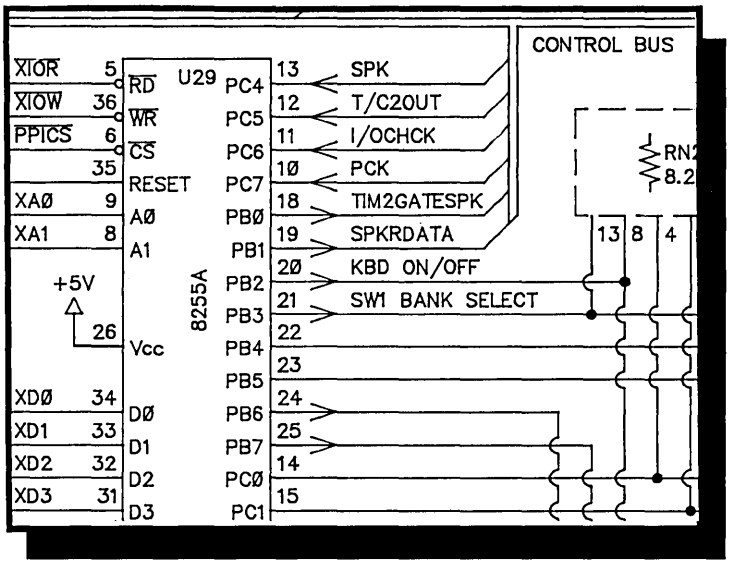
Webster's New World Spelling Checker
Simon & Schuster
1230 Avenue of the Americas
New York, NY 10020
(800) 223-2348

Word Perfect 5, Desktop Publishing In Style
Peachpit Press
1085 Keith Ave.
Berkeley, CA 94708
(415) 527-8555

Programmers' Handbook Of Printer Commands
Cardinal Point Inc.
P.O. Box 596
Ellettsville, IN 47429
(812) 876-7811

◆ ◆ ◆

MICRO CORNUCOPIA **XT SCHEMATIC**



At last you can plumb the mysteries of your computer with this single sheet schematic of the IBM XT's main board. A wealth of information for both True Blue *and* clone owners.

Need to know just how a non-maskable interrupt occurs (and how to mask it)? Is your keyboard dead (or do you just want to know how to disable it)? A trip through our schematic will answer your questions.

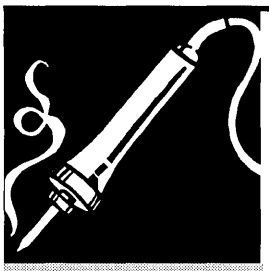
Although clones use slightly altered board layouts and different chip location names, they're close enough to the original for this schematic to be very useful. As an example — you have a dead clone. Lil sucker won't even beep. A look at the schematic shows the location of parallel port A. You know that the power on self test loads a checkpoint number into port A before each test. So now all you have to do is read port A with a logic probe to see how far the system went before it puked.

We'll include a list of these checkpoint numbers and some other pertinent trouble shooting information with the schematic.

IBM PC-XT Schematic $15.00

Phone Orders: (503) 382-5060 or 1-800-888-8087
Monday-Friday, 9 AM - 5 PM PST

Mail Orders: P.O. Box 223, Bend, Oregon 97709



Techtips

TECHTIPS

The Extirpated Coprocessor

All the popular AT BIOSes now include a setup program in firmware so you can easily set all the CMOS parameters. It's nice to not have to dig out the setup diskette, but here's an interesting gotcha that all BIOSes seem to have. It involves the 80287 coprocessor. No one mentions the coprocessor in their firmware setup menu, but everyone seems to detect and add it automatically.

The key is "add," because once the 80287 is flagged as being installed, no one seems to notice when it's gone. Programs which take the CMOS configuration as gospel (Lotus 1-2-3 V2.0 and Quattro, for example) leave you scratching for answers to their weird and wonderful behavior.

The solution is to dig out that unloved diagnostic diskette and run setup. In the process, it'll notice the lack of an 80287 and tell you it's removing it. Now how could all those BIOS programmers have made the same omission?

On a vaguely related topic, those short on budget and long on faith in Intel's conservative nature have probably observed by now that the 80287-6 almost invariably works fine in 10 MHz ATs, where the coprocessor clock is 6.67 MHz. (If you print this, be sure to save space in the next issue for the requisite letter from Intel explaining why anyone who spends too little on one of their parts is taking a horrible chance.)

John Axline
1216 E. Plymouth St.
Glendora, CA 91740

Floppy File Recovery Revisited

Tammy trashed one of her floppies the other day. Skewered it somehow. (I never ask how these things happen.) Happy ending though — the important files came off okay.

Of course I couldn't recover the punctured file, but the process got me thinking about file recovery in general. Barring physical damage to the disk, DEBUG provides a simple method for recovering files lost due to directory and FAT corruption. What if we replace the faulty directory and FATs with new copies defining a single huge file. Then the entire disk will be accessible through that one file.

You need a good directory and FAT, so use COPY to concatenate 362,496 bytes worth of files together on your hard drive (or just use one of Dave's editorials). Copying the resulting file (BIG.BIG) to floppy gives you access to the new directory and FAT.

To make a file copy of the directory, put the BIG disk in A: and use DEBUG as follows —

```
A:\>DEBUG
-NBIGDIR.DIR ;name the file
-RCX         ;load file size
CX 0000      ;into CX
:E00         ;7 sectors
-L 0 0 5 7   ;load 7 sectors,
             ;starting at sec 5,
             ;from drive 0 (A:),
             ;to memory offset 0
;put scratch disk in A:
-W0          ;save the dir
Writing 0E00 bytes
```

I got into trouble with DEBUG the first time I tried this. Seems DEBUG uses memory around offset 0 for its own scratch RAM. So you need to name the file and load its size *prior* to loading the directory. Or, you could load the directory at an offset of 200 or so. Either way will keep DEBUG from overwriting your carefully read directory.

Use the same method to save a copy of the FAT —

```
;BIG disk is in A:
-NBIGFAT.FAT ;name the file
-RCX         ;load file size
CX 0000      ;into CX
:400         ;two sectors
-L 0 0 1 2   ;load two sectors,
             ;starting at sec 1,
             ;from drive 0 (A:),
             ;to memory offset 0
;put scratch disk in A:
-W0          ;save the FAT
Writing 0400 bytes
```

Hang on to BIGDIR.DIR and BIGFAT.FAT for use in the future. For now, load them onto the trashed disk like this —

```
;put scratch disk in A:
-NBIGDIR.DIR
-L0          ;load good dir
;put trashed disk in A:
-W 0 0 5 7   ;write the new dir
             ;to the trash disk
;put scratch disk in A:
-NBIGFAT.FAT
-L0          ;load good FAT
;put trashed disk in A:
-W 0 0 1 2   ;replace both FATs
-W 0 0 3 2
-Q           ;quit DEBUG
```

A directory of the trashed disk should now show only BIG.BIG and 0K free.

I used two parameters that may vary with different DOS versions (3.21 here): the size of a full disk (362,496 bytes), and the number of sectors per FAT (2). Run CHKDSK on a floppy to confirm the disk size. The FAT size comes from the boot record of the floppy.

DEBUG will read the FAT size as follows —

```
A:\>DEBUG
-l 0 0 0 1   ;load first sector
-DB          ;dump from offset
             ;0bh
-Q           ;quit DEBUG
```

Sectors per FAT shows up in the first

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 ¼ inches by 1 ¾ inches.

C SCIENTIFIC LIBRARY

C programming library for computations in research and mathematical analysis. Each routine is designed and documented for use by technical specialists and programmers.

Comprehensive Over 500 functions
Superb Documentation Matrix Math
Microsoft or Turbo C Statistics More

Eigenware Technologies
13090 La Vista Drive, Saratoga, CA 95070
(408) 867-1184

Reader Service Number 137

F L A S H

THE DISK ACCELERATOR

- EASY to Install
- Cache up to 32 MEGS of EXTENDED and or EXPANDED
- Buffers up to 26 DEVICE driven drives
- Comes with 2 FREE utilities!!!!!

ORDER NOW $69.95
(800) 25-FLASH

SOFTWARE MASTERS 6352 North Guilford Ave.
Indianapolis, In 46220 / (317) 253-8088
$5.00 Shp/hnd in USA & CANADA, $15.00 overseas.

Reader Service Number 106

"Enhance! is a must for any power user!" C. Gengler

Enhance! *your DOS command line interface.*

- advanced command line editing, retrieval.
- full symbol alias defining, modifying, and processing.
- robust file management. . . move, copy, append, list remove via powerful and substantial extensions of the DOS wildcard file specification support. allows multiple commands manuvering
- RAM resident can load/run in expanded memory.
- Unix (TM),VMS (TM)-like interface.
- enhances, not replaces, DOS 2.0-4.0/COMMAND.COM.

Cortex Computing Corporation
P.O. Box 116788, Carrollton, Tx 75011
$79.95 214-492-5124 $79.95
includes a free "I've been **Enhanced!** *" T-shirt!*

Reader Service Number 128

STOCKS OPTIONS FUTURES

Turn Your PC Into A MARKET QUOTATION MONITOR

100 page book covers satellite and radio data reception of financial news and quotes for your PC. $19 (includes demo diskette). Free informative catalog of:

- Data receivers and kits
- Quote processing and display software
- Descrambling software utilities

303-223-2120 $5 Demo Diskette

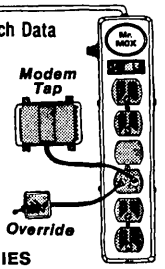
DATArx
111 E. Drake Rd. Suite 7041
Fort Collins, CO 80525

Reader Service Number 133

Mr. MOX $99.95

Modem operated power controller for your PC. Makes any PC with external modem remote-accesssable! Software included.

by Epoch Data



Order From:
KENMORE COMPUTER TECHNOLOGIES
30 Suncrest Dr., Rochester NY 14609 (716) 654-7356

Reader Service Number 135

Why you want BATCOM!

BATCOM is a batch file compiler that compiles your ".bat" files to ".exe" files to make them faster, more professional, and more capable. **BATCOM** extends DOS with new commands so you can read keyboard input, perform arithmetic, use subroutines, and much more. In addition, **BATCOM** protects your source code, and you can distribute your compiled programs without royalties. For IBM PC. **$49.95.**

Wenham Software Company
5 Burley St.
Wenham, Ma. 01984
(508)-774-7036 FREE catalog.

Reader Service Number 124

DID YOU TYPE "CD" TODAY?

DOS users, why crawl around your hard disk with CD when you can JUMP instead? JUMP offers instant four-way access to any area of your hard disk: via menu choice, shorthand or symbolic name, or pathname. JUMP also has remote program execution features that will delight GEM users. Introductory price just $34.95 with special bonus! Write for free info on our complete line of DOS and CP/M programming and word processing products.

Cranberry Software Tools
P.O. Box 681
Princeton Junction, NJ 08550-0681

Reader Service Number 121

LATEST AWARD BIOS
PC/XT ☆ 286 ☆ 386
Support for:

- ✧ Enhanced Keyboards
- ✧ EGA & VGA Graphics
- ✧ 3.5 inch Floppies
- ✧ More...

Authorized AWARD Distributor
(800) 423-3400
KOMPUTERWERK, INC
851 Parkview Blvd
Pittsburgh, PA 15215

Reader Service Number 126

The $25 Network

Try the 1st truly low cost LAN

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

Skeptical? We make believers!

Information Modes
P.O. Drawer F
Denton, TX 76202
817-387-3339

Reader Service Number 149

8051
Z8 / Super8
C COMPILER

*** Call today for a FREE technical bulletin ***
MICRO COMPUTER CONTROL
P.O. Box 275 – Hopewell, NJ 08525 USA
Telex 9102404881 MICRO UQ
(609) 466-1751

Reader Service Number 100

OPT-TECH SORT/MERGE

Extremely fast Sort/Merge/Select utility. Run as an MS-DOS command or CALL as a subroutine. Supports most languages and filetypes including Btrieve and dBase. Unlimited filesizes, multiple keys and much more! MS-DOS **$149.** XENIX **$249.**

(702) 588-3737
Opt-Tech Data Processing
P.O. Box 678 - Zephyr Cove, Nv 89448

Reader Service Number 64

CROSS ASSEMBLERS

PseudoCode releases the PseudoSam professional series of Cross assemblers. All popular processors. Macros, Conditional Assembly, and Include Files. Virtually unlimited size. For IBM-PC's MS-DOS 2.0 or greater with manual $50.00. Simulators and disassemblers also available. (MI res. 4% tax). S&H USA $5, Canada $10, Foreign $15. Visa/MC.

KORE Inc.
6910 Patterson S.E.
Caledonia, MI 49316 616-791-9333.

30 Day satisfaction guaranteed
or purchase price refunded.

Reader Service Number 136

two bytes listed — 00 02 on my system.

This procedure probably has limited usefulness (damage must be confined to the directory and FATs) and only makes sense for text or data file recovery. (It's pretty dang hard to tell where executable files start and end.) And breaking out the individual files from BIG.BIG poses some problems even for pure text files.

Consider what happens when you load this monster into your favorite text editor. The editor cruises along until it sees the ^Z (EOF) at the end of the first text file. And grinds to a halt.

You'll need to massage BIG.BIG with a disk editor like EZZAP or with DEBUG (much more of a pain). Replace the ^Zs with something innocuous like those silly smiley face characters. Now you can easily break out the files.

If you don't want to hassle with making the BIGDIR.DIR and BIGFAT.FAT files, I've placed copies on the Micro C BBS and on the Issue #47 disk.

Larry Fogg
Micro C Staff

Software Fix For AT RTC

If you use an IBM AT or AT clone, you've probably noticed that the computer's CMOS clock isn't very accurate. The real time clock in my AT, for instance, loses about three minutes per month. There isn't much point to a real time clock if you have to reset the system time when you boot the computer.

A peek at the technical reference manual indicates that the CMOS clock is driven by a 32,768 Hz crystal controlled oscillator. The accuracy and stability of this oscillator are responsible for the accuracy of the CMOS clock.

If the crystal frequency varies as little as 1 Hz, the clock will be in error by 73 seconds per month. Crystal tolerances of .001% are easily achievable. Such a crystal would be no more than .328 Hz off frequency in this application, yet could still result in a clock error of 24 seconds each month.

With this sort of sensitivity to crystal frequency, how do you build an accurate clock? There's a panoply of relevant technical issues you'll need to consider if you plan to pursue a hardware fix. Suffice it to say, it's possible to get the oscillator frequency where it belongs and keep it there, if you work at it hard enough.

After all, a ten dollar quartz watch does an admirable job of time keeping, and in less space than that allotted the CMOS clock in an AT. But, you probably don't want to haul out a lot of test equipment and perform surgery on your computer just to make the clock run on time.

Crystal accuracy depends greatly upon the consideration given its fabrication by the manufacturer. Stability, on the other hand, is an inherent property of quartz crystals, a result of the physical laws that govern their operation.

Consequently, while a crystal might be a few Hz from where you want it, you can expect it to stay there. This is fortunate because it allows us to make the CMOS clock run very accurately, without doing anything to the hardware.

The technique is straightforward and easy to grasp. To start with, once and for all we set the CMOS clock to the correct time and date. It immediately commences losing or gaining time. We, however, have noted the time and date on which the CMOS clock was properly set. Additionally, we know the rate at which the clock accumulates error.

Every time the computer is booted up, the CMOS clock will report the time, albeit incorrectly. We know when the clock was set so it's easy to determine how much time the clock thinks has passed in the interim.

Of course, the clock is wrong. But because we're privy to the error rate we can correct this and determine how much time has really passed. If we know this, we know the current time. Once we have the correct time, we pass it to the system through the BIOS to set the time and date to their correct values.

A couple of assumptions: 1) You know, or have available to you, a method for accurately determining the clock error rate; 2) This error is constant over a long period of time.

You can obtain a credible determination of the CMOS clock error using fairly obvious, but tedious, methods. The second assumption is, in fact, just that; you have to assume a constant error rate. The discussion on crystal stability justifies this as reasonable.

Two C programs put all this together. The one called fixclock performs the correction. (See Figure 1.) Invoked from within your autoexec.bat file upon boot up, its operation is transparent and its only action is to set the correct system time.

You can select either local or Greenwich Mean Time as the format passed to the system. It knows whether or not daylight savings time is in effect, setting the system time appropriately if you've selected the local time option. Details are provided in the source file.

The program clockerr (on the Micro C Issue #47 disk and BBS) lets you extract the clock error rate as painlessly as possible and also generates the other clock statistics you'll need to put on the fixclock command line.

Operation of both these programs is detailed at the start of their respective source files. One caveat however: when you are prompted for the time by clockerr, it expects you to enter the local time in your time zone.

So if daylight savings time is in effect, the time you enter must reflect this fact. Simple enough, but some parts of the country don't observe daylight savings time.

Never mind, pretend that you do and enter the time accordingly. Later, if you've put the right parameters on the command line for fixclock, the system time will always reflect the correct time, whether daylight savings time is observed or not.

Gregory D. Knox
1132 Knollwood, Apt. A
Schaumburg, IL 60194

♦ ♦ ♦

PC Tech

X16 Upgrade Special!

PC Tech has obtained a limited quantity of very high performance SCSI controllers suitable for upgrading the PC Tech X16A or X16B. This upgrade will dramatically improve the performance of any bus based hard drive system.

1:1 Interleave supported!

Support for large drives!

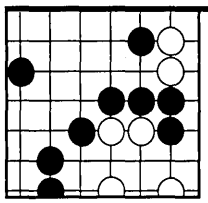
Only $125 complete!

Complete kit includes SCSI interface cable, interface chip, and SCSI controller. Quantities limited. A limited quantity of fast hard drives are also available at very reasonable prices.

Contact: **PC TECH**
907 N.6th St.
Lake City, MN 55041

612/345-4555
or
612/345-5514(FAX)

Reader Service Number 3



Last Page, Continued . . .

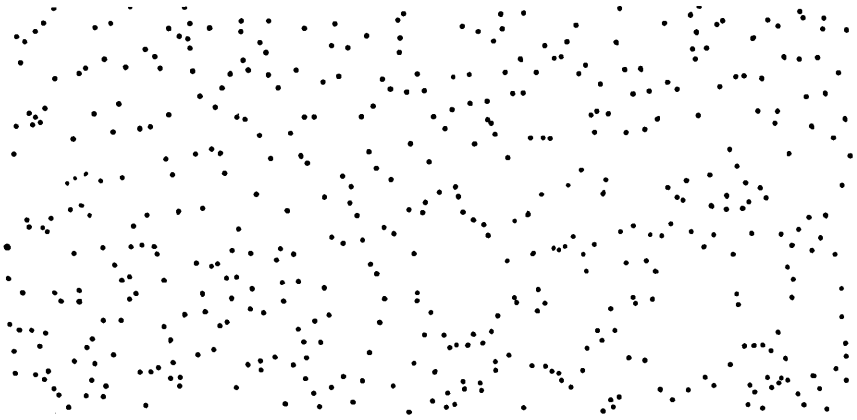


Figure 1 — X vs Time for Chaotic Set.

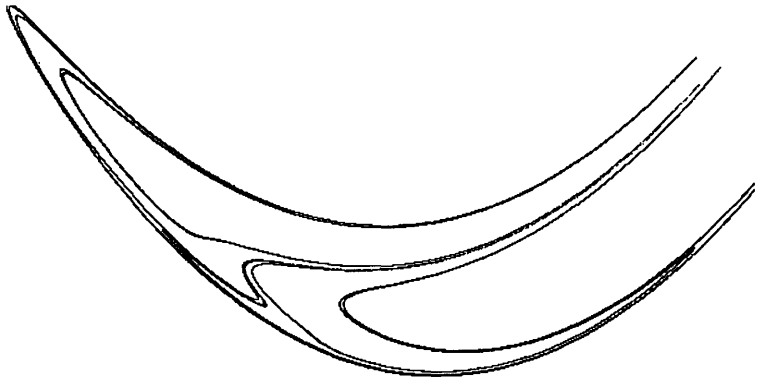


Figure 2 — Strange Attractor for Figure 1.

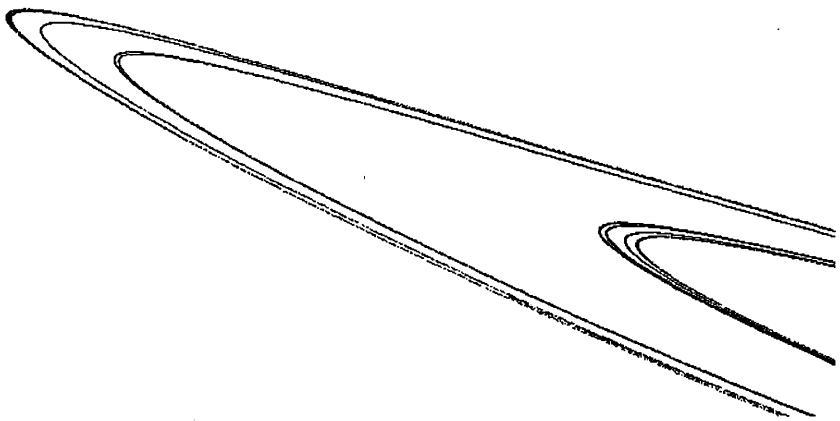


Figure 3 — Zooming in on Figure 2.

agreement, it's "strange." Figure 2 shows the strange attractor corresponding to the chaotic set in Figure 1.

The fractal (or self-similar) nature of the system becomes apparent when we zoom in on any area of the attractor. The deeper we go, the more complex (and detailed) the attractor becomes, and yet the more orderly it seems.

Pictures

I wrote a program in Turbo Prolog using the BGI (Borland Graphics Interface) to generate Figure 1 and the attractors in Figures 2 and 3. I captured and massaged them with PaintShow Plus (from Logitech) and Pizazz (from Applications Technology). I exported them (as .PCX files) to Ventura. For more information about the screen capture, see Tidbits, this issue.

♦ ♦ ♦

68000

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

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

Star-K Software Systems Corp.
P. O. Box 209
Mt. Kisco NY 10549
(914) 241-0287 / Fax (914) 241-8607

Reader Service Number 40

Is There A Gap In Your Info?

Fill in your Back Issues of Micro C today!

ISSUE #1 (8/81)
Power Supply
1/2 PFM.PRN
16 pages

ISSUE #2 (10/81)
Parallel Print Driver
Drive Motor Control
16 pages

ISSUE #3 (12/81)
4 MHz Mods
Configuring Modem 7
Safer Formatter
Reverse Video Cursor
FORTHwords Begins
16 pages

ISSUE #4 (2/82)
Keyboard Translation
More 4 MHz Mods
Modems, Lync, and S10s
Undoing CP/M ERASE
Keyboard Encoder
20 pages

ISSUE #5 (4/82)
Word Processing
Two Great Spells
Two Text Editors
Double Density Review
Scribble, A Formatter
20 pages

ISSUE #6 (6/82)
BBI EPROM Programmer
Customize Your Chars
Double Density Update
Terminal In FORTH
24 pages

ISSUE #7 (8/82)
6 Reviews Of C
Adding 6K Of RAM
Viewing 50 Hz
On Your Own Begins
24 pages

ISSUE #8 (10/82)
SOLD OUT

ISSUE #9 (12/82)
BBII EPROM Program
Relocating Your CP/M
Serial Print Driver
Big Board I Fixes
Bringing Up WordStar
32 pages

ISSUE #10 (2/83)
SOLD OUT

ISSUE #11 (4/83)
SOLD OUT

ISSUE #12 (6/83)
256K for BBI
Bringing Up BBII
dBase II
Look at WordStar
Double Sided Drives for BBI
Packet Radio
5 MHz for Kaypro
40 pages

ISSUE #13 (8/83)
CP/M Disk Directory
More 256K for BBI
Mini Front Panel
Cheap Fast Modem
Nevada COBOL Review
BBI Printer Interface
Kaypro Reverse Video Mod
44 pages

ISSUE #14 (10/83)
BBII Installation
The Perfect Terminal
BBI Video Size
Video Jitter Fix
Slicer Column Begins
Kaypro Color Graphics Review
48 pages

ISSUE #15 (12/83)
Screen Dump Listing
Fixing Serial Ports
Playing Adventure
SBASIC Column Begins
Upgrading Kaypro II To 4
Upgrading Kaypro 4 To 8
48 pages

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

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

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

ISSUE #19 (8/84)
Adding Winchester To BBII
6 MHz On The BBI
Bulletin Boards
Track Buffering On Slicer
4 MHz For The 820-I
64 pages

ISSUE #20 (10/84)
HSC 68000 Co-Processor
DynaDisk For The BBII
Serial Printer On BBI Sans S10
Cheap & Dirty Talker For Kaypro
Extended 8" Single Density
72 pages

ISSUE #21 (12/84)
Analog To Digital Interface
Installing Turbo Pascal
Low Intensity BBI Video
Turbo Pascal, The Early Days
80 pages

ISSUE #22 (2/85)
Xerox 820-II To A Kaypro-8
Sound Generator For the STD Bus
Reviews Of 256K RAM Expansion
In the Public Domain Begins
88 pages

ISSUE #23 (4/85)
Automatic Disk Relogging
Interrupt Drive Serial Printer
Low Cost EPROM Eraser
Smart Video Controller
Review: MicroSphere RAM Disk
Future Tense Begins
86 pages

ISSUE #24 (6/85)
C'ing Into Turbo Pascal
8" Drives On The Kaypro
48 Lines On A BBI
68000 Versus 80x86
Soldering: The First Steps
88 pages

ISSUE #25 (8/85)
Why I Wrote A Debugger
The 32-Bit Super Chips
Programming The 32032
Modula II
RS-232C: The Interface
104 pages

ISSUE #26 (10/85)
Inside ZCPR3
Two Megabytes On DSI-32
SOG IV
The Future Of Computing
Graphics In Turbo Pascal
104 pages

ISSUE #27 (12/85)
SOLD OUT

ISSUE #28 (2/86)
Pascal Runoff Winners
Rescuing Lost Text From Memory
Introduction To Modula-2
Inside The PC
104 pages

ISSUE #29 (4/86)
Speeding Up Your XT
Importing Systems From Taiwan
Prototyping In C
C Interpreters Reviewed
Benchmarking The PCs
104 pages

ISSUE #30 (6/86)
PROLOG On The PC
Expert Systems
Logic Programming
Building Your Own Logic Analyzer
256K RAM For Your 83 Kaypro
PC-DOS For Non-Clones
104 pages

ISSUE #31 (8/86)
RAM Resident PC Speedup
Practical Programming In Modula-2
Unblinking The PC's Blinkin' Cursor
Game Theory In PROLOG and C
104 pages

ISSUE #32 (10/86)
Public Domain 32000: Hardware And Software
Writing A Printer Driver for MS-DOS
Recover A Directory By Reading & Writing Disk Sectors
96 pages

ISSUE #33 (12/86)
SOLD OUT

ISSUE #34 (2/87)
SOLD OUT

ISSUE #35 (4/87)
SOLD OUT

ISSUE #36 (6/87)
Mouse Control
Build A Midi Interface
For Your PC
Designing A Database, Part 2
Interrupts On The PC
Digital To Analog Conversion, A Designer's View
96 pages

ISSUE #37 (9/87)
Desktop Publishing On A PC
Build Your Own Hi-Res Graphics Scanner For $6, Part 1
Designing A Database, Part 3
Controlling AC Power From Your PC
Expanded Memory On The PC/XT/AT
Uninterruptable Power Supply For RAM Disks
96 pages

ISSUE #38 (11/87)
Parallel Processing
Laser Printers, Typesetters And Page Definition Languages
Build A Graphics Scanner For $6, Part 2
Writing A Resident Program Extractor In C
96 pages

ISSUE #39 (1/88)
PC Graphics
Drawing The Mandelbrot And Julia Sets
Desktop Graphics
Designing A PC Work-station Board
Around the TMS-34010
96 pages

ISSUE #40 (3/88)
The Great C Issue
11 C Compilers
Writing A Simple Parser In C
C++, An Object Oriented C
Source Level Debugger For Turbo C
96 pages

ISSUE #41 (5/88)
Artificial Intelligence
3-D Graphics
Neural Networks
Logic Of Programming Languages
Applying Information Theory
96 pages

ISSUE # 42 (6/88)
Maintaining PCs
Keeping Your Hard Drives Running
Troubleshooting PCs
XT Theory of Operation
Simulating A Bus
Ray Tracing
96 pages

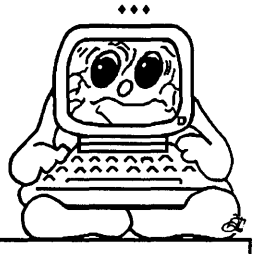
ISSUE #43 (9/87)
Building Databases
Build a C Database
Selecting a dBase III Compatible Compiler
Working with Paradox
Designing Custom PC Cards
Accessing dBase III Plus Records from Turbo Pascal
96 pages

ISSUE#44 (11/88)
Object-Oriented Programming
A Taste of Smalltalk
Actor
Thinking Objectively
Building MicroCad
Peripheral Technology-PT68K-2
Hercules Graphics Printer Dump
96 pages

ISSUE #45 (1/89)
Computer Aided Design
CAD In A Consulting Business
Choosing PCB Layout Systems
Building Circuits With Your Computer
Secrets of Optimization
Finding Bargains in the Surplus Market
MASM 5.1
96 pages

Issue #46 (3/89)
Software Tools
The Art of Disassembly
Handling Interrupts With Any C
Hacking Sprint: Creating Display Drivers
Greatest C Compilers
Turning A PC into An Embedded Control System
Practical Fractals
96 pages

♦♦♦



To Order:

Phone: **1-800-888-8087**
Mail: **PO Box 223**
Bend, Oregon 97709

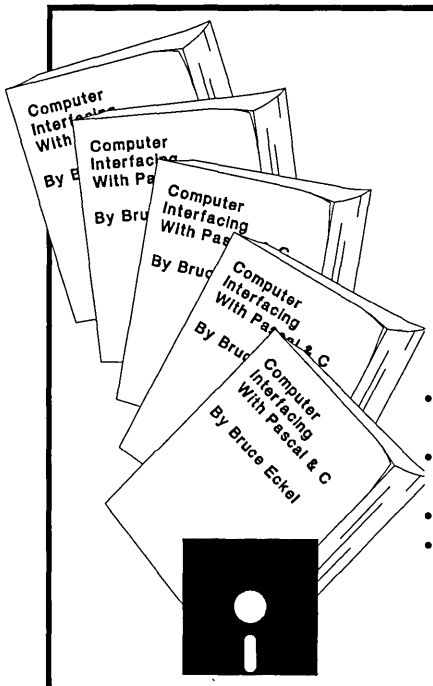
United States,	
Issues #1-34	$3.00 each ppd.
Issues #35-current	$3.95 each ppd.
Canada, & Mexico	
All issues	$5.00 each ppd.
Foreign (air mail)	
All Issues	$7.00 each ppd.

ADVERTISERS INDEX

Issue 47

** Contact Advertiser Directly.

When you write for information, please tell these folks you read about their products in **Micro Cornucopia.**



Now Available From Micro C

Computer Interfacing with Pascal & C
by Bruce Eckel

- Use your PC parallel port for digital input and output
- Build an Adapter Card for your PC
- Control a stepper motor
- Design and build electronic circuits

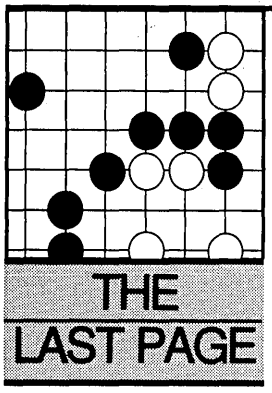
"With wit and superb technical figures, Bruce captures the essence of making electrons out of bits and vice versa."
Jeff Dunteman, *Dr. Dobbs*

Only $30 ppd.
Includes Book & Disk

Order From:
Micro Cornucopia
PO Box 223
Bend, OR 97709
1-800-888-8087

Issue #48
Tools for Physically Impaired

- **Disabled Folk and Computers**
- **Three State of the Art OCR Scanners**
- **PCX Compatibility Issues (Trying to Make a Standard)**
- **Fast File Transfer Via the Parallel Port**
- **The Terminal Diet**
- **Hardware Control of Video Attributes**



By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Strange Attractors

Order In Chaos

I always thought that publishing a magazine might lead to chaos. All fractals aside, it finally has.

Some folks think chocolate cake's the main attraction. Others prefer a pretty face, the Lakers, a stimulating conversation, a sun-ripened tomato, or a stream gleaming with mountain trout. Nothing particularly strange about these attractors, I hope.

But consider something slightly more technical — the attractors which live in "state" or "phase" space, a curious, abstract place for describing "state" or "dynamic" systems.

A dynamic system can be anything:

- we can describe by knowing the values of variables;
- whose current state depends on its previous state.

The system doesn't have to be quantifiable. For example, we could easily conjecture a system where logical changes (is it yes? is it on? has something happened since we checked last?) determine the next state.

Sentences

In a speech recognition system, for example, each word in the sentence not only carries its own weight, but affects (and is affected by) all the other words in the sentence.

Consider a sentence as a dynamic system which changes state through the addition of words and punctuation marks —

Initially we have nothing (an empty state, or a "no sentence" state). We add a word (the one word is the new sentence). We add a word (the two words are the new sentence). We add a word (the three words are the new sentence). We add a punctuation mark (three words + punct = new sentence). And so on.

When we speak, words necessarily follow each other. When we write, they follow or displace one another (by replacing or being inserted between words). Each new state represents an attempt to clarify the sentence. (But as many of you know, each new state can clarify or confuse — one reason that parsing sentences is so difficult.)

The sentence in its next state is very sensitive to its current state (or condition).

We can say that "meanings" are the attractors in the dynamic system, "sentence." At any state the sentence may have no meaning (an extinction state), one meaning (a stable state), or many meanings (possibly a chaotic or pun state), depending on any number of things — point of view, reading or writing skill, etc.

Mathematic Attraction

Folks from many diverse fields (mathematics, physics, engineering, biology, computer science, business, economics, etc.) are trying to understand dynamic systems. Their discoveries have led to at least one agreeable conclusion — anything of interest which changes state is incredibly complex.

Mathematicians and computer scientists try to twist meaning out of complex systems by representing them with equations (or rules) and pictures.

Some particularly useful pictures exist in phase (or state) space, where each point holds all the information needed to describe a dynamic system at any one time (or state).

For example, suppose a system varies (changes state) depending on a variable — such as size or number. Call the variable "X," and call its rate of change "R." Then equations like —

```
Nextx = RX(1-X)
```

can describe the system.

Here, 1 represents unity (the entire system or all we can have of it) and 0 represents nothing (the system devoid of occupants). Either extreme state (when Nextx = 1 or 0) translates into oblivion, since there's either nothing left to act on the current state (when Nextx = 0) or nothing left to act on (when Nextx = 1, and Nextx is everything!).

The previous equation, the so-called standard map or logistic equation, has been well-studied by mathematically-inclined folks in many fields. They've discovered that it (and presumably the dynamic system it describes) behaves unpredictably. In general, any system we describe with a non-linear equation or equations will behave unpredictably.

They're unpredictable because they're extremely sensitive to initial conditions. Nearby values of X in one state may lead to values (of Nextx), which are far apart in the next state. (Or even the next county.)

We can complicate matters even more by increasing the number of variables in a system (i.e., our representation of a system). For example, these two rules for changes in state —

```
Nextx = AX - (1-Y²)
Nexty = BY
```

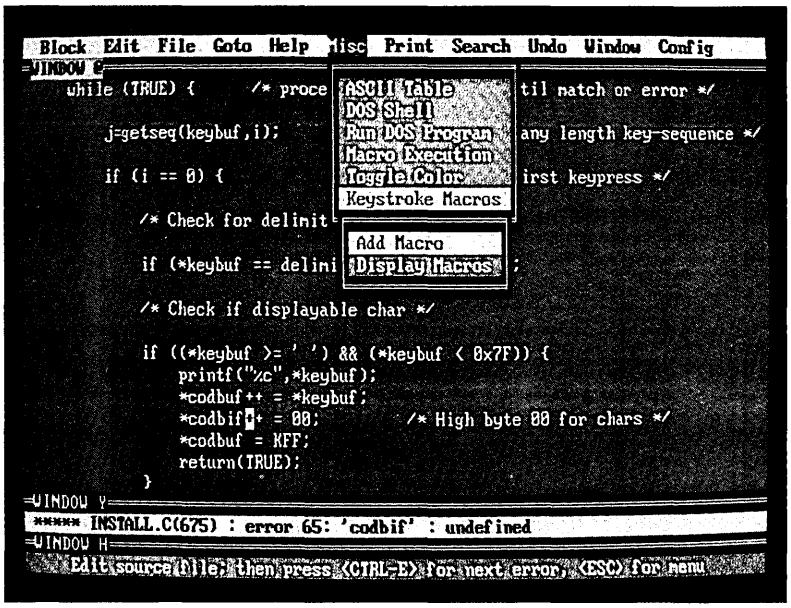
can present an infinite number of states (Nextx values) responding to infinitely small changes in the condition of the previous state (or X value). This infinity of values is the chaotic set for this dynamic system.

We can see this chaos easily by plotting the values of each state (Nextx) in time. (The X axis is time. The Y axis is the value of each X. See Figure 1.)

Order In Chaos

Yet, there's a certain order in the most chaotic systems. One way to see this order is in phase space. When we plot values of Nextx against values of current X, we uncover the attractor for the chaotic set. This attractor lives in phase space and consists of all the points in the chaotic set. By mutual

Continued on page 93



VEDIT PLUS

#1 PROGRAMMABLE EDITOR

NEW VERSION 3.0

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

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

Until now, if you wanted the best Undo, the best compiler support, regular expressions and column blocks you chose BRIEF™. If you wanted unlimited keystroke macros, the best configurability, "off the cuff" command language macros and blazing speed, you chose VEDIT PLUS.®

Now the Choice is Easy

The all new VEDIT PLUS 3.0 gives you the best Undo of any editor, the best compiler support, unequaled windows, true regular expressions and extensive new features. We're leading the way with easy to use pull down menus, context sensitive help, a pop-up ASCII table, new printing options and much more. Incredibly, VEDIT PLUS 3.0 is now twice as fast as before and, at only 60K in size, it loads fast!

Completely Configurable

Change a few keys or redefine the entire keyboard, VEDIT PLUS adjusts to your editing style in minutes. You can even create new editing functions using simple keystroke macros or fine tune existing ones. VEDIT PLUS is so configurable that it easily emulates other editors and word processors (WordStar and Word Perfect emulation included). Quickly access editing functions with a single key or through the pull-down menus.

Try before You Buy

We challenge you to experience the dazzling performance and exceptional features that make VEDIT PLUS the best choice. Our evaluation disk includes the complete editor.* Learn VEDIT PLUS using our extensive "training" macro that gives instructions in one window while you experiment in another. See for yourself why no other macro language comes close.

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

Supports the IBM PC, XT, AT and PS/2 including DESQview, Microsoft Windows, PC-MOS/386, Concurrent DOS and most networks. Also available for MS-DOS, CP/M-86 and FlexOS. (Yes! We support windows on CRT terminals.) $185.

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

Features of VEDIT PLUS 3.0

- Simultaneously edit up to 37 files of unlimited size.
- Variable sized windows; multiple windows per file.
- Execute DOS commands and other programs.
- Flexible "cut and paste" with 36 "scratch-pad" buffers.
- Block operations by line, character or column.
- Search with pattern matching or regular expressions.
- Configuration—determine your own keyboard layout, create your own editing functions, support any screen size.
- Select window colors, support 43 line EGA, 50 line VGA.

EASY TO USE

- Modern pull-down menu system. Pop-up ASCII table.
- Context sensitive on-line help is user changeable.
- Multi-level Undo (100 to 1000 levels). Undo keystroke by keystroke or line by line.
- On-line integer calculator (also algebraic expressions).
- Keystroke macros speed editing, menu function "hot keys."

FOR PROGRAMMERS

- Automatic Indent/Undent for "C," PL/I, PASCAL.
- Match/check nested parentheses, e.g. "{" and "}" for "C."
- Flexible macro runs popular compilers and automatically moves cursor to each error in your program. Easily changed to support new compilers and assemblers.

FOR WRITERS

- Word wrap, paragraph formatting and justification.
- Convert to/from Wordstar and mainframe files.
- Flexible printing; fully adjustable margins and Tab stops.

MACRO PROGRAMMING LANGUAGE

- If-then-else, looping, testing, string compare, branching, user prompts, keyboard input, 24 bit algebraic expressions.
- Flexible windowing—forms entry, select size, color, etc.
- Simplifies complex text processing, formatting, conversions.
- Extensive 400 page manual with hundreds of examples.

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

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

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

Reader Service Number 7

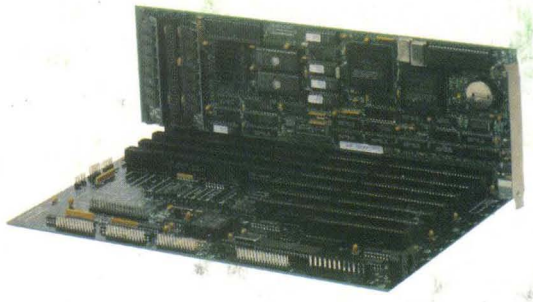
CompuView

VERY HIGH PERFORMANCE
Processors, Memory, and Display Adapters

The X24 High performance processor

- 12 or 16 MHz 80286 with NO WAIT STATES!
- Small size ("XT" height and length) passive bus design
- 1 to 4 Mbyte 0 wait state dynamic memory
- Fully "AT" compatible Award BIOS
- Runs DOS versions 2.2 and later, Xenix and OS/2

The X24 combines the best of motherboard and backplane designs in a 100% AT compatible system. Incorporating a 16 MHz 80286, the X24 processor is designed to operate with the PC Tech Advanced System Motherboard, which contains the peripheral interfaces (hard disk, floppy disk, two serial ports and a parallel port). The X24 processor can also be used with other totally passive bus backplanes. Most critical components including the microprocessor and up to 4 megabytes of fast memory are contained on a single PC size plug-in card. This allows the processor and main system memory to be serviced or upgraded without disturbing other peripherals such as serial ports and disk drives.



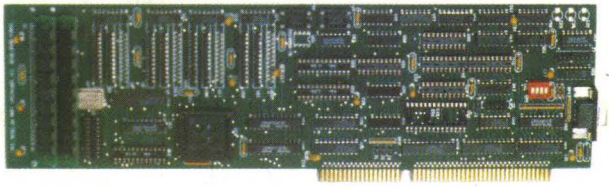
PC Tech X24 and ASMB

The PC Tech Advanced System Motherboard

- Built in "IDE" interface for AT interface type hard drives
- Fully AT compatible floppy disk support for 3.5", 5.25" drives, capacities of 360k, 1.2m and 1.44m
- Two serial ports and one parallel port
- 8 total expansion slots PC/XT/AT compatible (4 slots have 32 bit bus)

The PC Tech Advanced System Motherboard is designed to complement PC Tech's X24 and X32 high performance processor cards. It contains the mass storage interfaces necessary for a complete system, plus the basic I/O required in most systems. Extra care has been given to FCC compliance by design.

34010 Monochrome Graphics Adapter II



PC Tech Mono-II

- Up to 384k bytes display memory
- Up to 2 Megabytes program memory
- Software is RAM based, allowing complete operating software replacement and timing re-programming from the host bus
- 34010 program loader included. Assembler, debugger, and C compiler available.
- Full hardware and software CGA, MDA and Hercules emulation
- Single bit shared memory bit-map with optional resolution up to 2048 x 1536 (736 x 1008 standard)
- Very high resolution COLOR version available
- Custom 34010 software development available

The TMS34010 is a true general purpose graphics processor. PC Tech makes the total processing power of the 34010 available to both programmers and end users. Our 34010 Monochrome Graphics Adapter is designed to allow programming from the PC/XT/AT host bus. You can completely replace our 34010 software with yours to directly harness the incredible image processing power of the TMS 34010 for your application. We make a complete set of development tools available, including an assembler, C compiler, program loader, 34010 debugger, and PC interface tracer/debugger. Our standard product includes support for extended CGA, MDA and Hercules emulation as well as a host addressable graphics bit- map. We also support and recommend the DGIS graphics interface standard (from Graphic Software Systems) for applications development as an alternative to native 34010 software development. Ready to run drivers are available for most major applications software packages as well.

Custom Designs Available

PC Tech will license most products for non-exclusive manufacture. We will also customize any of our designs to better meet your needs on our in-house CAD systems. All of our standard products are available in private label versions.

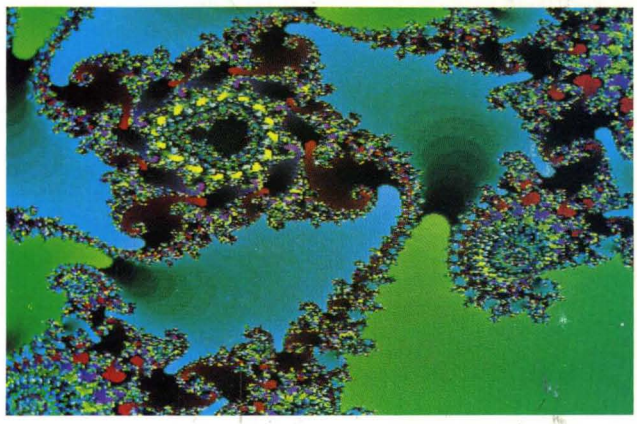
About PC Tech

PC Tech has been designing, manufacturing and marketing high performance PC related products for over three years. Our standard product line includes processor, memory, and video products. All products are designed, manufactured and supported in our Lake City, Minnesota facilities.

Designed, Sold and Serviced By:

PC Tech

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



High resolution fractal produced on the PC Tech COLOR 34010

Reader Service Number 3

PC, XT, AT, DGIS, Hercules, and Award BIOS are trademarks or registered trademarks of their respective companies.