No. 38 Nov./Dec. 1987 $3.95

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Parallel Processing

This issue we have three articles on the transputer, a processor specially designed for parallel processing. The articles cover the transputer, communications between transputers, and creating a parallel C compiler.

Series begins on page 6

Laser Printers, Typesetters And Page Definition Languages page 20

The problems designers (and purchasers) face getting information onto paper.

Magic In The Real World page 28

What can you do with a $20 PC parallel card? A lot! Bruce tells you how.

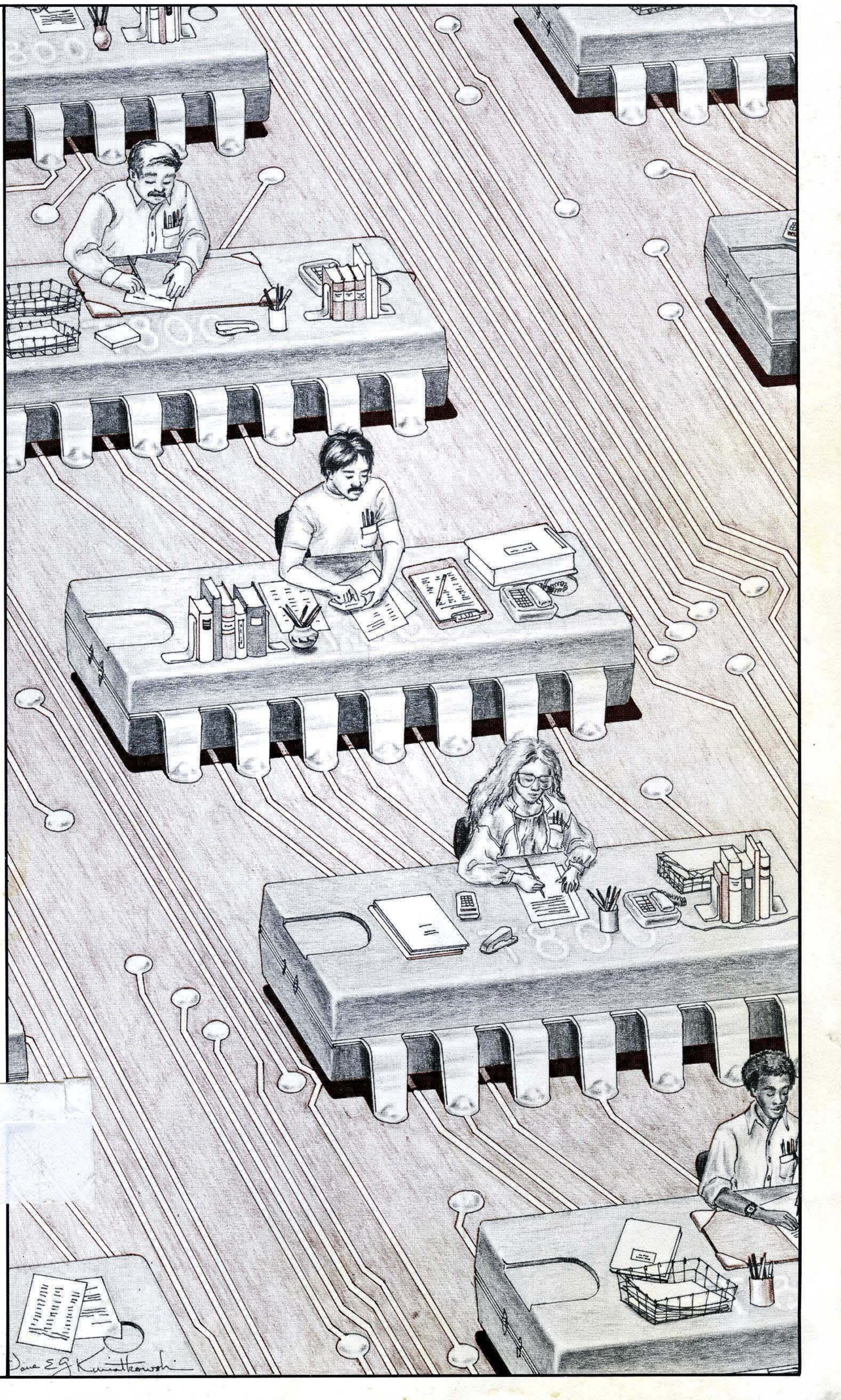
Build A Graphics Scanner For $6.00, Part 2 page 42

This time, John covers the hardware construction and begins the software.

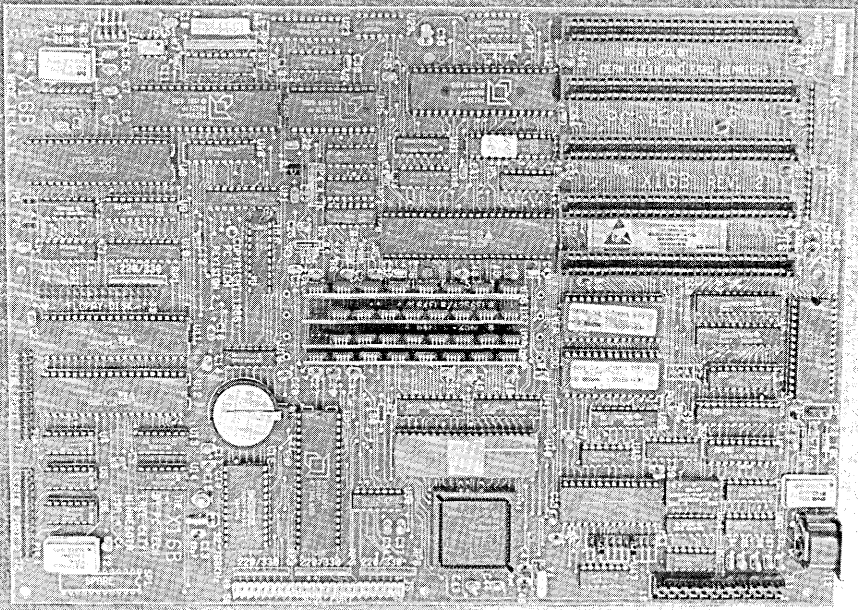
In Depth Turbo C page 56

Writing a resident-program extractor entirely in C.

0 74470 19388 3



10 MHz, 0 Wait States, American Made!



The X16B offers the highest performance and integration of any PC/XT compatible. With its 10 MHz, zero wait state operation it walks away from AT compatibles as well. On board is one megabyte of DRAM, a real time clock, floppy disk controller, and optional one or two serial ports, SCSI port and 8087.

The PC Tech SmartBIOS provides PC compatability with ease of use. We wrote it and we support it!

OTHER PRODUCTS

8087-1 Math Coprocessor for X16B

The X16 8087 Math Coprocessor runs at full CPU speed. That's 10MHz of number smashing power! The Math Coprocessor on the AT only runs at 2/3 of the CPU clock.

SCSI Port Option for X16B

Full SCSI port using the 5380. Software built into ROM BIOS for the OMTI 3100 hard disk controller achieves a 1 to 1 sector interleave.

FOUR MEGGER

Running out of spreadsheet room? Need something faster than a hard disk for those long compiles? The Four Megger is the answer. The Four Megger meets the Lotus/Intel/Microsoft expanded memory specifications and works in all PC and XT computers. The Four Megger also works in AT computers as expanded memory only.

NEW

16 MEGGER!

Expanded or Extended Memory for PC/XT/AT. Up to 16 Megabytes total. Expanded memory works as on FOUR MEGGER.

NEW

TMS 34010 HI-RES COLOR BOARD

Bring workstation graphics to your PC! This board has its own 32-bit graphics CPU running at 5 MIPS, with up to 4 M Bytes of memory plus 1 M byte display memory. Emulates CGA. Up to 1024 by 800 pixels, 256 colors from a palatte of 256k!

PRICES!

X16B / 1 Meg / RTC	$600.00
8 MHz version	$540.00
SCSI option	$25.00
Serial ports (2)	$38.75
8087-1/82188	$340.00
8087-2/82188	$260.00
Four Megger	~~$850.00~~ NOW $750.00
2S1P	$99.00
Memories	$95.00
Systems:	
The Box	$799.00
Mono System	$1,099.00
EGA System	$1,849.00

Call for system configuration details. PC Tech also sells hard drives and controllers, video boards and monitors, etc.

PC Tech

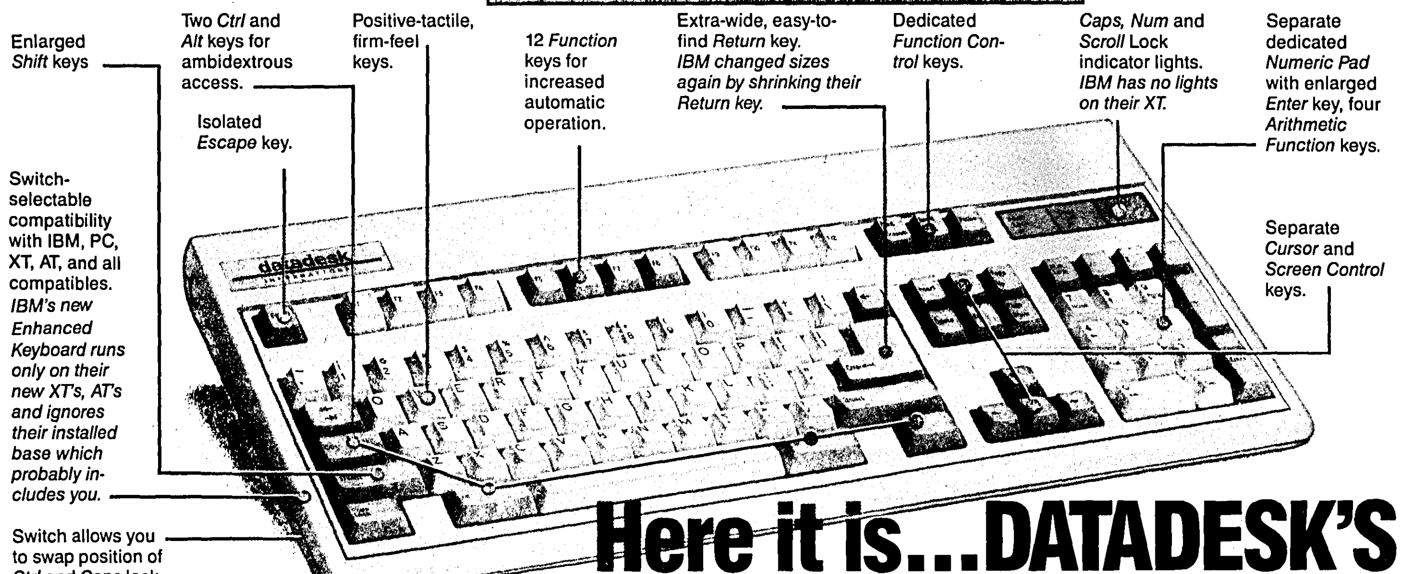
P.O. Box 128
904 North 6th Street
Lake City, MN 55041
(612) 345-4555

Watch for more innovative products designed and built by PC Tech!!!

COD, VISA, MASTER CARD, CERTIFIED FUNDS

PC, XT and AT are trademarks of International Business Machines

Borland's Turbo Lightning FREE

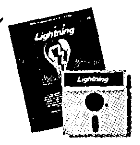

Enlarged *Shift* keys
Two *Ctrl* and *Alt* keys for ambidextrous access.
Positive-tactile, firm-feel keys.
Isolated *Escape* key.
12 *Function* keys for increased automatic operation.
Extra-wide, easy-to-find *Return* key. *IBM changed sizes again by shrinking their Return key.*
Dedicated *Function Control* keys.
Caps, Num and *Scroll* Lock indicator lights. *IBM has no lights on their XT.*
Separate dedicated *Numeric Pad* with enlarged *Enter* key, four *Arithmetic Function* keys.
Separate *Cursor* and *Screen Control* keys.
Switch-selectable compatibility with IBM, PC, XT, AT, and all compatibles. *IBM's new Enhanced Keyboard runs only on their new XT's, AT's and ignores their installed base which probably includes you.*
Switch allows you to swap position of *Ctrl* and *Caps* lock. *IBM moved Ctrl to bottom row, you have no choice!*
datadesk

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

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

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

Get Borland's Turbo Lightning ™ For FREE!



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

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

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

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

SPECIAL OFFER!

ONLY $149.95 FOR BOTH KEYBOARD & SOFTWARE*

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

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

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

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

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

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

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

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

The Only Alternative

datadesk™ INTERNATIONAL

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

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

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

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

MADE IN USA

BOTH TURBO-101 ENHANCED™ KEYBOARD AND BORLAND'S TURBO LIGHTNING™ SOFTWARE FOR ONLY:

$149.95* LIMITED OFFER

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

NO. UNITS:______
PAYMENT: ☐VISA ☐MC ☐CHECK
AMOUNT ENCLOSED: $______

Enter

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

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

Editor & Publisher
David J. Thompson

Associate Editors
Gary Entsminger
Cary Gatton

Technical Department
Larry Fogg

Director of Advertising
Laura Logan

Distribution Manager
Tracey Anthony

Accounting
Sandra Thompson

Order Department
Tammy Westfall

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

SUBSCRIPTION RATES:

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

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

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

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

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

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

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

AROUND THE BEND

By David Thompson

Parallel Editorial

Last year I was vaguely curious about parallel processing. This year it's a lot more interesting. So interesting, in fact, that I tried it at home:

- Erin was to clear the table and put leftovers in the fridge.
- Sandy was to wash.
- Jennifer's task was drying and putting away the dishes.
- I was the supervisor/task scheduler.

It seemed like the perfect time, since it was my night to do dishes.

What happened, however, was not pretty. By the time I'd finished proposing the experiment, Erin had disappeared into her room to feed Popcorn (her pet rat), Sandy had tossed me the sponge and headed downstairs to finish the illustrations for this issue, and Jennifer... well, Jennifer is a teenager.

This experiment left no doubt in my mind that the supervisor has the most difficult task. And, there's no guarantee that a task will be handled any faster on a parallel system than on a workable one. Especially if there are teenagers in the house.

However, mine was not the definitive study in parallel processing. I had no dedicated hardware, and no special language tools. Using: "You fed your rat an hour ago," doesn't seem to work.

Anyway, for a more definitive look at parallel processing, check out the three transputer articles in this issue. They are from Bernt Roelofs, Hans Bieleman, Klaas Wijbrans, and Rob Kurver, students of Andy Bakkers at Twente University in Holland. Better yet, find a friend and read the articles in parallel.

Hard Drives (Again)

After the article in issue #36, I've gotten some updates.

First, Microscience called and asked if they could send me a drive. I said sure. But, I specified that it had to be off-the-shelf, and it had to be their cheap 20-meg model, the one I'd reported was having problems.

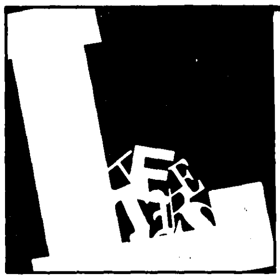
They sent a 725 (20 meg 85 ms), and I'm certain they followed my instructions to the letter. When we applied power, the drive spun up, then shut itself down. Permanently.

(continued on page 88)

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA

NOV./DEC. 1987 - ISSUE NO. 38



Letters

Polyboost for the Kaypro 2000

Having had such good experience with Kaypro's CP/M machines, it was only natural that I should consider their K2000 when I branched out into MS-DOS. I've had a lot of success with the 2000 in spite of its underwhelming screen and disk I/O speed. I had been hoping that some speedup scheme would eventually crop up.

After a year of waiting for a hardware solution, I came across a neat product called Polyboost from Polytron. As I had been pleasantly impressed by Polytron's PolyWindows Plus, I decided to order Polyboost on the blind.

I've been very happy with the improved keyboard, screen, and floppy performance provided by this software accelerator. Even an already fast package like PC-Write runs noticeably quicker. Kaypro, if you're listening, why not package Polyboost with the 2000?

Polyboost is available for $79.95 from:

Polytron Corp.
1815 NW 169th Pl. Suite 2110
Beaverton, OR 97006
(800) 547-4000

Hans Austermuhle
Sierra Leona 650
Lomas Chapultapec
11000 Mexico D.F.

Like, Wow

I like your new cover. I liked your old cover. I even liked the cover before that. I like the cover because you write a good magazine and whatever it looks like, I enjoy seeing *Micro C* arrive in the mail.

It's very much like the way I feel about my wife's hair - I like however she wears it because I like her.

Roger Paige
KAC
Berry Hill
Frankfort, KY 40601

Challenger Problems

Bob G. Roberts' letter in *Micro C's* February-March issue (#34) was just great. Not so much for the letter but for the fact that you allowed him to voice his problem. I had similar problems with a mail order firm which not only sold me a mother board plagued by incompatibilities, but also refused to repair the board when it proved defective.

I wrote three letters directly to them and one to the publisher of *Computer Shopper* (in which they frequently advertise). I've done everything I could short of flying to their door and Vaselining the handle.

The editorial in the April-May *Micro C* (#35) proved to be fertile ground for me since you mention the board sold by Challenger Computer (the villain in my story). This board in 10 MHz dress shows many software incompatibilities. It also proved to be useless with various hardware additions.

In my opinion, it would be best to avoid doing business with Challenger Computer. They may have solved their compatibility problems, but their business philosophy remains unchanged.

If your readers would like additional opinions on clones and compatibility, they can write my company. We are offering free photocopies of a pamphlet we publish on this issue. Please enclose a couple of stamps to help cover postage.

Robert L. Sabaitis
Lee Consulting
14040 Salem
Redford, MI 48239

Editor's note: You could probably clean up many of the hardware incompatibilities by slowing down the system from 10 MHz to 8 MHz. The software problem is a BIOS problem. I don't have a cure for that.

Where's The Mag???

Alas, sadness and despair prevail. The bimonthly rendezvous with joy and true happiness has not occurred despite being six whole days into the dreary month of expectation. Not that anyone's existence should rest in its entirety on receipt of a single periodical...

Prose aside, may I simply report that I have not received the August-September issue as of yet, and I miss it sorely. Lost in the mails, no doubt. Evil mail carriers!

Yes, I confess. I am one of those who tosses the rest of the world aside while greedily reading each newly-arrived issue from cover to cover in one sitting. Then, having gorged and reveled in exquisite ecstasy, I wait impatiently for the rising of the second moon. Oh good people, hear the plea of this your humble reader; PLEASE send my issue with no further delay. I'll do (almost) anything. I'll learn Forth. I'll enjoy Pascal. I'll even give up C for 20 minutes (shudder).

David W. Stojan
10310 Lybert Rd.
Houston, TX 77041

Editor's note: Yes, we were a bit behind with issue #37, but rest assured that by the time you receive this issue (#38), you will probably have #37 as well. We were having an awful lot of fun at SOG. (And all you can think about is the magazine. Aren't you ashamed?)

Unconceived Program Destroyed

Elle M. Enno's program sounded so good that some deviant created a Trojan Horse with the same name and number of bytes. The Fido Net people felt that the only way to protect large numbers of people from losing their FATs was to kill it on every BBS. I believe they were successful and all real and phony copies have been destroyed.

Grant Raddon
2806 NE 11th
Portland, OR 97212

(continued on page 92)

More Magic from Blaise. Turbo C TOOLS™

Magic is easy with Turbo C TOOLS in your bag of tricks. New Turbo C TOOLS™ from Blaise Computing is a library of compiled C functions that allows you full control over the computer, the video environment, and the file system, and gives you the jump on building programs with Borland's new C compiler. Now you can concentrate on the creative parts of your programs.

The library comes with well-documented source code so that you can study, emulate, or adapt it to your specific needs. Blaise Computing's attention to detail, like the use of function prototyping, cleanly organized header files, and a comprehensive, fully-indexed manual, makes Turbo C TOOLS the choice for experienced software developers as well as newcomers to C.

Turbo C TOOLS provides the sophisticated, bullet-proof capabilities needed in today's programming environment, including removable windows, "sidekickable" applications, and general interrupt service routines written in C.

The functions contained in Turbo C TOOLS are carefully crafted to supplement Turbo C, exploiting its strengths without duplicating its library functions. As a result you'll get functions written predominantly in C, that isolate hardware independence, and are small and easy to use.

Turbo C TOOLS embodies the full spectrum of general purpose utility functions that are critical to today's applications. Some of the features in Turbo C TOOLS are:

◆ **WINDOWS** that are stackable and removable, that have optional borders and a cursor memory, and that can accept user input.

◆ **INTERRUPT SERVICE ROUTINE** support for truly flexible, robust and polite applications. We show you how to capture DOS critical errors and keystrokes.

◆ **INTERVENTION CODE** lets you develop memory resident applications that can take full advantage of DOS capabilities. With simple function calls, you can schedule a Turbo C function to execute either when a "hot key" is pressed or at a specified time.

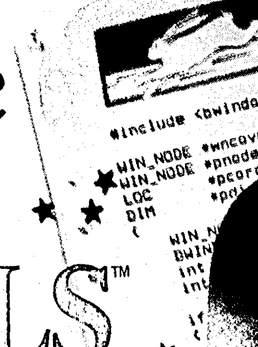
◆ **RESIDENT SOFTWARE SUPPORT** lets you create, detect, and remove resident utilities that you write with Turbo C TOOLS.

◆ **FAST DIRECT VIDEO ACCESS** for efficiency, and support for all monitors including EGA 43-line mode.

◆ **DIRECTORY AND FILE HANDLING** support let you take advantage of the DOS file structure, including volume labels and directory structure.

In addition to Turbo C TOOLS, Blaise Computing Inc. has a full line of support products for Microsoft, Lattice and Datalight C, Microsoft Pascal and Turbo Pascal. Call today for details, and make magic!

Turbo C TOOLS supports the Borland Turbo C compiler, requires DOS 2.00 or later and is just $129.00



```
#include <bwindow.h>
WIN_NODE *wncover(pnr
WIN_NODE *pnode1
LOC      *pcorn
DIM      *pd
WIN_N
BWIN
int
int
```

BLAISE
[illegible]N U

[illegible]OWER TOOLS PLUS *$99.95*
Screen and window management including EGA support; DOS memory control; ISRs; scheduled intervention code; and much more. For Turbo Pascal.

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

Turbo ASYNCH PLUS *$99.95*
Interrupt driven support for the COM ports. I/O buffers up to 64K; XON/XOFF; up to 9600 baud; modem and XMODEM control. For Turbo Pascal.

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

C TOOLS PLUS *$175.00*
Windows; ISRs; screen handling; multiple monitors; EGA 43-line text mode; direct screen access; DOS file handling and more. For MS and Lattice C version 3.00 and later.

LIGHT TOOLS *$99.95*
Windows; ISRs; EGA 43-line text mode; direct screen access; DOS file handling and more. For the Datalight C compiler.

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

VIEW MANAGER *$275.00*
General screen control; paint screens; block mode data entry or field-by-field control with instant screen access. For C or MS-Pascal.

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

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

TO ORDER CALL TOLL FREE 800-333-8087
TELEX NUMBER-338139

YES! I want to make magic!
Enclosed is $_____ for _____ copies of _____.
☐ Please send me more information on your products.
CA residents add Sales Tax. Domestic orders add $4.00 for UPS shipping, $10.00 for Federal Express standard air.
Name: _____ Phone: (___) _____ - _____
Address: _____
City: _____ State: _____ Zip: _____
VISA or MC#: _____ Exp. Date: _____

BLAISE COMPUTING INC.
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Turbo C is a trademark of Borland International.

By Bernt Roelofs
Ingenieursbureau Bieware
Witbreuksweg 381-401
7522 ZA Enschede, The Netherlands

The Transputer
A Microprocessor Designed For Parallel Processing

If your interests are anything like those of our SOG attendees, you'll join in on this discussion. At least here, you won't have to stay up well into the morning to get the details.

Computer applications today are demanding more and more processor power. Over the past decade these power demands have resulted in new processors with their powerful instruction sets, large RAM addressing schemes, and higher clock speeds.

Some say there's no limit to single-processor technology, but there is an alternative: parallel processing.

The British firm, INMOS, has designed the transputer, the first commercially-available microprocessor specially designed for parallel processing.

In this article, I'll first look briefly at parallel processing systems and then explore some of the hardware specifics of the transputer. Finally, I'll consider the transputer's instruction set and operation with emphasis on its parallel processing features.

Parallel Processing

Parallel technologies have appeared in mini- and mainframe computers for a long time. Cray, CDC, and Floating Point Systems offer processing units that allow pipelining, concurrent evaluation instructions, arrays, and digital signal processing.

Such systems aren't very flexible because these processors are often designed to solve specific problems. And, in the best cases, the number of processors can't be increased despite the need for greater processing power.

Computing systems distributed over several processing units, so-called MIMD (Multiple Instruction, Multiple Data) systems, are more flexible. MIMD systems frequently employ a number of general-purpose microprocessors (e.g., a Z80 or 68020) so that the system can solve a variety of problems.

But designers of MIMD parallel processing systems must decide how the processors will communicate (interconnect). Most interconnection schemes fall into one of four categories:

- shared-memory systems
- common bus systems
- switched bus systems
- network systems

Shared-memory, common, and switched bus systems all transfer huge amounts of data quickly. But physical layout, memory bandwidths, and/or bus bandwidths restrict their expandability.

Using direct processor-to-processor communications overcomes the expandability problem. Typically, the communications are high-speed serial links. With such a system, it's possible to add more processing units - without violence to the existing network - to gain power.

The precise scheme of interconnection between processors in these multiple processor networks is commonly referred to as the system's "topology." Many topologies exist: they can be structured as a mesh, a cube, hypercubes, or a tree.

Proponents of parallel processing systems based on network architectures also believe that many computing problems can be mapped into subproblems, and that groups of different algorithms can be hidden in "black boxes."

The programmer houses the "black boxes" on different processors in the network. These boxes constitute a kind of "process" or "task-hiding" similar to the programming concept of data hiding. Only the interconnections between the boxes need to be well-defined and understood.

Such a structural approach to problem solving, when applied consistently, can eventually yield a program that's easily implemented on a parallel system with point-to-point communications.

The Transputer Chip

INMOS hopes the transputer chip will be used in much the same way as the discrete transistor was used a few years ago: as the basic building block of larger systems. Indeed, the word transputer derives from a combination of the words "transistor" and "computer."

Each transputer is equipped with hardware "links" capable of high-speed bidirectional serial communication. There are four such links on each transputer.

These links allow any single transputer to be part of a multiprocessor network.

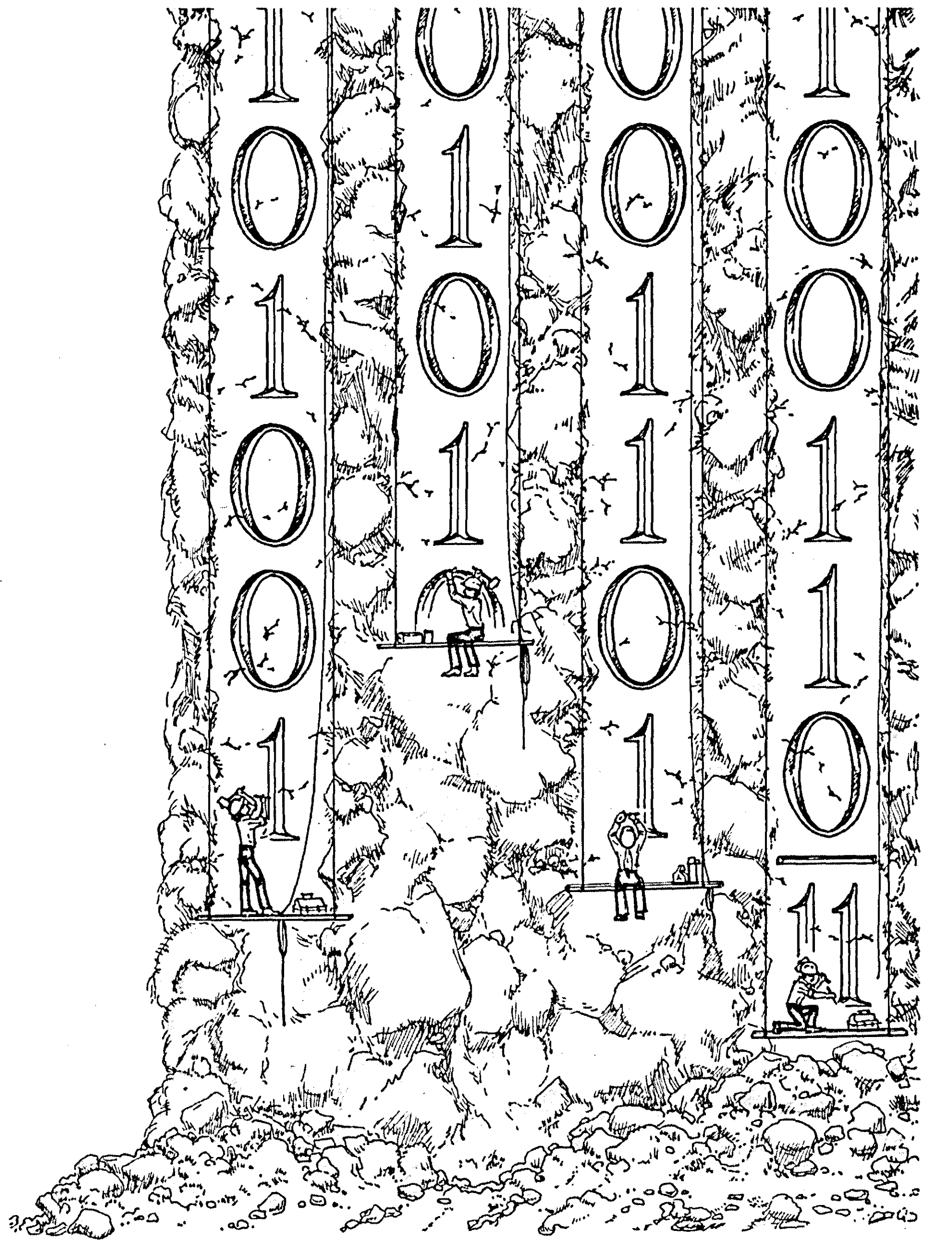
The transputer has other on-chip facilities, including a hardware timer and a task scheduler. With the task scheduler, the transputer is capable of multitasking with virtually no software support.

The INMOS transputer family currently consists of three processors: a 16-bit T212 and two 32-bit chips, the T414, and the T800. The T800 has an on-board floating point co-processor. The T414 is currently available in 15 and 20 MHz versions. The T800 will be available in 20 and 30 MHz versions. INMOS also has a number of transputer support products, including a disk processor and a graphics controller.

Transputers have fast on-chip memory. This memory (2K on the T212 and T414, 4K on the T800) is mapped into the low end of a transputer's address space. The transputer accesses its internal memory without waitstates while it adds at least two processor cycles for external memory.

Addresses are one word wide, yielding a contiguous address space of four gigabytes for the 32-bit T414 and the T800. The address bus is multiplexed with the data bus.

Timing for external memory is configurable. After reset, the transputer reads externally-supplied configuration data from a PAL or a PROM. This data informs the transputer of the external



memory type (static or dynamic), the speed of the memory, and the number of refresh cycles required.

Links & On-Chip Timer

The transputer's link engines are key to its success. Each transputer has four serial links, each providing two high-speed serial channels: one input, one output.

These links operate concurrently with the transputer's CPU, using Direct Memory Access to get data to and from memory. Thus, communication overhead is very small, even when all four links are running at the same time.

The links currently operate at a maximum speed of 20 MBits/sec, yielding an effective unidirectional data rate of 800 kilobytes/sec for the T212 and T414, and 1.8 megabytes/sec for the T800.

The links utilize a handshake in which each byte sent must be acknowledged before the next byte is sent. The T800 has an improved handshake mechanism, thus its faster transfer rate.

Links are assigned locations in the transputer's internal memory. Each input and output channel has its own memory address.

A program communicates across a link by issuing an input or output instruction. These instructions include the memory start address of the data and the length of the data block.

The input or output instruction invokes the link engine. Once it's started, the CPU is free to do something else.

The link engines don't do any buffering, so communication only takes place when a sending process and a receiving process are ready to communicate (i.e., one side has issued an output instruction and the other side has issued an input instruction).

The transputer features an on-chip timer that can be set or read from a program. It also allows a process to be descheduled up to a certain time. The timer is also used as a timebase for the multitasking engine.

Sequential Transputing

Although the transputer was primarily developed to support parallel processing, we can also view it as a conventional microprocessor with added parallel features.

The transputer is largely a RISC (Reduced Instruction Set Computer) machine. It has limited addressing modes and a relatively small number of instructions, especially when contrasted with some other 32-bit microprocessors. But, in part, because of this, it features a very fast, microcoded processing unit that runs at 15, 20 or 30 MHz.

The transputer's working registers are organized as a register stack - three words deep. These registers are (from top to bottom) A, B and C.

We access the register stack through the A register. When we pop its value, its contents come off the stack, and those of the B and C registers move up. Conversely, when we push a value onto the stack, it's placed in the A register; the old contents of the A register move into the B register; and the C register will contain the value that was contained in the B register. The contents of the C register are lost.

Programmers used to conventional addressable registers may have difficulty adjusting to the transputer's register stack.

In addition to the register stack, the transputer has three other major registers: the instruction pointer, the operand register, and the workspace pointer register.

Instructions

Transputer instructions are one byte long. Since the transputer fetches one word at a time, 32-bit transputers are effectively equipped with a four-byte instruction cache. Instructions consist of two separate fields: a four-bit operation code and a four-bit operand.

As a RISC machine, the transputer is designed to allow many instructions to resolve to a single byte; it's obviously impossible, however, to code all operations

(continued next page)

(continued from page 7)

in a single byte. Therefore, some operations require more than one instruction.

Of the 16 basic operation codes, 13 are frequently used - like load, store, jump, and call. These instructions all require an operand, which is partially supplied by the four-bit operand field.

Since four bits per operand often aren't enough, the transputer has two special instructions, "pfix" and "nfix," that load its operand register. These instructions shift the operand register four bits to the left and place their operand in the four bits cleared by the shift (the lowest four bits or least significant nibble). With these instructions, it's possible to load the operand register with any value up to the maximum word length of the transputer.

This unconventional instruction set allows us to code many frequently-used operations in only one byte. Decoding is simple and quick. Expressing a large constant, however, requires eight bytes with this INMOS approach, whereas other 32-bit processors require less. But we encounter this so seldom in program code it doesn't pose a problem.

Workspaces

Each task or process running on a transputer has its own workspace. The transputer's workspace pointer register points to the base address of the workspace for the currently executing process. The workspace lies above the base address.

Memory locations within a small offset from this workspace are "local," and can be accessed with extreme speed using 'load local' ("ldl") and 'store local' ("stl") instructions. If a memory location is within 16 words of the workspace pointer, it can be accessed with a one-byte instruction.

When a process's workspace resides in the transputer's internal memory, workspace data can be accessed in one processor cycle (50 nanoseconds on a 20 MHz T414). In this case, the workspace acts much like a large set of registers on a conventional microprocessor. Later, we'll see how workspaces enhance multitasking on a transputer.

Parallel Transputing

As we've seen, the transputer can be used as a building block for parallel systems, with its links forming the communication channels between processing.

In itself, however, a transputer also has parallel features like a task scheduler, in hardware, which can schedule processes on its own. Task switching takes less than one microsecond.

Processes that are running on a transputer can have one of the following statuses:

- running
- waiting in a queue to be executed
- waiting for input
- waiting for output
- waiting for a certain timer value

In the first two cases, the process is active; in the latter three cases, it's inactive.

Its unconventional instruction set allows us to code many frequently-used operations in only one byte.

Process Switching

Processes run either in low or high priority. A high priority process can interrupt a low priority process at any time, and continues until it's finished or until it must wait for communication or for the timer. Low priority processes are descheduled after they've been running for a preset time.

A low priority process is allowed to finish its current instruction and its information saved in reserved memory before it must give way to one with high priority.

Depending on the instruction being performed at interruption, the task switch time from high to low priority can take from 19 to 58 processor cycles (1 to 3 microseconds on a 20 MHz T414).

As soon as a high priority process has to wait, an interrupted low priority process is rescheduled (if there are no other active high priority processes).

Low Priority Switches

The manner of scheduling differs for process switches between two (or more) low priority processes.

Descheduling only occurs when certain instructions are executed, and the programmer or compiler must take care that the register stack doesn't contain information that shouldn't be lost.

This places an additional burden on the programmer or compiler writer, but the transputer stores much less process information during a low priority to low priority switch.

If descheduling occurs due to timeslicing, the workspace pointer is put in a linked list of active-but-not-running processes. If descheduling occurs because a process becomes inactive, the workspace pointer is saved in a memory location associated with the event for which the process has to wait. As soon as the event has occurred, the process becomes active and is put in the linked list of active processes, waiting to be executed. All this is done automatically in microcode.

The linked lists are managed with two registers that contain pointers to the first and last processes in the list. The workspace of every descheduled process also contains a pointer to the next process in the list.

Internal Channels

When a transputer is running several processes, these processes will typically want to communicate. For this reason, the transputer offers internal channels. Internal channels are used in exactly the same way as links. The only difference is that any memory location will work in place of the reserved locations used for the links.

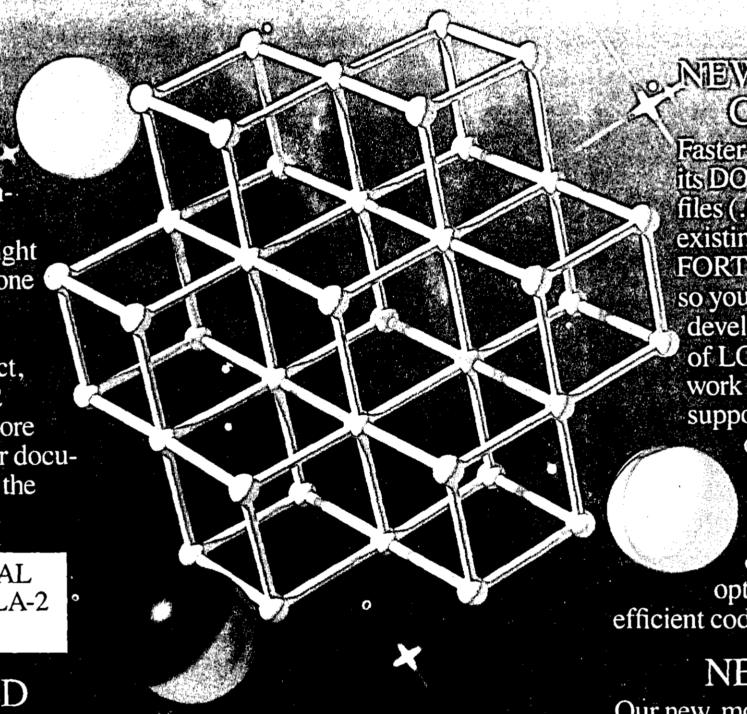
When an output instruction is performed on a specific channel, the memory location used will contain the workspace of the sending process. Another process that wants input from that channel performs an input instruction and will find the workspace pointer of the sending process. The workspace of that process now also contains information about the data to be sent and the communication takes place.

The big and obvious advantage of this internal channel approach is that programs can use internal channels in the same way as external channels. Programs can easily be adjusted to run on several transputers as well as on one.

■■■

SOFTWARE ENGINEERING COMES OF AGE.

ANNOUNCING LOGITECH MODULA-2 VERSION 3.0



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

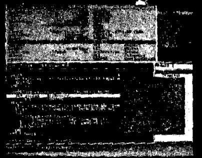
FREE TURBO PASCAL TO LOGITECH MODULA-2 TRANSLATOR

NEW, IMPROVED COMPILER

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

NEW, IMPROVED DEBUGGERS

Time gained with a fast compiler can be lost at debug time without the right debugging tools. With the powerful Logitech Modula-2 Debuggers you can debug your code *fast*, and dramatically improve your overall project throughput.



The Post Mortem Debugger analyzes the status of a program after it has terminated while the dynamic, Run Time Debugger monitors the execution of a program with user-defined breakpoints. With their new, mouse based, multiple-window user interface these powerful debugging tools are a pleasure to use.

NEW, INTELLIGENT LINKER

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

NEW EDITOR

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

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

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

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number ________ Expiration Date ________

Signature ________

Name ________

Address ________

City ________ State ________

Zip ________ Phone ________

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

To place an order call toll-free:
800-231-7717
In California:
800-552-8885

LOGITECH

LOGITECH, Inc.
6505 Kaiser Drive, Fremont, CA 94555
Tel: 415-795-8500

In Europe: LOGITECH, Switzerland
Tel: 41-21-87-9656 Telex 458 217 Tech Ch

In the United Kingdom: LOGITECH, U.K.
Tel: 44908-368071 Fax: 44908-71751

Turbo Pascal is a registered trademark of Borland International. VAX and VMS are registered trademarks of Digital Equipment Corp.

Between Transputers:
The Communication Manager

Your assignment, should you choose to accept it, is to study this carefully. Commit it to memory, understanding that you may be one of only a handful who understands its ramifications. This is total immersion into parallel processing.

Although my subtitle, "The Communication Manager," sounds like the start of an intriguing novel about some fellow in the communications game, what follows is really much more interesting. This is about controlling communications between networked transputers.

The purpose of a communication manager is to relieve a program (or programs) from dealing with dataflow. A communication manager can also emulate more than one hardware link between two transputers.

To give you a feeling for the mechanisms involved (e.g., the proper use of priority levels and other hardware features of the transputer), I'll first describe a simple, dedicated messagepasser. Then I'll discuss a more generic messagepasser and communication manager.

The Mandelbrot Demo

Calculating and displaying a Mandelbrot set is one way to demonstrate the meaning of "real" parallel computing. The calculation is processor intensive, requiring solving a complex equation many, many times.

For every pixel on a screen, we calculate the Mandelbrot equation with slightly different parameters so that every point on the screen is represented by a unique value. The number generated for each pixel on the screen is mapped to a color. Thus, we can represent a Mandelbrot set on a color monitor.

What concerns us here is not the Mandelbrot equation itself, but the fact that it requires a separate calculation for each pixel on the screen (and the fact that it makes sensational pictures).

In principle, if you have as many computers as there are pixels on your screen, the maximum time it takes to calculate the complete screen will be roughly equal to the time it takes to calculate a single pixel. (The duration of the calculation varies with the parameters passed to it.)

We decided to write a generic program in which the number of computers was set in a variable.

We wrote our Mandelbrot algorithm to calculate a number of pixels at a time. This process reads in the coordinates of the first pixel and outputs a block of data containing the number of iterations (the number which ultimately renders the color on the screen) for the next n pixels. All the transputers run this same algorithm as a single task or "process."

We have a program running on the PC which controls the generation of the Mandelbrot picture. It issues pixel coordinates, receives calculated blocks of data, and translates the data into colored pixels on the screen.

The Mandelbrot demo uses a "farm network topology." Two links are used on every transputer; one connects to the previous transputer (the one that is nearer to the host) and the other one to the next transputer (the one that is further from the host). See Figure 1.

The Mandelbrot Messagepasser

To connect (in a software sense) the calculating processes on all the transputers to the control process on the host, every transputer has, in addition to a calculation process, a dedicated messagepasser.

The messagepasser is the communication manager in this application. It's a dedicated messagepasser, as opposed to a general purpose communications manager, because it only knows how to deal with the kind of communication required by the Mandelbrot demo.

When the calculation process has finished calculating a block of pixels, it passes the address of its result (the data block) to the messagepasser. Once the calculating process sends this pointer, the messagepasser requests the next pixel coordinates from the host.

The messagepasser takes care of buffering and passing results back to the host while, at the same time, receiving new coordinates from the host (controller). The calculating process is therefore almost immediately able to start calculating the next block of pixels.

To balance the work load among a number of transputers, the controller (the PC in this case) sends new pixel coordinates into the network only after it has received back a block of data.

When the controller sends out the new pixel coordinates, they go to the first messagepasser in the line. The first messagepasser checks its own processor. If the processor isn't busy, then the messagepasser puts the coordinates into a special buffer for its own processor; otherwise, it passes the coordinates to the next transputer in the line.

The messagepasser connects its own transputer with the next and previous

We decided to write a generic program in which the number of computers was set in a variable.

By Hans Bieleman
Ingenieursbureau Bieware
Witbreuksweg 381-401
7522 ZA Enschede, The Netherlands

transputers in the (farm) network. It arbitrates the flow of messages so that a message can pass through even while its own process is also receiving and sending messages.

Messagepasser Processes

In order to make optimal use of the transputer, the messagepasser is divided into several parallel processes. A closer look at the transputer's features makes this "division of labor" obvious.

First, the hardware channels. Reading and writing by the hardware channels (also known as "links") into the memory of the transputer is performed via Direct Memory Access (DMA). Thus, there's little CPU overhead once communication begins.

A process which wants to communicate through a hardware link will always be descheduled. The transputer will reschedule the process only after the message has been sent or received. Thus, transputer processes sleep as long as they're waiting for communications.

When a process is descheduled, the transputer puts its ID at the end of a ready queue. It can take some time before the process has worked its way back to the start of the queue and is able to run again. Therefore, to use the hardware links efficiently, it's best to use the largest possible message. In this way, a process will be descheduled a minimum number of times. (Thus, in this case it makes sense to calculate and pass 100 or even 1,000 pixels worth of data at a time, rather than 1.)

Synchronization

The use of internal channels offers a sophisticated way to synchronize processes without semaphores or other fancy software tricks.

For instance, let's say two processes need to share data. One might be ready to send data to the other, but the other isn't ready to receive. The sending process will be descheduled until the receiving process is ready.

When the receiving process is ready, the sending process (in our example) will be automatically rescheduled. Meanwhile, the sending process hasn't been pestering its processor to check if the second (receiving) process is ready.

As you can see, the hardware scheduling features of the transputer channels (or links) really save time.

Figure 1 - A Farm Network Topology

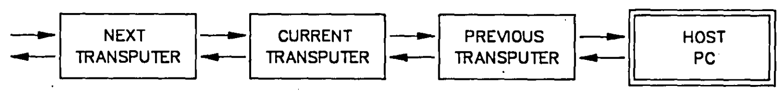
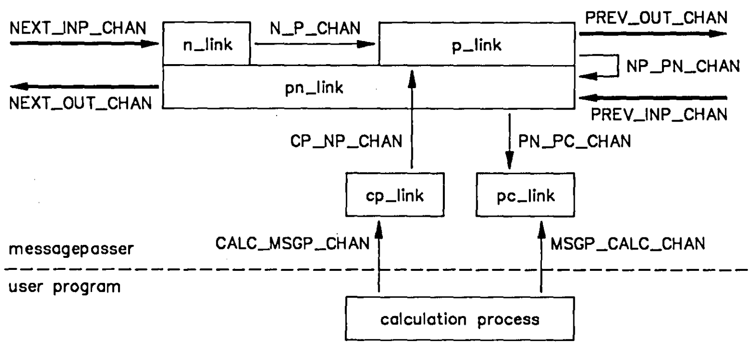


Figure 2 - Process Structure



Process-Structure

Let's have a look at Figure 2, the process-structure.

The boxes represent different processes, all of which run concurrently on one transputer. The arrows represent channels. A bold arrow represents an external (hardware) channel. I'll refer to processes with lower-case names and to channels with upper-case.

As Figure 2 illustrates, the messagepasser consists of five different processes, which are all started by a parent (boot) process (not shown in the illustration). The parent process kills itself after booting the child processes. (Obviously not a traditional American process.)

Three processes run in high-priority: n_link, p_link, and pn_link. They're high-priority processes because communication between the next- and the previous-transputer won't depend too much on the processes on the current transputer. Only high-priority processes on the current transputer can slow down communications. (The Mandelbrot calculation has no other high-priority processes.)

(continued next page)

(continued from page 11)

High-Priority Processes

Running at high-priority has a few major effects on the execution of the process.

First, a high-priority process will not be descheduled unless it has to wait for a channel to become ready or for the timer to match a certain count.

Once a channel becomes ready, the process is scheduled again. As long as there are processes on the high-priority ready queue, no low-priority processes will be executed.

To give low-priority processes a chance to execute, it's wise to keep the high-priority processes as short as possible. Therefore, it's sometimes necessary to divide a single process into a high-priority and a low-priority process. Communication and synchronization can be done with an internal channel. What this means is that a high-priority process will be almost immediately executed after it's been put on the ready queue.

The n_link process is a very simple piece of code. It merely reads in a block of data from the previous transputer and sticks it directly into memory.

Then a pointer to the block is passed on to the p_link process (rather than the contents of the block).

This pointer is passed on an internal channel. The data transfer between processes on the same transputer is thus very fast. n_link can put received data in one of two blocks. Once it's sent a pointer to the first block to p_link, n_link starts filling the second block.

p_link will start reading the N_P_CHAN only after it has passed the contents of the first block to the previous transputer. Because p_link hasn't yet read the N_P_CHAN, n_link can't send a pointer to the second block. So n_link is descheduled.

n_link will only be rescheduled after p_link has read the pointer to the second block. At this time, the first block is empty. Thus, the internal link helps synchronize the two processes.

Because we have a receiving process on NEXT_INP_CHAN and a transmitting process on PREV_OUT_CHAN, reception and transmission can take place at the same time (thanks to DMA).

The p_link process not only passes data blocks from N_P_CHAN to PREV_OUT_CHAN, but also transmits the calculated blocks on the current transputer to the previous transputer. After the block has been sent, p_link passes the pointer to that block to the pn_link process by means of the NP_PN_CHAN.

pn_link then reads in the first set of coordinates coming from the host and places it at the first location of the block. In this way, we are always sure that the block is empty before putting in new coordinates.

When the pn_link reads coordinates from the previous transputer, it first checks to see if there's a request for new coordinates from the current transputer. If so, the new coordinates are put in the waiting empty block. If not, the coordinates are passed on to the next transputer. Once the new coordinates for the current transputer have been received, pn_link sends the datablock's pointer to pc_link.

The messages are so small that hardly any time is required to send or receive them. pn_link can do this job sequentially and only runs after an input on the incoming channels. Because pn_link will wait for input most of the time, an output to the pn_link on the next transputer is serviced immediately.

The pc_link process acts only as a buffer to hold the next coordinates for the calculating process. cp_link buffers the calculated block. So, the calculating process can start calculating the next set of pixels and doesn't have to wait for the p_link to service it.

The calculation process can read them in from the pc_link. The calculation

Figure 3 - Seven Layer Network

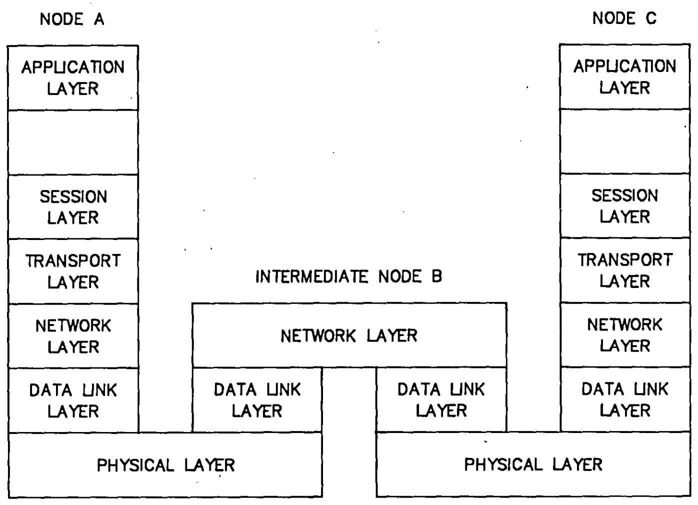
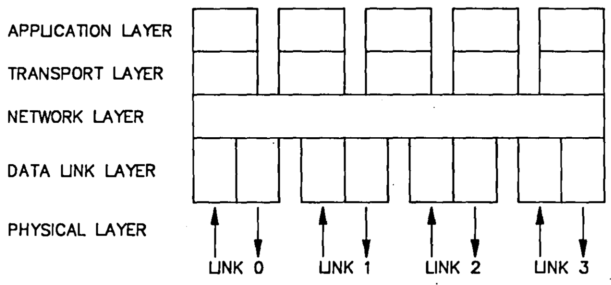


Figure 4 - Messagepasser On A Single Transputer



process is always fully loaded, so every transputer is never idle.

The use of links also has the advantage that the pc_link and cp_link processes are descheduled; they don't consume processor time to check now and then if a service is requested from them. If the calculating process wants a service, the processes are rescheduled.

Because we use internal links, and because both pc_link and cp_link will wait on the calculating process, the calculating process will not be descheduled to wait for a channel to become ready and it's able to keep on running.

The Mandelbrot messagepasser demonstrates how we can combine transputer features (e.g., internal channels, process priority levels, and characteristics of the hardware scheduler) into an efficient whole.

A Generic Messagepasser

A network of transputers is just like any other computer network. It cannot claim divine right or other forms of exemption. As with other computer networks, it's best to keep the sending, receiving, routing, error detection and correction, etc., out of user sight.

A network of transputers is just like any other computer network. It cannot claim divine right or other forms of exemption.

One of the goals of a generic messagepasser is to support multitasking. Multitasking for a system of transputers is a minor extension, but it's of great use to the user because he can run a second task.

For instance, in a control system, it might be handy to start a statistical task or a distributed database. In later stages, dynamic load balancing of the processes on the transputers should be possible. So let me give you an overview of a generic messagepasser.

The ISO-OSI 7-Layer Model

The ISO-OSI network model provides a good starting point for a general purpose messagepasser. If we look briefly at the ISO-OSI model for a network, we can see that it's divided into several layers (see Figure 3).

Not all layers of the ISO-OSI model need to be implemented for a transputer messagepasser since the model is general and flexible in nature. The physical layer is already implemented by the transputer hardware. The wires of the external channels form the Physical layer.

The messagepasser on a single transputer will roughly look like Figure 4.

The Data link layer consists of several parallel processes - two each for the eight hardware channels or four links of the transputer.

The function of the Data link layer is to receive message-packets, put them in a message-block, check them for parity, and pass a block pointer to the Network layer.

The Network layer takes care of the routing of the messages. This layer has the following tasks. It interprets the message-blocks coming from the Data link layer.

The Network layer looks at the type of message and the destination of the message. If the block is meant for another node, the network layer will look into a routing table to find the best link to output the block on. In case of obstruction of the best link, an alternative link may be chosen from the table.

Other things may also influence the Network layer's choice of a link. If the block is meant for this node, the pointer to the message-block is passed on to the Transport layer.

Message-blocks coming from the Transport layer are interpreted on their destination address. The source address is completed with the node ID (NID). The Network layer then routes the block to an appropriate link.

In case of receive or transmit errors from the Data link layer, the Network layer issues a check on the erroneous link/node.

The upper layers have identical processes running for every task, so we can keep the task administration simple.

The Transport layer takes data from the Session layer and splits it up into message-blocks. It also determines the message type. (The types of messages currently defined are in Figure 5.)

Figure 5 - Message Types

- general resource call
  direct resource call
  resource call acknowledge
- general path call
  direct path call
  path call acknowledge
- message request
  normal message
  confirmed message
  express message
- retransmit packet
  check link
  check node

The resource call types are used to find a resource in the network (i.e., disk systems, screen i/o, etc.). The path calls find and build a specific or general virtual channel to another process. The message types are used when a virtual channel between two processes is established and communication can take place. The last are some miscellaneous types used to check channels, etc.

The Transport layer splits up messages to be sent into message-blocks with a maximum size of 255 bytes. This message-size is a compromise between the performance advantage of long messages and the ability to send short, express messages through the network. Smaller messages are allowed, so there's little overhead in sending characters to a screen on another node (a host computer, for example).

Received message-blocks are assembled into a single message, and the pointer to this messages is passed on to the Session layer.

The higher layers (Session layer through Application layer) implement the interface to a kernel. The type, length, and destination of messages are determined here.

Thanks From Transputerland

A good discussion of a general messagepasser would fill a book. I hope my brief discussion has given you at least a little insight into the purpose and utility of a messagepasser.

I wish to thank Prof. Andy Bakkers of the University of Twente, The Netherlands, for guiding me into (parallel) transputerland.

■■■

Developing A Parallel C Compiler

Powerful New Extensions For A Familiar Language

Parallel processing isn't much good if you have to rewrite everything you have from scratch. Here, Klaas and Rob show us how an old favorite can be taught new tricks, and how we can learn them, too.

The transputer has made it possible to build large parallel processing systems, which increase linearly in speed with the addition of processors. However, to use these systems, we need a good programming language.

This programming language needs to be powerful and should support special transputer features like channels, links, and the ability to initiate processes.

INMOS (British marketers of the transputer) provides a high-level language called Occam for parallel programming. Occam is quite powerful - with its parallel constructs: par and alt. And Occam knows the channel data type, which gives us some control over the placement of processes on different transputers.

But it has several disadvantages. It's a new language, with a small base of source code and programming know-how. It's a static language: every variable must be allocated at compile time, and there's no recursion. (Although some folks claim that any problem can be solved without dynamic allocation of variables or recursion, it's often much easier to use recursion.) Also, it has a smaller set of statements and operators than other, more popular languages.

C, on the other hand, is well-known and well-suited for systems programming. Plus, there are vast C libraries.

Unlike Occam, C supports recursion and dynamic allocation of variables. C has a powerful expression syntax and range of statements. For our purposes, the only thing C lacks is a special set of statements and operators for the transputer.

To obtain both the advantages of C, and the advantages of the transputer, the C language had to be extended. And, these extensions had to fit into the C environment.

This article describes these extensions.

The Channel Datatype

The channel datatype is new, providing synchronized communication between processes. A programmer uses channel variables to transmit data between processes. The processes may be running on the same transputer or on separate transputers.

From the programmer's point of view, communication is the same regardless of whether or not the two communicating processes are on the same chip. The transputer's specialized hardware handles both types of message-sending with equal facility.

Using the keyword "channel," one can declare a variable of the channel type:

```
channel chan1, chan2, chan3;
```

Declared this way, each channel variable will occupy one word in memory. Sending a message with the aid of a channel variable is most simply performed with an assignment operator. For example, the fragment:

```
int a;
a=chan1;
```

will read an integer value from the channel chan1, and store it in the variable a.

The use of a channel is not restricted to simple expressions. Channels can also be used in expressions like:

```
chan2 = chan1 + a * (char) chan3
```

This statement inputs a character value (1 byte) from the channel "chan3,"

Figure 1 - Two Independent Processes

```
channel PtoQ;

P()
{     . . .
      PtoQ = 1;
      . . .
}
Q()
{     . . .
      printf("Received from P: %d\n", (int)PtoQ);
      . . .
}
```

Figure 2 - Par Construct

```
par {
      statement1;          /* subprocess 1 */
      statement2;          /* subprocess 2 */
          .
          .
          .
      statementn;          /* subprocess n */
}
```

By Rob Kurver and Klaas Wijbrans
UNICOM
Oldambt 69
Utrecht, The Netherlands

Irisstraat 10
7641 VT Wierden
Holland

and multiplies it by the integer "a." Then it takes this result, adds the integer value from "chan1," and outputs the final result to "chan2."

In this example, you can see that channel variables behave much like conventional variables. When we read channel variables, their values are taken from another process instead of from a location in memory. Writing to a channel variable sends the value to another process.

The number of bytes transferred via a channel depends on the context. Expressions and sub-expressions in a C statement are typically governed by a certain data type. A channel variable in an expression or sub-expression assumes this governing data type:

```
channel dogdays;
    struct CATSMEOW {
    int array[40];
    } bigtime;
    dogdays = bigtime;
```

In this example, the governing data type is the structure "CATSMEOW." The number of bytes sent by the channel is equal to the size of this data type (160 bytes or 40 * sizeof(int)).

In this example:

```
    channel tohostpc;
    double result;
    long count;
    tohostpc = result / count;
```

the compiler converts "count" to a double, performs the division, and outputs the resulting value as a double to the channel "tohostpc."

Channel variables can also be used in a boolean expression:

```
if (chan1) .....
```

When declaring channels, there's no restriction on the complexity of the declaration. This means that arrays of channels, and structures containing channels, all are possible. So it's legal to say:

```
    struct P {
        channel a;
        channel b[50];
        };
```

From the programmer's point of view, communication is the same regardless of whether or not the two communicating processes are on the same chip.

Channel Behavior

Since a channel is a means of communication between two processes, whenever a process outputs a value via a channel, another process must input that value.

Suppose we have a program (like the one in Figure 1) with two independent parallel processes, P and Q.

For channel communications, you must have one inputting process and one outputting process, and both have to communicate an equal number of bytes. If these conditions are not met, the behavior is undefined.

The Link Interfaces

The transputer has four high-speed bidirectional serial links; each link has an input and an output channel. It's interesting to note how the link interfaces are integrated into the transputer instruction set. A link behaves like any other channel, with only a few differences:

- A link communicates between processes on different transputers.
- The links are DMA based, and thus consume no processor time.
- The links are placed at fixed addresses.

It's not difficult to access the links from within a C program using pointers. On conventional machines, we commonly access a memory mapped I/O register by creating a pointer to the type of that register, and initializing this pointer with the address of that I/O register. The same technique can be used to access the links on the transputer:

```
channel *Link0_Out = LINK0OUT;
```

This declaration defines a pointer to the channel Link0_Out, and initializes this link with the value LINK0OUT. This value stands for the address of this link, 0x80000000 on a T414 transputer. To send information across this link, the following expression suffices:

```
*Link0_Out = value
```

This expression sends the value "value" over the link.

The Par Construct

The par construct allows the C programmer to instantiate several parallel processes within the body of a program. There are two variants of the par construct: replicated and non-replicated.

A non-replicated par construct consists of the keyword "par," followed by a complex statement. See Figure 2.

Statements nested within the complex

(continued next page)

(continued from page 15)

statement are executed in parallel. A compiled par statement will continue to execute until all of these subprocesses have terminated.

On the transputer, the par statement translates into machine instructions quite easily. Each subprocess in a par statement is instantiated with the aid of only two transputer instructions - startp and endp.

The startp instruction, not surprisingly, starts each of the subprocesses. At the end of the par statement, each subprocess finishes by executing an endp instruction. After the last subprocess has ended, the calling process resumes execution. See Figure 3.

The replicated par construct behaves somewhat differently. With it, one can start any number of processes, using a loop expression. The replicated "par" resembles C's "for." See Figure 4.

Statements within the braces are started as independent processes for each iteration of the loop. Each subprocess receives its own copies of the variables in the loop expression.

The replicated par statement in Figure 5 computes the square root of each element in an array.

The replicated par construct's loop expression isn't restricted to simple expressions. It's possible, for example, to start a process for each element of a dynamic list. See Figure 6.

The Alt Construct

Alt is another important language extension. Like the par, it's closely tied to hardware features of the transputer. The alt construct is used to wait for certain events, or it can be used instead of a C switch statement or nested if statements. The syntax of the alt mirrors that of C's switch statement:

```
alt {
    guard guardexpression:
        code;
    guard guardexpression:
        code;
    default:
        code;
    }
```

In the alt construct two new keywords appear: 'alt' and 'guard.' The keyword 'alt' indicates the start of an alt statement; 'guard' compares to case labels in a switch statement.

However, there are some principal differences. Guards are evaluated at runtime whereas case statements are evaluated at compile time. Therefore, guard expressions aren't restricted to constant expressions, but can use complex expressions that must be evaluated at runtime.

The guards in an alt statement are evaluated in the order in which they're placed within the alt braces, and the result of a guard expression can be active or inactive. The code of the first guard that's active will be executed. If none of the guards are active, the process executing the alt will be descheduled until one of the channel or timer guards becomes active.

Altogether there are five types of guards:

- boolean guard
- channel guard without boolean
- channel guard with boolean
- timer guard without boolean
- timer guard with boolean

The boolean guard is the simplest. A boolean guard consists of the keyword guard, followed by an expression with a boolean result. If the result of the boolean expression is true, the guard is active; otherwise, the guard is inactive. The boolean guard can be used instead of nested if statements.

Figure 3 - Begin & End Subprocesses

```
The fragment:

                int a,b;
                par {
                     a=3;
                     b=4;
                }

compiles to:

     ldc    .6-.7      ; par {
     ldlp   -$6
     startp
.7   j      .5
.6   ldc    $3         ; a=3;
     stl    $7
     ldlp   $5
     endp
.5   ldc    .9-.10
     ldlp   -$c
     startp
.10  j      .8
.9   ldc    $4         ; b=4;
     stl    $e
     ldlp   $b
     endp
.8   ldc    .12-.11    ; }
     ldpi
.11  stl    -$1
     ldc    $3
     stl    $0
     ldlp   -$1
     endp
.12
```

The example:

```
if (a) b=1;
else if (a23) b=2;
else if (a5) b=3;
else b=4;
```

is equivalent to:

```
alt {
    guard a:   b=1; break;
    guard a23:  b=2; break;
    guard a5:  b=3; break;
    default:    b=4; break;
    }
```

Channel guards are the second type of guard, consisting of an expression with a channel address and an optional boolean expression. The channel guard is active if the channel it points to is active (ready for input or output) and the optional boolean expression is true. The channel guard is very useful for multiplexing multiple internal channels to one external channel:

```
alt {
    guard &intchannel[0]:
        intchannel[0] = *link0_in;
        break;
        guard &intchannel[1]:
        intchannel[1] = *link0_in;
        break;
    }
```

In this case the process sleeps until one of the guards is ready.

Timer is the third kind of guard. Like the channel guard, it may be used with or without a boolean expression. For example, we can use a timer guard to time out a channel by using the timer guard in an alt with some channel guards. The code in Figure 7 times out a channel for 1,000 internal cycles.

There's also a replicated alt construct, whose syntax mirrors the syntax of the replicated par construct:

```
alt (expression; looping_condition; expression)
    {
    guard guardexpr1:
        code;
    guard guardexpr2:
        code;
        ....
    }
```

The guards within the braces are evaluated for each iteration of the loop until one of the guards is ready. Like the par-construct, the replicated alt-construct may contain complex looping conditions.

This construct is useful if a dynamic list of channels has to be watched for activity. This situation can occur in a messagepasser or an operating system. See the code in Figure 8.

Using Parallellism

To use these parallel extensions beneficially, you should divide a program into parallel subprocesses.

There are several ways to speed up a program by rendering it in parallel. The next example illustrates how parallelism can enhance a program on a single transputer. (Multiple transputer parallelism is a subject for another article or a book!)

Parallelism can reduce IO overhead. For example, in most programs, IO is performed sequentially; that is, the next sequence is used often:

```
while (!ready) {
    Read_block_from_file();
    Process_block();
    Write_block_to_file();
    }
```

In this case, buffering in itself won't speed execution very much. However, on a transputer where the I/O occurs via the links, we can decrease execution times considerably by separating the read, process, and write sequences into processes:

```
while (!ready) {
  par {
      Read_Next_Block_from_file();
      Process_current_block();
      Write_Last_Block_to_file();
      }
  }
```

In this way, the transputer will simultaneously use the input link to fill the read buffer, the output link to empty the write buffer, and the processor itself to process the current block.

Toward A C Standard For Parallel

The C compiler with the parallel extensions we've described is an effective tool for programming a transputer-based parallel system. It provides the programmer with a familiar syntax and the power he needs to exploit concurrency.

It's my hope that these extensions will serve as the basis for a C standard for parallel programming constructs, and that other implementors will adopt or expand upon this work. ■■■

Figure 4 - Replicated Par Resembles "For" Statement

```
par (expression; looping_condition; expression)
{
     statement1;          /* subprocess 1 */
     statement2;          /* subprocess 2 */
         .
         .
         .
     statementn;          /* subprocess n */
}
```

Figure 5 - Square Root

```
char *malloc();
double *array;
array = (double*) malloc(K*sizeof(double));
     . . . .
par(i = 0; i < K ; i++)
{
     array[i] = sqrt(array[i]);
}
```

Figure 6 - Dynamic List

```
struct P {
     struct P *next;
     } p;
  . . . .
par (; p ; p = p->next)
{
     evaluate_element(p);
}
```

Figure 7 - Timing Out A Channel

```
alt {
     guard &intchannel:  /* channel to wait for */
          outchannel = intchannel;
          break;
     guard timer=timer+1000:  /* this guard if no message in
                                 1000 clock cycles. */
          printf("Error: no response from channel\n");
          exit(0);
          break;
}
```

Figure 8 - Checking For Active Channels

```
struct P {
     struct P next;
     channel in;
} *p;
. . . .
alt (; p ; p = p->next) {
     guard &p->in:
          out = p->in;
}
```

If You Don't Have WindowDOS 2.0, You're Wasting Time!!

" When Baba Ram Dass said "Be here now, remember," designers of hard disk utilities should have paid heed. A powerful manager like XTREE can track files and subdirectories and execute DOS commands, but it isn't memory resident. Handy pop-up DOS commanders like PopDOS may be here now, but they lack the power of a full-fledged disk manager. After much meditation, the developers of WindowDOS 2.0 have come up with the best answer yet to the guru's paradox.
Until now, the closest thing to a real RAM-resident disk manager was version 1.0 of WindowDOS. If offered a full-screen pop-up menu and could rename, copy, and delete files. But it couldn't move files, format disks, or rename subdirectories—which XTREE can. Now version 2.0 is here, and its a winner. Its RAM resident (using less than 50K) but offers all the power of a nonresident disk manager."
—Patrick Marshall, WindowDOS 2.0 Product Review, PC World, May, 1987

Once you've experienced the convenience of instant access to DOS commands, you'll never be satisfied with returning to DOS to list files, format disks, or copy, rename, or erase files. Nor will you be happy with a DOS shell, because shell programs are just as inaccessible as DOS when you are using an application program. ***Only one program combines memory-residency with the power of a full-featured disk manager:*** **WindowDOS Version 2.0.**

Features Not Found In DOS
- ♦ Sort directories in 8 ways--or not at all
- ♦ Copy, erase, and move groups of files
- ♦ Find any file in seconds
- ♦ Display default directory of any drive with a single keystroke
- ♦ Display graphic tree
- ♦ Global copy & erase commands
- ♦ Copy function prompts you to insert another disk when necessary
- ♦ Display hidden files and subdirectories
- ♦ Display file contents in various formats and page forward/backward
- ♦ Display Wordstar files in readable format
- ♦ Unique RAM Environment function shows name, size, location, and interrupts of every program in memory
- ♦ Rename subdirectories for instant reorganization
- ♦ Hide and unhide subdirectories
- ♦ See and change file attributes
- ♦ Send control codes to printer
- ♦ Switch default printer
- ♦ Password "lock" your system
- ♦ Set AT Real-Time Clock
- ♦ 5-minute screen-blanking function
- ♦ Input response macros

Enhances These Functions
- ♦ Format disks (faster than DOS)
- ♦ Make and erase subdirectories
- ♦ Copy, rename, and erase files
- ♦ Copy files to printer or COM ports
- ♦ Display disk free space and other media information
- ♦ Check and set the time and date

Benefits
- ♦ ***Saves Time***—No waiting to exit or reload programs. Instant access to DOS functions whatever your current task. Easily saves 10 or more minutes a day.
- ♦ ***Comprehensive***—Broad range of commands, including many not supplied by DOS. Satisfies the needs of both new and advanced users.
- ♦ ***Simplifies DOS***—No need to remember exact DOS commands. Intuitive interface and "point and shoot" design saves keystrokes and prevents mistakes. Group file "tagging" avoids the drudgery of repetitive commands.
- ♦ ***Security***—Capability to hide/unhide subdirectories, password "lock" a computer, and check for unwanted programs in RAM helps secure data and prevent unauthorized access.

WindowDOS 2.0 Addresses "RAM Cram" Like No Other Program!!!

1. Designed specifically to be loaded first, unlike most memory-resident programs that insist on being loaded last.
2. Uses a hot key combination that does not have an associated ASCII value—prevents key conflicts with other programs.
3. Unique RAM Environment function lets you monitor the locations, memory costs, and interactions of all programs in memory, including the currently running program. Great for power users/developers.

Other Information
- ♦ Not copy protected
- ♦ Uses only 51K of memory
- ♦ Supports EGA & Hercules
- ♦ Runs memory-resident or as a stand alone program
- ♦ Uninstall command
- ♦ PC/XT/AT/100% Compatibles
- ♦ ***Order Today--Only $49.95***

WindowDOS Associates • Box 300488-C • Arlington, Tx 76010 • 817-467-4103

HALTED

SERVING NORTHERN CALIFORNIA SINCE 1963!

SPECIALTIES

HARD DRIVES + CONTROLLERS

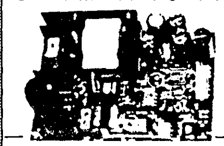
XEBEC 'ACCESS' 10 MB EXTERNAL HARD DISK— • IEEE-488 (GPIB/HPIB) Interface. • Attractive Enclosure 12" x 6" H x 10" W. • 10 Meg Tandon Drive.

$ 298.00

XEBEC IEEE-488 HARD DISK CONTROLLER CARD .... $ 99.00

TANDON 10 MEG HARD DRIVE .. $ 99.00
With Controller .... $ 169.00

SWITCHING POWER SUPPLY— IDEAL FOR DISK DRIVES— 50 Watt + 5V 4 AMP +12V 3 AMP •4" x 6" x 1-3/4"



$ 29.95

SHUGART SA801R 8" FLOPPY DRIVE— 600K SSDD • Still Sealed in Factory Boxes ....

$ 149.00

CORVUS Mirror Server— VCR Hard Disk Backup for Corvus Omninet. *NEW!!* **$ 25.00**

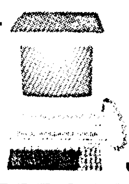
CORVUS Utility Server, Printer Interface For Omninet- Includes Manual + Software. *NEW!!* **$ 35.00**

4 PIN MODULAR WALL JACK with Strain Relief. 10 for **$4.** Case of 325 for **$ 99.00**

CORVUS HARD DRIVES + SOFTWARE AVAILABLE NOW!! PLEASE CALL!!

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

CORVUS CONCEPT WORKSTATION • 68000 Based System. • 512K Memory. • Connects to Omninet • 132 Column Monochrome Monitor (Operates in either Vertical or Horizontal Format • Used Working Condition.



$ 295.00

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

XEBEC 1410 SASI HARD DISK CONTROLLER. $ 99.00

HALF HEIGHT EXTERNAL DRIVE ENCLOSURE— Attractive Low Profile Chassis 19x15x2-3/4" • Fits snug under PC • BEZEL for 5-1/4" & 3-1/2" Dr. • 60 Watt P.S. w/ DC Fan & Cables. **$ 89.00** Complete

QBUS SASI HOST ADAPTER.
•Made by Xebec •Brand New!!
$35.00

YOUR COMPLETE ELECTRONIC SUPPLY STORE!

3-1/2" FLOPPY DRIVE— Toshiba FDD4408K13 • 720 or 360K Bytes • Includes: 5-1/2" Brackets, XT Black + AT Biege Front Panels • *Ideal* for usage with DOS 3.2 or our XT High Density Controller.

$ 139.95

• ***NEW!! TI 787 Terminal —***
• **120 CPS Thermal Printer**
110 to 9600 BAUD MODEM: Bell 212, 103 + Racal 3400 Compatible.
• ***Acoustic/Direct Connect***
• ***RS-232 Interface,*** Includes Manuals, Cables & Paper **$ 249.00**

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

$ 29.95

3 CONVENIENT LOCATIONS:

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

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

827 EVELYN AVE., SUNNYVALE, CA 94086

WE SHIP C.O.D.

MAIL ORDERS

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

Call NOW! (408)732-1573

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

By David Thompson

Laser Printers, Typesetters, And Page Definition Languages

Getting It Off The Screen And Onto The Paper

We're right on the edge of a new era. An era where quality type and graphics will be as cheap and available as dot matrix output used to be. That may not seem very important but it is, for even today a very, very few have a lock on the best of this technology.

For years, the standard typesetting companies, Compugraphic, Linotype, VariTyper, and Mergenthaler, had the publishing industry by the serifs. They produced huge electromechanical marvels (designed by Rube Goldberg) for which they were the sole sources of parts, service, accessories, updates, fonts, you name it.

These monstrosities had motors, strobe lights, mirrors, lenses, light-tight paper holders (okay, somewhat light-tight paper holders) transparent fonts... Their weight was determined by the structural limitations of a concrete slab.

Three years ago, a local printer purchased one of these impressive machines. He got the model with all the latest high technology.

Two weeks after it arrived, it died. He called to see if I'd take a look before he called in the $150 per hour technician from Portland.

When I popped the lid, I found a 6800 (not a 68000) talking to real TTL ICs (not low-power Shotky), a bus cage that would embarrass an S-100 owner, and a power supply straight out of *Welders' Week Magazine*.

After I'd found the problem, he mentioned that the monster didn't have an RS-232 port. Together we called around the country to find one. I've forgotten what he finally paid for that single (used) RS-232 port, I think it was $500. (Less than half-price.)

For years the small typesetting houses have grumbled about the high costs of equipment and maintenance. But the prices protected them, too. Now, as their customers purchase computers and laser printers, their business is declining. Oh, a few are busy, with their Linotronic typesetters and PostScript RIPs, but it's temporary and expensive. I've heard $80,000 for the typesetter and $10,000 a year for service. When 600 dpi and 1000 dpi lasers show up on desktops, more and more customers will disappear. Forever.

This transition from centralized low-tech typesetting to laser printers is marking the end of the dark ages. No longer will quality type be held hostage by a few large corporations.

How Laser Printers Work

Laser printers (and laser typesetters) scan dots across a piece of paper, a line at a time. Laser printers (300 - 600 dpi) deposit bits of carbon onto standard typewriter paper. Laser typesetters (600 - 2500 dpi) deposit bits of light onto photographic paper.

In each case, the characters on the paper are made up of dots just like the characters on your screen. Your hi-resolution monochrome screen has a resolution of about 100 dots per inch, so each one-inch by one-inch square on the screen contains 10,000 dots (called pixels).

Your common, house-variety laser printer has a resolution of 300 dots per inch. That means that each square inch that it prints contains 90,000 dots.

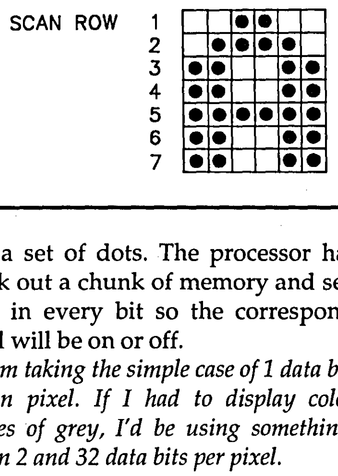
Now, let's see. The screen has 30 to 40 one-inch-square blocks, the laser printer has nearly 100. If the screen's controller or the laser printer's processor has to calculate whether each pixel is supposed to be on or off, things can get pretty slow.

For years, video designers have used a shortcut. Your video card has a character ROM. The controller says it wants an "A" and that it's scanning the fifth row of a line of characters. The character "A" is a 41 hex. So 41 hex is combined with the scan row (5 in this case) and sent to the ROM as an address (415 hex). The 8-bit value at 415 hex determines which pixels will be on and off as the beam scans the A's position (at least the A's fifth row). See Figure 1.

As long as you always want the same size and shape "A" (or "B," or "C"...), you can just grab it out of the ROM. No muss, no fuss. However, if you want a narrower, higher, or fancier "A," or if you want to draw pretty pictures - everything changes.

With graphics, the screen becomes just a set of dots. The processor has to block out a chunk of memory and set a 1 or 0 in every bit so the corresponding pixel will be on or off.

Figure 1 - Scanning an A

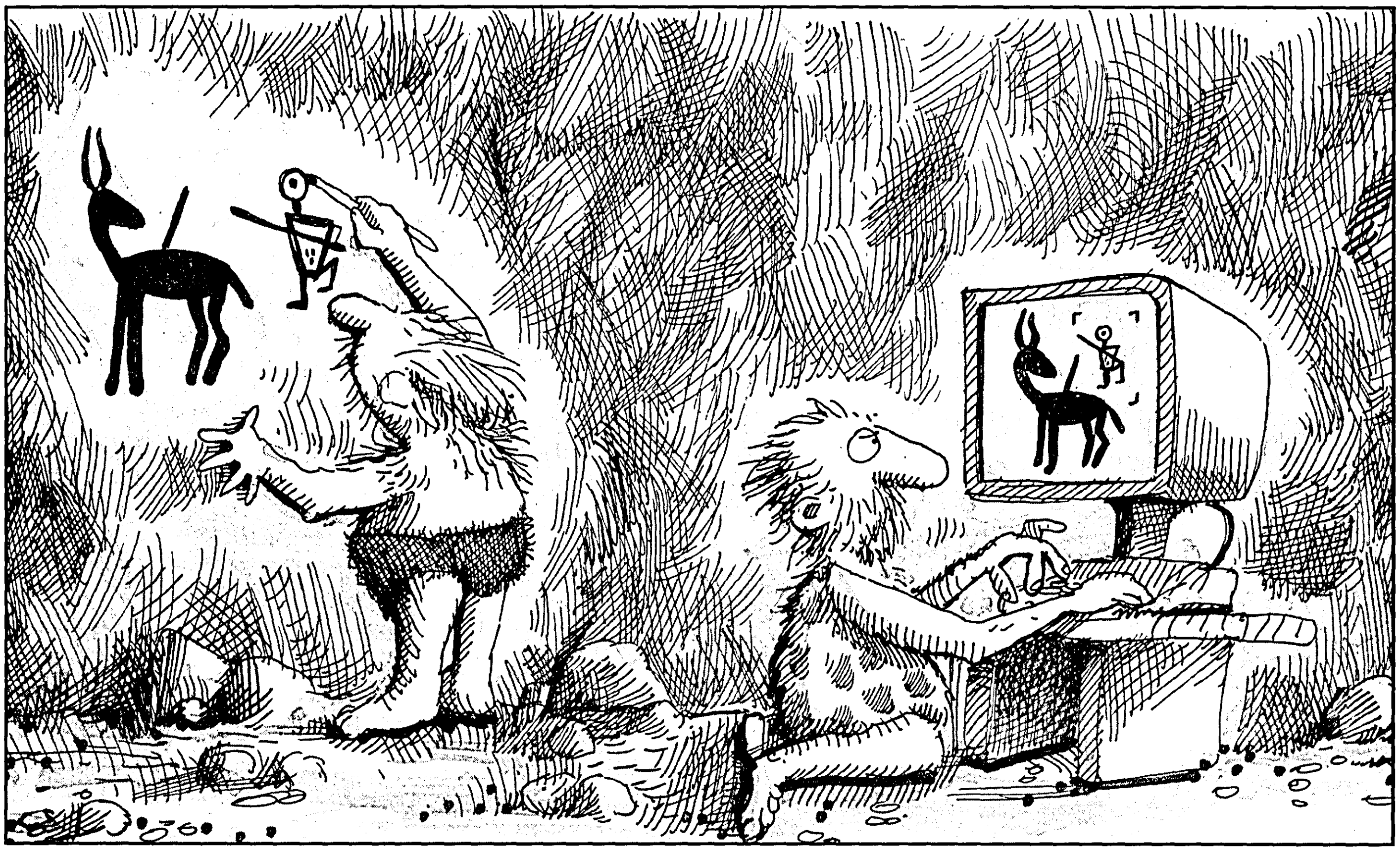


I'm taking the simple case of 1 data bit per screen pixel. If I had to display color or shades of grey, I'd be using something between 2 and 32 data bits per pixel.

Laser Printers

The very same thing happens in a laser printer or typesetter. Most laser printers come with built-in ROM-based fonts. These fonts, like the screen fonts, are fixed in height, width, shape, weight, everything. Some printers come with lots, some with just a few. If your printer comes with what you want, great.

These fonts are very easy for the laser printer's processor to use. The processor specifies the character and scan line (again, a unique location in the ROM). The contents at that ROM location deter-



This transition from centralized low-tech typesetting to laser printers is marking the end of the dark ages.

mine which spots on the paper become black and which white. It's that easy. The processor deals with a few addresses rather than a whole bunch of dots.

There's another variation on this theme. The computer can download one or more of these pixel (dot) fonts into the printer's memory and then the printer's processor simply addresses a character's location in RAM rather than ROM. Again, the processor doesn't have to create the character, it simply addresses it.

Pixel Font Limitations

This addressing system is easy, but it has some limitations:

- You must have in ROM or RAM a pixel map of every character. This means that larger characters (like headlines) really eat up memory (because each character is made up of lots of dots). Thus, a printer which might have room in RAM for two or three 9-point fonts might choke on a single 32-pointer. (This is true despite the fact that pixel-mapped type designers often reduce the resolution of larger faces to as little as 75 dots per inch, giving their headlines that ragged appearance.)
- You must have a copy of the font for each size of each face. (That's assuming that the size and face are available.) With ROM-based fonts costing up to $100 each ($400 for a ROM pack containing 4 fonts), you'll think twice about ordering 8-point Helvetica when you already have 10-point Helvetica. At $100 per font, a reasonable font library can cost many times the price of the printer.

A short definition here: A font is the alphabet, plus numbers and common symbols, all the same size and style. The old-time typesetters had font drawers. Each drawer held one size of one face. So there'd be a font drawer full of 9-point Helvetica and the next one down might be 12-point Helvetica. If he were setting 9-point Helvetica, he'd pull out that drawer and start picking out characters, one at a time. It's easier now.

Size, by the way, is the height of the characters and is measured in points. A point is 1/72 of an inch. The 9-point characters (like you're reading now) are about 1/8 of an inch high.

Now, this paragraph is set in a font called '9-point Palatino Italic.' This font is just one size of a face called 'Palatino Italic.' And, Palatino Italic is just one of a family of faces. The family is made up of Palatino, Palatino Italic, Palatino Bold, and Palatino Bold Italic.

(continued next page)

(continued from page 21)

If you purchase a type face, then you can make it any size you wish. If you purchase a type font, then you're stuck with whatever size it is.

Vector Fonts

Now we come to the processor (and memory) intensive portion of laser printing. It's possible to define a character's outline using vectors. (Go X degrees for Y units, circle left, alamand right, curtsy to your partner...) The computer sends the printer the vector definition of the face (Palatino, Helvetica, Bookman...) then it sends the size (usually something between 5 points and 30 points), and finally, one or more text characters.

The printer's processor then takes a character, grabs its vector definition, scales it to size, and then puts the character into image memory. Once all the characters have been calculated and written into image memory, the image is printed onto paper.

Many laser printers are rated at 8 pages per minute, but that's a bit misleading. It can take up to 45 minutes to complete all the vector calculations and print the first copy of the page. Additional copies of that page come off at 8 pages per minute.

More Than Characters

Unfortunately, pixel and vector character tables, character sizes, and characters aren't enough. The printer also needs to know where to put each character (including vertical and horizontal space between characters, how to underline or overline...), how to draw a line or box, how to fill a space with grey or black, how to print a graphics image (size it, scale it, squeeze it, stretch it, and rotate it).

Some of the early interface designers simplified communications between the computer and the printer by sending only pixels to the printer.

Each 8-inch line requires 2400 pixels (bits) and there are about 3000 lines in a page. At 9600 baud (after taking out start and stop bits), you'd get about three lines per second. So it takes about 1,000 seconds (16 minutes) just to transfer the image. And that's after the computer has created it. Of course, the computer also has to do lots of calculations, and it needs a pretty good understanding of the printer. Change the printer and you change the calculations.

Page Description Languages

But what if you gave the printer a very smart and very fast processor and a very powerful, standard, page description language (PDL)?

A PDL is like any other high-level language. A high-level language is a shorthand way of writing lots of assembly language. A PDL is a shorthand way of telling a printer where you want characters, lines, images, and anything else that a printer can construct out of dots.

And, if it's talking PDL, the computer shouldn't care what kind of printer is on the other end of the pipe (its resolution, paper type, whether it's laser, photo cell, ink jet, or pinhead). It's up to the printer to reproduce, as best it can, what the computer specified.

The standard PDL is PostScript. Hewlett Packard has adopted it, so has IBM, and it's becoming available on more and more printers.

PostScript printers are the most expensive. They need a PostScript interpreter, a decent processor (usually the 68000), and lots of memory (about 1.5 meg, minimum). Also, Adobe, originator of PostScript, gets a royalty out of every PostScript printer sold (I've heard that the PostScript license adds about $2,000 to the price of each unit).

Printer Engines

A printer's engine is the combination of laser, drum, toner cartridge, paper transport, etc.

1. Standard Cannon Engine: This is the original engine and appears to be based on Cannon's inexpensive (under $1,000) copier. Print quality is quite good at the boundaries between black and white areas, but blacks aren't very black and large areas of black are mottled.

The drum is part of the toner cartridge assembly, so replacing the toner (every 3,000 copies) is relatively expensive ($120 to $200). Users report that print quality is worst at the beginning and at the end of a cartridge's life, so many people use new and old units for proofing and save their mid-life cartridges for the final output. Engine is rated for about 100,000 copies.

Hewlett Packard laser printers, Apple laser printers, and many others (especially the least expensive) use the Cannon engine.

2. Ricoh Engine: This engine is a newer offering, but it's already available in a number of the latest, fanciest printers. Its drum is separate from the toner so toner replacement is cheaper. Also, its blacks are much blacker and more even (much easier for print shops to reproduce well), and its engine is rated for 300,000 copies. I calculate that over its lifetime, a Ricoh-based printer will produce copies for three cents each. A Cannon-based printer will run about ten cents each. That's counting the costs of supplies (toner, paper, drums...) as well as the cost of the printer.

3. Other Engines: There are numerous other companies coming out with newer, sharper (eventually, higher resolution) engines. NEC, for instance, has designed its own engine based on a 300 dpi LED device. Instead of directing a laser beam with mirrors, NEC stuck a row of LEDs onto a bar directly above the drum. Nothing moves, the LEDs just turn on and off to control where the dots will fall. This process should be cheaper and much less subject to alignment problems than the standard laser devices.

Printer Emulation

This is a tricky one. You're going to have to do some investigation on your own to see what you need and what you're willing to pay for.

Many of the cheapest laser printers ($1800 - $2400) do little more than emulate dot matrix or letter quality printers. They may come with 4 to 10 resident fonts (in ROM), but don't count on uploading more fonts or printing graphics. Only a few of the systems in this price range emulate the HP LaserJet.

The LaserJet's printer control language (PCL) has become a defacto standard for 300 dpi laser printers. Anything that talks to laser printers should at least talk PCL. PCL is kind of a very limited PDL.

The original LaserJet wasn't particularly fancy (128K and 8 built-in fonts) so emulation isn't particularly difficult. You can purchase additional fonts in plug-in cartridges (at about $400 each). Be careful, you can easily spend $4,000 to $10,000 on extra fonts. Also, don't plan on printing graphics larger than a postage stamp.

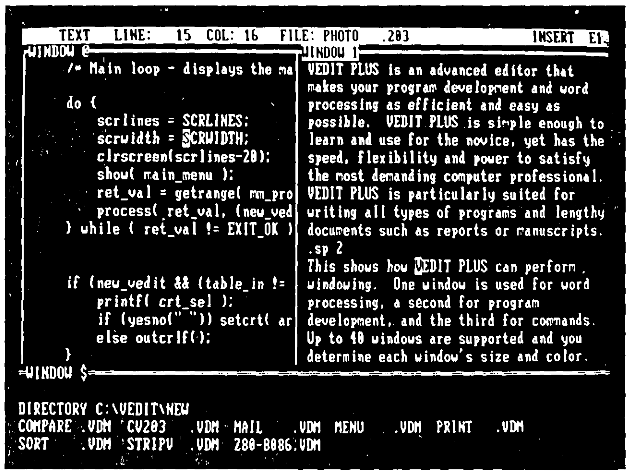
The printers on the next level ($2500 - $4,500) emulate the HP LaserJet Plus. These printers will let you load fonts into their RAM, plus you can also print larger graphics. The LaserJet Plus comes with 512K, but you should try to get at least 1 meg of RAM and have an upgrade path to 2 or 3 meg.

The highest level printers ($4,000 - $8,000) should support PostScript (as well as emulating the HP LaserJet Plus).

(continued on page 24)

New, Lower Prices for CP/M

- **VEDIT Version 1.40...........$49 (Single file, no windows)**
- **VEDIT PLUS Version 2.32......$79 (Multiple file, no windows)**
- **VEDIT PLUS Version 2.33......$95 (Current version with windows)**



```
   TEXT   LINE:   15  COL: 16   FILE: PHOTO   .203                     INSERT  E1
WINDOW @═══════════════════════════════WINDOW 1══════════════════════════════════
     /* Main loop - displays the ma│VEDIT PLUS is an advanced editor that
                                   │makes your program development and word
     do {                          │processing as efficient and easy as
         scrlines = SCRLINES;      │possible.  VEDIT PLUS is simple enough to
         scrwidth = SCRWIDTH;      │learn and use for the novice, yet has the
         clrscreen(scrlines-20);   │speed, flexibility and power to satisfy
         show( main_menu );        │the most demanding computer professional.
         ret_val = getrange( mm_pro│VEDIT PLUS is particularly suited for
         process( ret_val, (new_ved│writing all types of programs and lengthy
     } while ( ret_val != EXIT_OK )│documents such as reports or manuscripts.
                                   │.sp 2
                                   │This shows how VEDIT PLUS can perform
     if (new_vedit && (table_in != │windowing.  One window is used for word
         printf( crt_sel );        │processing, a second for program
         if (yesno(" ")) setcrt( ar│development, and the third for commands.
         else outcrlf();           │Up to 40 windows are supported and you
     }                             │determine each window's size and color.
=WINDOW $═════════════════════════════════════════════════════════════════════════

DIRECTORY C:\VEDIT\NEW
COMPARE .VDM  CV203   .VDM  MAIL    .VDM  MENU    .VDM  PRINT   .VDM
SORT    .VDM  STRIPV  .VDM  Z80-8086.VDM
```

VEDIT PLUS

#1 PROGRAMMABLE EDITOR

FREE Fully Functional Demo Disk *

Stunning speed. Unmatched performance. Total flexibility. Simple and intuitive operation. The newest VEDIT PLUS defies comparison.

Try A Dazzling Demo Yourself.

The free demo disk is fully functional - you can try all features yourself. Best, the demo includes a dazzling menu-driven tutorial - you experiment in one window while another gives instructions.

The powerful 'macro' programming language helps you eliminate repetitive editing tasks. The impressive demo/tutorial is written entirely as a 'macro' - it shows that no other editor's 'macro' language even comes close.

Go ahead. Call for your free demo today. You'll see why VEDIT PLUS has been the #1 choice of programmers, writers and engineers since 1980.

Available for IBM PC, Tandy 2000, DEC Rainbow, MS-DOS, CP/M-86 and CP/M-80. (Yes! We support windows on most CRT terminals, including CRT's connected to an IBM PC.) Order direct or from your dealer. $185.

Compare features and speed	BRIEF	Norton Editor	PMATE	VEDIT PLUS
'Off the cuff' macros	No	No	Yes	Yes
Built-in macros	Yes	No	Yes	Yes
Keystroke macros	Only 1	No	No	100+
Multiple file editing	20+	2	No	20+
Windows	20+	2	No	20+
Macro execution window	No	No	No	Yes
Trace & Breakpoint macros	No	No	Yes	Yes
Execute DOS commands	Yes	Yes	Yes	Yes
Configurable keyboard Layout	Hard	No	Hard	Easy
'Cut and paste' buffers	1	1	1	36
Undo line changes	Yes	No	No	Yes
Paragraph justification	No	No	No	Yes
On-line calculator	No	No	No	Yes
Manual size / index	250/No	42/No	469/Yes	380/Yes
Benchmarks in 120K File:				
2000 replacements	1:15 min	34 sec	1:07 min	6 sec
Pattern matching search	20 sec	Cannot	Cannot	2 sec
Pattern matching replace	2:40 min	Cannot	Cannot	11 sec

Call for 286 / XENIX Version
Fully Network Compatible

- Simultaneously edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname support.
- Horizontal scrolling - edit long lines.
- Flexible 'cut and paste' with 36 'scratch-pad' buffers.
- Customization - determine your own keyboard layout, create your own editing functions, support any screen size.
- Optimized for IBM PC/XT/AT. Color windows. 43 line EGA.

EASY TO USE

- Interactive on-line help is user changeable and expandable.
- On-line integer calculator (also algebraic expressions).
- Single key search and global or selective replace.
- Pop-up menus for easy access to many editing functions.
- Keystroke macros speed editing, 'hot keys' for menu functions.

FOR PROGRAMMERS

- Automatic Indent/Undent for 'C', PL/I, PASCAL, etc.
- Match/check nested parentheses, i.e. '{' and '}' for 'C'.
- Automatic conversion to upper case for assembly language labels, opcodes, operands with comments unchanged.
- Optional 8080 to 8086 source code translator.

FOR WRITERS

- Word Wrap and paragraph formatting at adjustable margins.
- Right margin justification.
- Support foreign, graphic and special characters.
- Convert to/from WordStar and mainframe files.
- Print any portion of file; selectable printer margins.

MACRO PROGRAMMING LANGUAGE

- 'If-then-else', looping, testing, branching, user prompts, keyboard input, 17 bit algebraic expressions, variables.
- Flexible windowing - forms entry, select size, color, etc.
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Menu-driven tutorial

VEDIT and CompuView are registered trademarks of CompuView Products, Inc. BRIEF is a trademark of UnderWare, Inc. PMATE is a trademark of Phoenix Technologies Ltd. Norton Editor is a trademark of Peter Norton Computing Inc.

* Demo Disk is fully functional, but does not readily write large files.

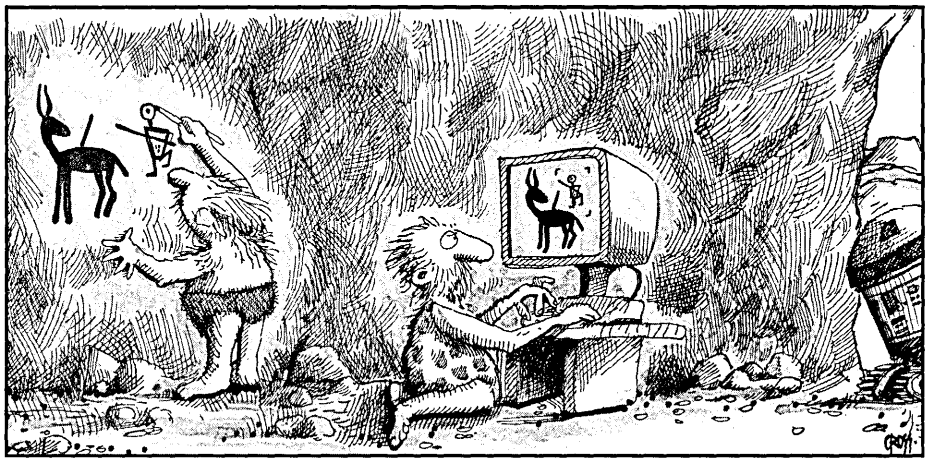
CompuView

1955 Pauline Blvd., Ann Arbor, MI 48103 (313) 996-1299, TELEX 701821

(continued from page 22)

There are other page description languages (such as DDL), but they have fallen out of favor in the past few months as the major players have moved into the PostScript camp.

You probably won't need PostScript support if you're just doing a simple in-house newsletter. But if you're planning to take output from other folks, or if you're proofing copy that will eventually go to a typesetter, then PostScript is probably worth the additional bucks.



If you can wait until the higher resolution laser printers hit the market, then wait.

Buy A Printer Now?

If you can wait until the higher resolution laser printers hit the market, then wait; even if you're only buying a 300 dpi unit. I wouldn't be surprised to see the simplest 300 dpi printers hit $1,000 within six months. The fancier 300 dpi LaserJet Plus and PostScript printers should also come down to the $2,000 range by summer.

I expected to see 400 and 600 dpi printers this last summer. Well, I saw one of each, both distributed by typesetting companies.

Compugraphic makes the 400 dpi printer. It has 6 meg of RAM, comes with 73 fonts (their representative didn't have detail on faces), an Agfa engine, PostScript support, and a 20 meg hard drive. It sells for $29,995. After which you have the privilege of paying at least $325 per month for support. The Compugraphic was designed to work with the Macintosh network.

Varityper makes the 600 dpi unit. It has 6 meg of RAM and a 20 meg hard drive, a 68020 processor, and a Panasonic engine. For your $18,750 you get four faces (their versions of Helvetica, Times, Courier, and symbol) and a chance to pay $3,800 a year for support. If you pass on the support then they charge you $180 for the first hour and $120 for each additional hour for service. The Varityper works with the PC.

I'm waiting for 600 dpi with PostScript for under $10,000. So is every small newspaper, magazine, newsletter, corporation, type house, and mom & pop grocery. Some might even settle for 400 dpi, especially if it's cheap.

How Soon 600+?

No one is saying anything, hinting anything... It's worse than ferreting out what Borland's working on. I've heard rumors about Kyocera, Toshiba, Ricoh, Texas Instruments, HP, you name it.

Actually, it's very likely they are working on a 600 or more dpi unit. Everyone knows what everyone wants and the payoff for the first in the marketplace will be huge. Absolutely huge.

Anyway, in May a TI printer rep called from the factory. He was calling all us editors to make sure we knew about a new TI printer. So, on a lark I responded:

"Is the first shipment on the 400 dpi unit still scheduled for July?"

"Uh, yes, I believe so."

TI didn't release a 400 dpi printer in July, or August. But, it was fun.

Of course not everyone is waiting for higher resolution. Recently I heard the owner of a typesetting house swear softly as he recalled the day a state agency got a 300 dpi model. The agency planned to proof manuals on the laser before sending text in for typesetting; however, they're now using the 300 dpi output for all their manuals.

Bits And Pieces

We're now using Ventura 1.1. It's quite an upgrade with its new hyphenation algorithm, ability to crop line art, fixed width spaces (perfect for listings), overscores, downloading PostScript fonts, font sizes from 1 to 254 points, printing on up to 18 by 24 inch sheets...

However, I've found two serious problems with 1.1 which weren't problems with 1.0. It's harder to read normal-size text on the screen. In fact, anything smaller than 10-point is usually illegible. (Interestingly, it's easier to read the 9-point when you display it smaller than normal.)

Also, the text cursor often gets strange. Put the cursor on a 9-point character and hit the delete key and surprise, another character in the line will disappear. Insert a character and it'll show up somewhere else in the line. In 1.1 they are scaling their screen fonts, but they aren't properly calculating the character widths - so you think you're somewhere, Ventura thinks you're somewhere else.

When the problems occur, it's nearly impossible to edit text (while in Ventura), something I do a lot. Using larger type would be a solution, purchasing custom screen fonts from Bitstream might be another (but it's $290 per set).

If you can live without the new features of Ventura 1.1, you'll probably be happier with 1.0. Otherwise, wait for 1.2.

Ventura Support

The nonprofit Ventura Users Group has taken on the task of researching the hardware and software that can be used with Ventura. They're looking at printers, monitors, printer fonts, and screen fonts. They say they're also helping members with Ventura problems.

I've just received a copy of their monthly magazine and it looks really good. If you use Ventura, you should subscribe ($24 per year).

Ventura Users Group
16160 Caputo Dr.
Morgan Hill, CA 95037 ■■■

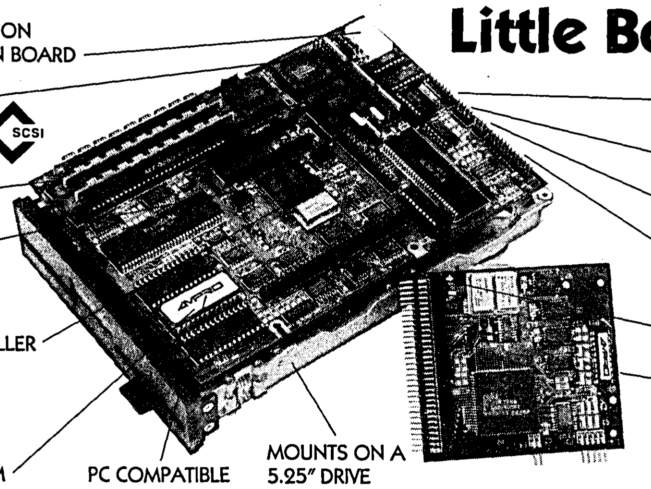
Reliable, Cost Effective Solutions for Computerization

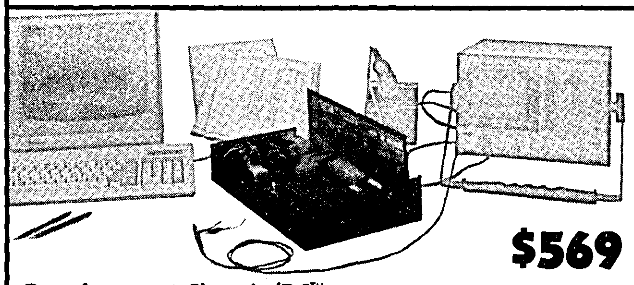
World's smallest PC — and CMOS too!

A Motherboard and 4 Expansion Cards in the Space of a Half-Height 5-1/4″ Disk Drive!

from **$395**
(Qty 100, $336)

Little Board™/PC

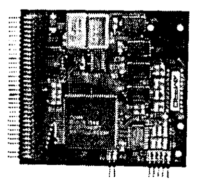

3 WATTS!
5 VOLT ONLY OPERATION
± 12 V GENERATED ON BOARD
SCSI BUS OPTION (HARD DISK, ETC)
SCSI
256 K RAM (512 K, 768 K OPTIONS)
POWERFUL V40 CPU
FLOPPY DISK CONTROLLER (3.5″/5.25″, 360 K/720 K, 1.2 MB)
UP TO 128 K EPROM/RAM/NOVRAM BATTERY BACKED RT CLOCK
PC COMPATIBLE ROM-BIOS
MOUNTS ON A 5.25″ DRIVE (5.75″x8″x1″)
2 RS232C SERIAL PORTS
SPEAKER PORT
KEYBOARD PORT
PARALLEL PRINTER PORT
PC BUS
4 MODE VIDEO CONTROLLER OPTION FITS ENTIRELY WITHIN BOARD DIMENSIONS



$569

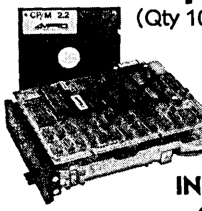
Development Chassis/PC™
"Known Good" PC bus project development environment for Little Board/PC (not included).

$179



CMOS Video Controller
4-mode CMOS video controller for Little Board/PC.

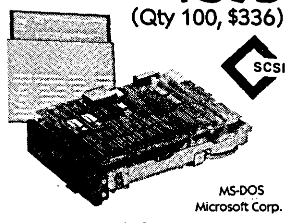
$195
(Qty 100, $166)



INCLUDES CP/M! ™ DRI

Little Board™
World's least expensive single board system.

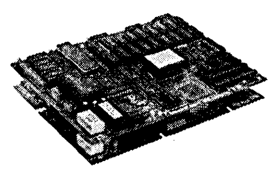
$395
(Qty 100, $336)



MS-DOS Microsoft Corp.

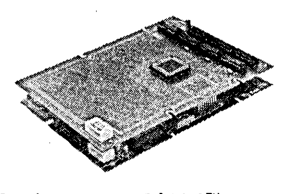
Little Board™/186
High performance single board MS-DOS system.

from **$189**



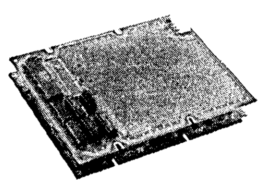
Expansion/186™
Multi-function expansion for Little Board/186. I/O, Serial, RAM, and Math Options.

$119



Project Board/186™
Prototype adapter for 80186 based projects and products.

$99



Project Board/80™
Prototype adapter for Z80 based projects and products.

Z80 TM ZILOG

Distributors • Argentina: Factorial, S.A. 41-0018 • **Australia:** Current Solutions (613) 720-3298 • **Austria:** International Computer Applications GMBH 43-1/45 45 01-0* • **Brazil:** Computadores Compuleader (41) 262-4866 • **Canada:** Tri-M (604) 438-0028 • **Denmark:** Danbit (03) 66 20 20 • **Italy:** Microcom (6) 811-9406 • **Finland:** Symmetric OY 358-0-585-322 • **France:** Egal Plus (1) 4502-1800 • **Germany, West:** IST-Elektronik Vertribes GmbH 089-611-6151 • **Israel:** Alpha Terminals, Ltd. (03) 49-16-95 • **Spain:** Hardware & Software 204-2099 • **Sweden:** AB Akta (08) 54-20-20 • **Switzerland:** Thau Computer AG 41 1 740-41-05 • **UK:** Ambar Systems, Ltd. 0296 435511 • **USA:** Contact Ampro Computers Inc.

AMPRO
COMPUTERS, INCORPORATED

67 East Evelyn Avenue • Post Office Box 390427, Mountain View, CA 94039 • (415) 962-0230 • TLX 4940302

ON YOUR OWN

Pushed Into Business

By Dave Coahran
SE 605D Mckenzie
Pullman, WA 99163
(509) 332-2789

This column is a happy accident. I happened to be talking to Dave (at SOG VI) about another subject when his controller project came up. The more he talked about the way things had fallen into place for his new product, the more it was obvious that he had a story you should all hear.

I didn't wind up "on my own" voluntarily - I was pushed. My friend Dave Demaray's lab instrument manufacturing business had outgrown its moonlighting electronic designer and assembler (me) and needed a full-time person. After 25 years as staff and faculty at Washington State University, I was ready for a change. I went to work for Demaray full time, but within six months Murphy, Gramm-Rudman, and the space shuttle explosion struck and I was an ex-employee of a bankrupt business.

After a few weeks in shock, I saw a way to support my addiction to food. One of Demaray's products had been the Gasograph, an instrument which measures gas production. It's used by breweries, yeast manufacturers, and bakers to measure the activity of yeast. It's mechanically complex and difficult to manufacture, but its users love it.

The demand for the Gasograph was still there, unfilled. Replacing it with a much simpler computer-controlled (now you see why this is in *Micro C*) machine would make it more versatile, easier to use, and perhaps most important, cheaper and easier to manufacture.

My design has 12 jars, each containing a dough sample immersed in a water bath for temperature control. Each jar is connected by a solenoid-driven valve to a measuring jar equipped with a pressure transducer.

For instance, one of the sample jars is connected to the measuring jar, then the connecting valve is closed, the pressure in the chamber measured and recorded in the computer, and then measured gas is vented to the atmosphere. The volume of gas released is calculated from the pressure reported by the transducer and the known size of the measuring jar. The cycle is repeated for each jar.

I decided to subcontract all the machined parts. After all, supporting a machine shop had been a severe drain for Demaray.

I'd do all the design, assembly, and testing. Later, if I couldn't keep up with sales, I would subcontract out more of the machine rather than hire employees. I'd seen the mass of paperwork required of employers and wished to avoid it.

I would have preferred to use the name Gasograph because of its good reputation, but it wasn't clear who owned it. So, regretfully, I chose the name Risograph, suggesting the rising of dough.

Demaray's salesman, John Collins, had started his own business after the company's downfall and was already representing several manufacturers of dough testing equipment. After discussing the Riso with several of his clients, he agreed to represent me.

After some discussion, we agreed on a commission of one-third of the selling price. John would absorb all costs of selling out of his share. Those costs included my expenses while demonstrating the Riso (attending bakers and brewers meetings, etc.). He thought he could sell 20 to 50 machines a year for $8,000 each.

The Gasograph had sold for about $10,000. Even estimating high, I figured my cost to build the Riso would be $2,500 apiece. This left around $2,800 per machine for me.

I didn't have much overhead since I could do the programming, assembly, and initial testing in my apartment and I had all the tools. Eight or ten sales a year would comfortably pay my food, rent, and disks. Even if John's sales estimate was two or three times too high, I'd be okay.

My experience with Demaray really helped. I knew which suppliers could furnish quality material in a reasonable time. I could get delivery on all parts within a couple months. I had capital enough to buy the components for a prototype and a few production machines and to live for several months without an income. I thought it would take three or four months to work out the details and build the first commercial unit. That was May, 1986.

I picked the Radio Shack model 100 portable computer (since replaced by the Tandy 102) as the Riso's controller. This may seem to be an odd choice, but it has several advantages, the primary being simplicity.

Most of the customers have had little (if any) computer experience. I didn't want to intimidate them with disks, operating systems, and A>.

Custom System Mods

I put the program (written mostly in BASIC) safely into ROM, and I brought out the system bus to an external connector for easy interfacing. I use the model 100's RS-232 port to upload data from the Riso into a standard XT. That way customers can analyze the data with their spreadsheets or databases (a terrific selling point).

Of course, the main attraction of the 100 is the price: $500 list, or $400 each in volume purchases. Not bad for a combined controller and terminal.

My printer was another bargain. After looking at several others, I chose the Citizen 120-D printer. It's far more versatile than necessary, durable, good-looking, and surprisingly low-priced.

There seems to be no reasonably priced interface available for the model 100, so I planned to design my own. The information needed to do this was surprisingly hard to find. Radio Shack publishes a service manual which contains useful information, but they were very slow delivering one.

I found a recommendation for *The TRS-80 User's Encyclopedia (Model 100)*, published by The Book Company, and ordered a copy. Eventually, a book on the Radio Shack color computer arrived. The Book Company's phone was disconnected, and they did not answer my letters. Apparently they're out of business.

Finally I got a copy of Carl Oppedahl's *Inside the Model 100*, published by Weber Systems. It is very helpful, though not error free.

With information available at last, I designed and tested an interface which would allow the model 100 to control the "outside world" (13 solenoid valves in this case) and to measure the output of a pressure transducer using an A/D converter. I hope to market the interface in addition to using it on the Riso. (The preceding is a paid commercial announcement.)

Unfortunately, I'd badly underestimated the work involved, and I'd underestimated how much of my time would be absorbed by earning a little income and the day-to-day routines of living. In August, I made the first crude gas production measurements using a tangle of boards, wire, solenoid valves, plastic tubing, and canning jars. Everything worked, pretty much.

By October I had a prototype, quick and dirty on the inside but reasonably good-looking on the outside. I showed my contraption at the annual meeting of the American Association of Cereal Chemists.

My demonstrator, plus my forecast of what the Riso should be able to do, earned a pleasing amount of favorable comment. Meanwhile, John talked up the Riso to his contacts.

After a few weeks in shock, I saw a way to support my addiction to food.

I had written a rudimentary data collection program, but I had nothing to analyze or display. However, the prototype was a good sales tool and it showed me what I needed to improve.

Working with my suppliers, I designed the first batch of six production Risos. The prototype's water bath did not have precise enough temperature control. I had had good experience with Sheldon Manufacturing, and asked them whether they had a bath with better control. They didn't, but had been planning to develop one.

I suggested some special features for their new product. I asked them to install two electrical outlets on the back of the baths. I planned to use two plug-mounted power supplies to run the Riso. That way, nothing beyond the plug would be above 24 volts. This would reduce the chance that someone would be shocked by a Riso. The cost of product liability insurance is so high that I've chosen to go without it.

EEMCO, a valve manufacturer, helped me design a manifold with valves built into it. This is a little more expensive than individual valves and fittings, but greatly reduces assembly work and the chance of leaks. Accra-Fab built the cabinet and made suggestions which improved its design.

By March I had most of the parts for the production machines. The programming was well along, and with the help of the U.S. Department of Agriculture's Western Wheat Quality Lab, I'd used the prototype to run a series of tests on dough. The results, along with suggestions from potential customers, led to changes and additions to the software.

By the beginning of June, the last parts had arrived (like everything else, ironing out the last details of design took longer than expected), and I'd assembled the first production Risos. I sent the program to Portable Computer Support Group to be put on ROM. They shipped me a prototype ROM, it worked correctly, so I ordered ten production ROMS.

Why didn't I burn my own? The model 100 expects to find an assembly language program in the ROM, and I have no desire to write the entire Riso program in assembler. PCSG has developed software which allows BASIC programs to run from ROM. Since BASIC is the model 100's native tongue, the Riso program is written mainly in BASIC, with a few assembly routines where time is critical. Yes, I'm slumming. But it works.

I delivered the first Riso late in June, 1987. Now, a month later, I have delivered four, and prospects look bright. The initial cost estimates were about right, and the sales estimates also seem good. I grossly underestimated the time and effort it would take to get into production. Luckily, I had the resources to survive.

Payment for the first Riso arrived in early August. In reality, "net 30" bills are seldom paid within 30 days (60 to 90 is more typical), and the seller has no effective recourse. Most of the payment will go into parts for the next batch of Risos, but receiving a check for my year's work gave me a big lift.

What's ahead? An extended version of the Riso program to add extra features and polish awkward spots. Current machines can be upgraded by installing the new ROM, of course. Will I go broke, or make a living? I'm optimistic, but time is the only test. Ask me at SOG VII.

■■■

Magic In The Real World:

Tools For Quick System Construction

This is three articles in one. Bruce combines a close look at parallel interface cards, monitoring the real world, and doing it with TSRs. If you don't find something priceless here, it's time to turn in your boots.

Frederick Brooks, author of *The Mythical Man-Month,* said, "First make it work, then make it fast." Put it another way: first make it work, then optimize it.

I'm as guilty as the next guy of building it the hard way - in hardware when I could build it in software, or in assembly language when a high-level language would work fine. We tend not to think of the power and productivity of new tools, but rather to hold on to what's comfortable.

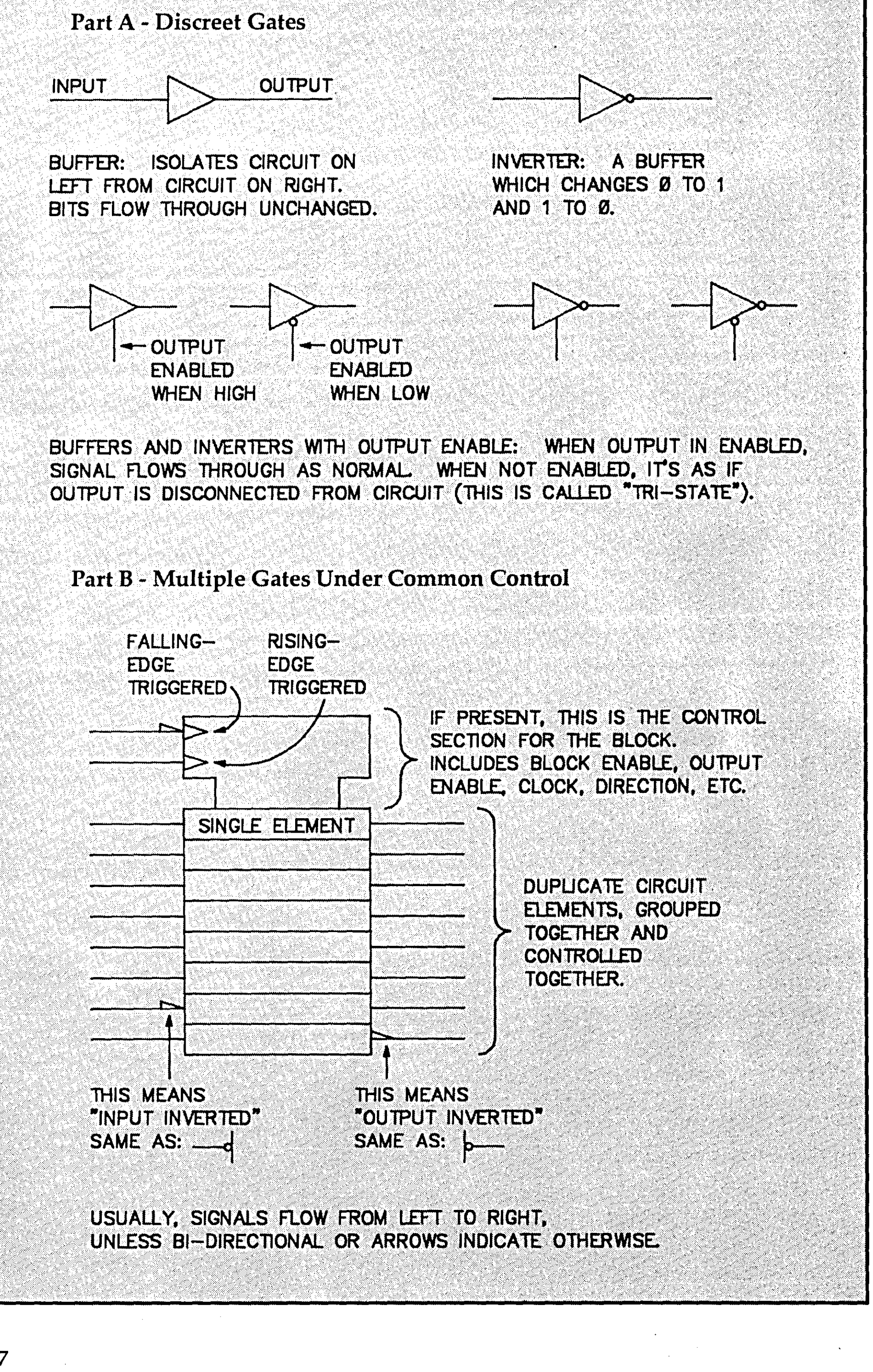
My subject this issue isn't a project. Instead, I'm going to talk about tools: one hardware and one software.

The hardware tool is a cheap parallel printer board from Taiwan which Micro-Sphere sells for $21.

The manual and circuit diagram which come with the board are rather abysmal, but the price is so good, and parallel I/O is so necessary to real-world control (it's been key to all my articles), that I feel it useful to create a complete set of diagrams for the board. This will enable us to both use and modify the board and will let me introduce you to PC adapter card design. (It turns out all printer cards have the same I/O addresses for the same functions, so even if the design of your card is different, you can write software based on these diagrams.)

The software tool is a cheap ($99) C compiler, Turbo C, from Borland. We'll use it (exclusively, without assembly language) to create and install a memory-resident interrupt driver activated by the printer board. To make it do something fun, I wrote a package to create Sidekick-like windows by writing directly to video

Figure 1 - Logic Symbols



Part A - Discreet Gates
INPUT
OUTPUT
BUFFER: ISOLATES CIRCUIT ON LEFT FROM CIRCUIT ON RIGHT. BITS FLOW THROUGH UNCHANGED.
INVERTER: A BUFFER WHICH CHANGES Ø TO 1 AND 1 TO Ø.
OUTPUT ENABLED WHEN HIGH
OUTPUT ENABLED WHEN LOW
BUFFERS AND INVERTERS WITH OUTPUT ENABLE: WHEN OUTPUT IN ENABLED, SIGNAL FLOWS THROUGH AS NORMAL. WHEN NOT ENABLED, IT'S AS IF OUTPUT IS DISCONNECTED FROM CIRCUIT (THIS IS CALLED "TRI-STATE").
Part B - Multiple Gates Under Common Control
FALLING-EDGE TRIGGERED
RISING-EDGE TRIGGERED
IF PRESENT, THIS IS THE CONTROL SECTION FOR THE BLOCK. INCLUDES BLOCK ENABLE, OUTPUT ENABLE, CLOCK, DIRECTION, ETC.
SINGLE ELEMENT
DUPLICATE CIRCUIT ELEMENTS, GROUPED TOGETHER AND CONTROLLED TOGETHER.
THIS MEANS "INPUT INVERTED" SAME AS:
THIS MEANS "OUTPUT INVERTED" SAME AS:
USUALLY, SIGNALS FLOW FROM LEFT TO RIGHT, UNLESS BI-DIRECTIONAL OR ARROWS INDICATE OTHERWISE.

By Bruce Eckel
EISYS Consulting
1009 N. 36th Street
Seattle, WA 98103

memory.

This is a simple application, but it should give you enough foundation to build more complex projects. In this one, we push a button or switch some logic to bring pin 10 on the DB-25 of the printer port to a logic zero, and a window pops up in whatever application is running. The window shows you the status of four of the other printer port lines.

But think of the possibilities: any data-logging, monitoring, control, alarm, etc., system can be run in the background on your PC while you run your regular applications. If you need to know something, the background task tells you instantly.

The Printer Board

I compared the design of four different printer boards (an AST and three cheap ones), and I'm convinced that, from a programmer's viewpoint, all printer cards look the same.

The documentation, however, doesn't. The MicroSphere board comes from Taiwan, famous for manuals which say things like: "Do not agitate the (illegible) or you may have a very great occurrence." There were errors, and the circuit diagram was awful. So I redrew it. Figure 1 shows the logic symbols I used. I've shown discreet gates with the usual symbols (see Figure 1A).

Chips which collect a group of gates under a single control are very common now (since so many circuits are bus-oriented); for those I use the IEC standard shown in Figure 1B. The notched block at the top of the chip represents the control unit for the group, and the rectangles stacked beneath it are the individual gates, drivers, flip-flops, etc.

I haven't shown power and ground pins or bypass capacitors on the diagrams.

(continued on page 31)

NEW LOW PRICE!
$59.95

Eco-C C Compiler

"This is the only package we reviewed that we would be willing to call a professional tool."
Computer Language, Feb., 1985

When the review mentioned above was written, the Eco-C C Compiler was priced at $250.00. Now you can have the same compiler for a mere $59.95. And that price is complete, including a library of 120 functions, all operators (except bit fields), structures, unions, long, floats, doubles, plus user's manual. We've even included a special version of the SLR Systems assembler and linker.

Benchmarks*
(Seconds)

Benchmark	**Eco-C**	Aztec	Q/C
Seive	29	33	40
Fib	75	125	99
Deref	19	CNC	31
Matmult	42	115	N/A

**Times courtesy of Dr. David Clark*
CNC - Could Not Compile
N/A - Does not support floating point

Eco-C requires 56K of free memory, 240K disk space (one or two disk drives or hard disk), Z80 CPU and CP/M 2.2 or later. We also have an MSDOS version at the same low price. Call today!

1-800-952-0472 (orders only)
1-317-255-6476 (information)

ECOSOFT INC. *(317) 255-6476*
6413 N. College Ave. • Indianapolis, Indiana 46220

Full Featured AT Motherboard fits XT or AT case! **$399**

(6/8MHz, $489 for 6/10)

Upgrade your XT to a real AT for about the price of an "accelerator" card

OR

build a space-saving AT from scratch.

Features:
Phoenix BIOS, 1MB memory (ØK installed), VLSI technology, 8 expansion slots, Clock/Calendar.

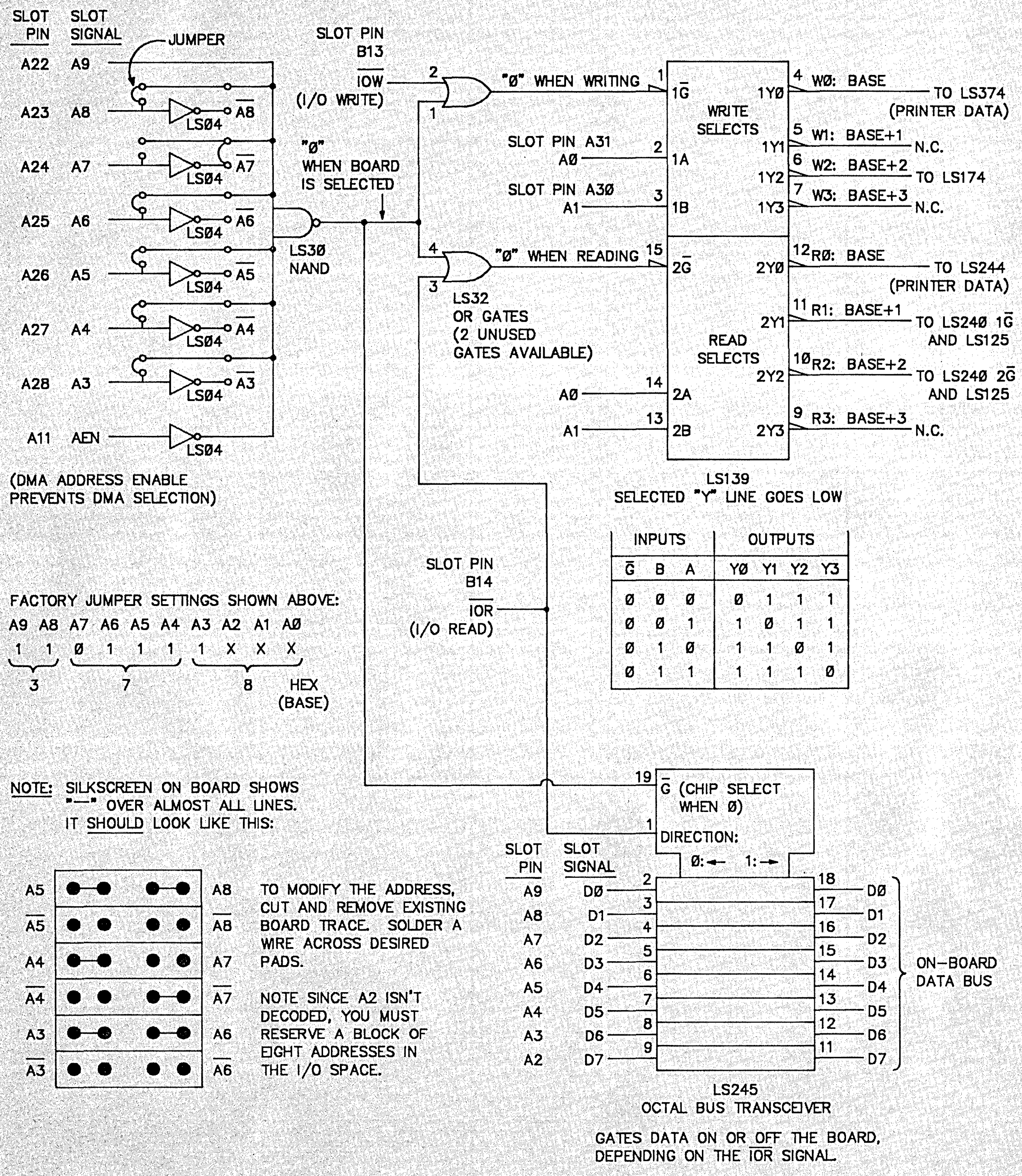
Add $95 for 1MB memory installed and tested.

Other selected components available include

• Toshiba 1.2MB floppy drive	$118
• Floppy/Hard drive controller	$194
• XT size case w/LEDS, lock, reset	$ 55
• Everex Magic I/O Par/Ser card	$ 69
• Everex EGA card	$195
• Mono/Graph/Printer card	$ 69
• Famous Datadesk Keyboard	$119
• High density floppies (10)	$ 10

Call SoftSide Systems at (503) 591-0870

Figure 2 - Parallel Printer Card Bus Interface



INPUTS			OUTPUTS			
$\overline{G}$	B	A	YØ	Y1	Y2	Y3
Ø	Ø	Ø	Ø	1	1	1
Ø	Ø	1	1	Ø	1	1
Ø	1	Ø	1	1	Ø	1
Ø	1	1	1	1	1	Ø

FACTORY JUMPER SETTINGS SHOWN ABOVE:

A9	A8	A7	A6	A5	A4	A3	A2	A1	AØ
1	1	Ø	1	1	1	1	X	X	X

$\underbrace{1\ 1}_{3}\ \underbrace{Ø\ 1\ 1\ 1}_{7}\ \underbrace{1\ X\ X\ X}_{8}$ HEX (BASE)

NOTE: SILKSCREEN ON BOARD SHOWS "—" OVER ALMOST ALL LINES. IT SHOULD LOOK LIKE THIS:

TO MODIFY THE ADDRESS, CUT AND REMOVE EXISTING BOARD TRACE. SOLDER A WIRE ACROSS DESIRED PADS.

NOTE SINCE A2 ISN'T DECODED, YOU MUST RESERVE A BLOCK OF EIGHT ADDRESSES IN THE I/O SPACE.

GATES DATA ON OR OFF THE BOARD, DEPENDING ON THE $\overline{IOR}$ SIGNAL.

(continued from page 29)

Address & Data

Figure 2 shows the address and data interfaces to the PC bus. This card isn't limited to the usual LPT0, LPT1 or LPT2 addresses of most printer cards - you can place it anywhere in the I/O address space (a very nice feature, since we can move it to an "unused" area to prevent interference with other hardware).

The printer card is I/O mapped - you talk to it by using the "port" commands in your programming language:

- the port[] array in Turbo Pascal
- the inportb() and outportb() functions in Turbo C

The card uses a contiguous block of eight I/O addresses. The lowest address (which I refer to as BASE) is set with jumpers on the card; all the other addresses are offsets from BASE.

I have to admire the designers - they know how to save money. You don't change BASE by setting dip switches or pushing jumpers onto pins or by any other friendly activity. You actually have to cut traces and solder jumper wires on the printed circuit board. Well, it IS only $21.

As you can see in Figure 2, we select the address by taking either the inverted or non-inverted address lines from the PC bus and NANDing them together through an LS30. To select the inverted or non-inverted address bit, place jumpers either before or after an LS04 inverter. Note the silkscreen is incorrect on the board - it shows an inversion bar over almost every symbol. I show the corrected version in Figure 2.

To move BASE to an unused area, you first need to locate one. The IBM technical reference manual gives a map of used and free I/O addresses. Other books also give this map (two good ones are: *The IBM PC From The Inside Out*, by Sargent and Shoemaker, and *Interfacing To The IBM Personal Computer*, by Eggebrecht, although the latter tends to have mistakes).

Figure 2 shows that address line A2

(continued on page 33)

Figure 3 - Printer Data Port

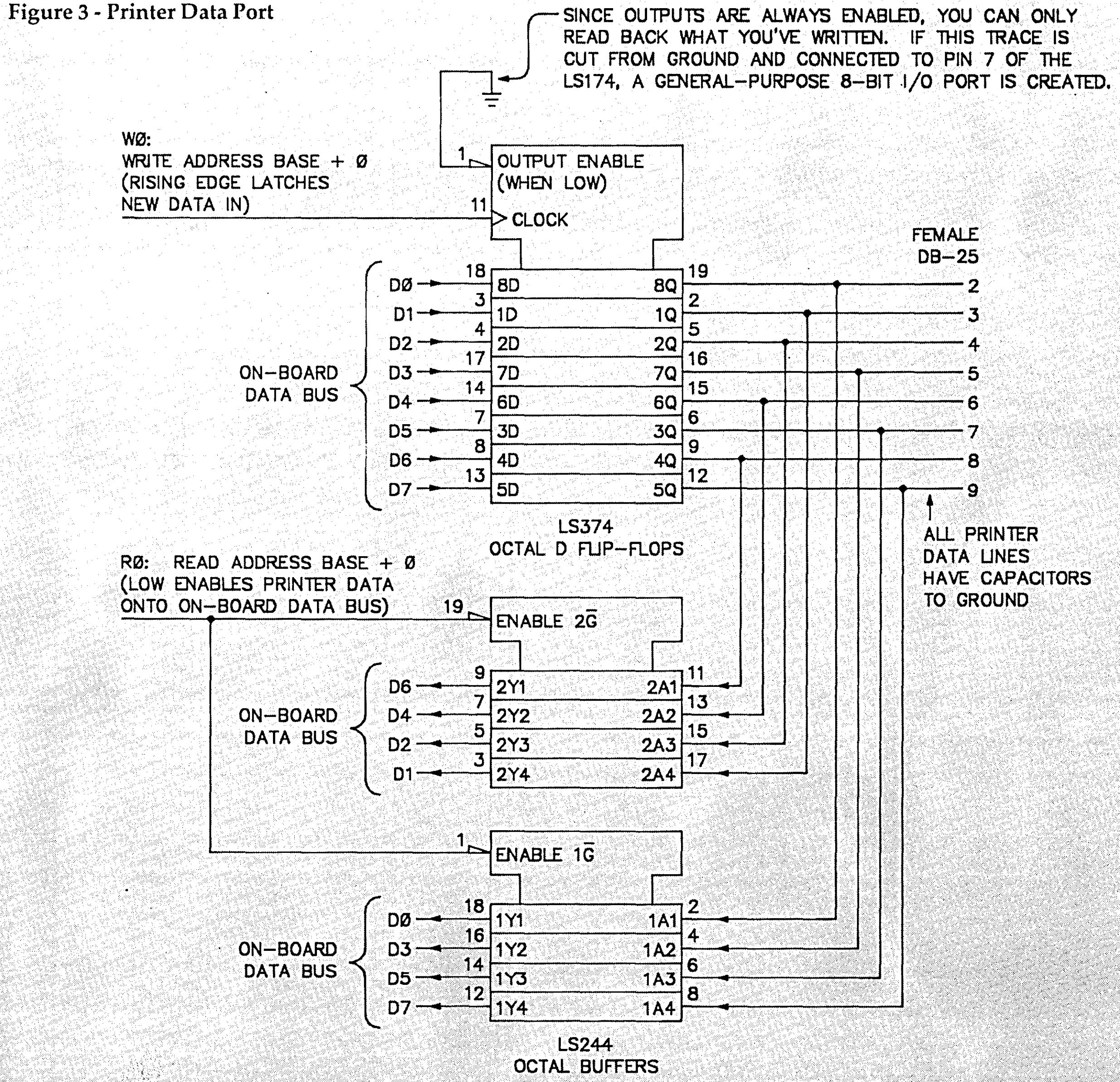
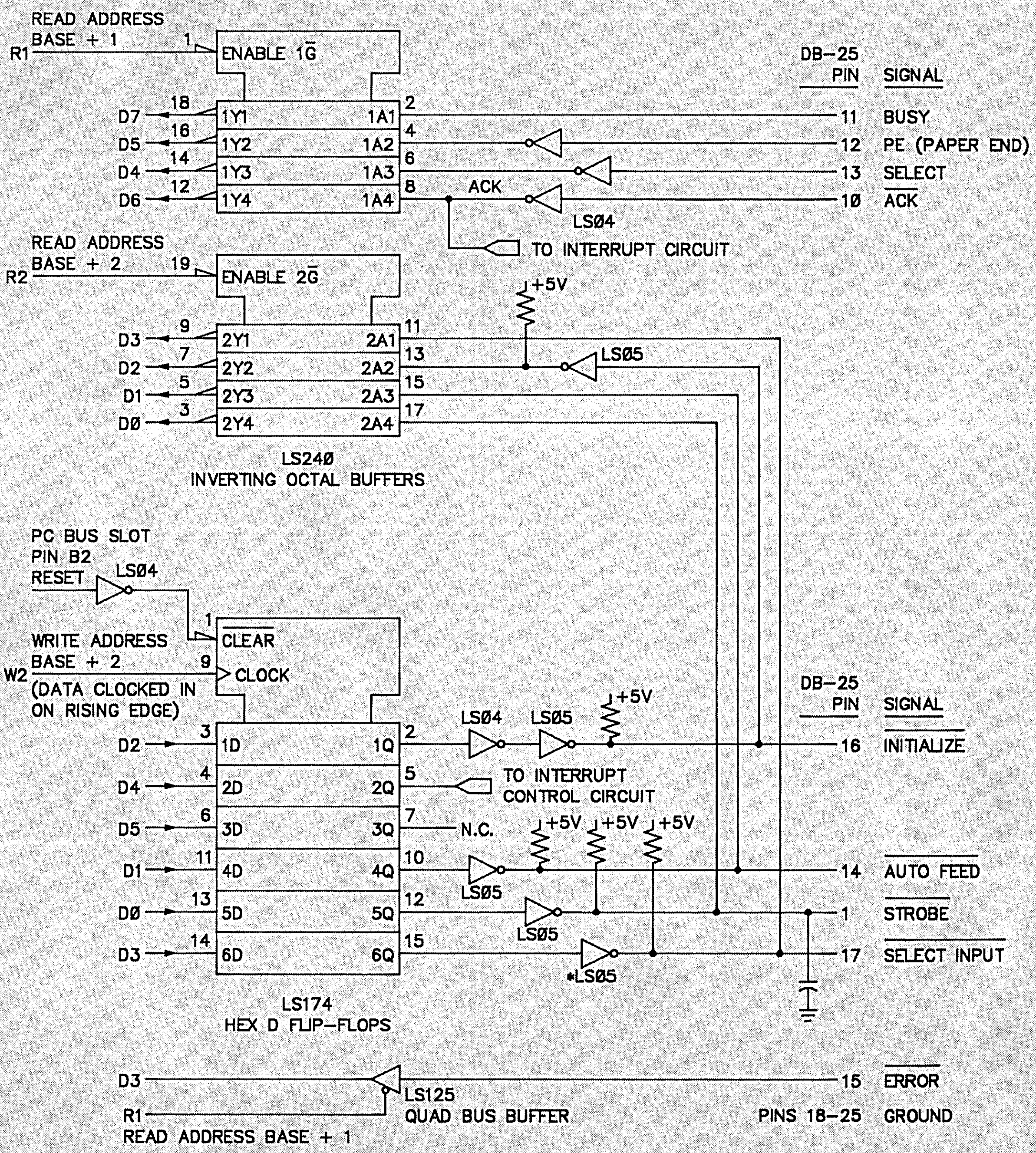

SINCE OUTPUTS ARE ALWAYS ENABLED, YOU CAN ONLY READ BACK WHAT YOU'VE WRITTEN. IF THIS TRACE IS CUT FROM GROUND AND CONNECTED TO PIN 7 OF THE LS174, A GENERAL-PURPOSE 8-BIT I/O PORT IS CREATED.
WØ: WRITE ADDRESS BASE + Ø (RISING EDGE LATCHES NEW DATA IN)
OUTPUT ENABLE (WHEN LOW)
CLOCK
ON-BOARD DATA BUS
D0 D1 D2 D3 D4 D5 D6 D7
8D 1D 2D 7D 6D 3D 4D 5D
8Q 1Q 2Q 7Q 6Q 3Q 4Q 5Q
FEMALE DB-25
LS374 OCTAL D FLIP-FLOPS
ALL PRINTER DATA LINES HAVE CAPACITORS TO GROUND
RØ: READ ADDRESS BASE + Ø (LOW ENABLES PRINTER DATA ONTO ON-BOARD DATA BUS)
ENABLE 2G
2Y1 2Y2 2Y3 2Y4
2A1 2A2 2A3 2A4
ENABLE 1G
1Y1 1Y2 1Y3 1Y4
1A1 1A2 1A3 1A4
LS244 OCTAL BUFFERS

Figure 4 - Printer Status and Control Lines



* LS05's HAVE "OPEN COLLECTOR" OUTPUTS, MEANING WHEN THE OUTPUT IS ZERO, THEY PULL DOWN ALMOST TO GROUND, BUT WHEN IT IS ONE, THE OUTPUTS DISCONNECT THEMSELVES (ALLOWING PINS 1, 14, 16, AND 17 TO BE DRIVEN FROM THE OUTSIDE WORLD). THE "PULL-UP RESISTORS" ARE NECESSARY TO PULL THE OUTPUTS UP TO A LOGICAL ONE.

(continued from page 31)

isn't decoded, which means you need to reserve a block of eight addresses instead of four - the upper four are images of the lower four.

Once you've located the new BASE address, change it to binary and determine the bit values of A3 through A8 (A0-A2 should be 0; A9 is fixed at 1). If a bit value is 1, you must use the positive jumper for that address line. If it's 0, use the complement. To change a bit, cut a section from the trace with an X-acto knife, then solder a short jumper wire across the desired solder pads.

The rest of Figure 2 is straightforward: the LS32 OR gates select reading or writing. These signals, along with address bits A0 and A1, are fed into the LS139 to generate read or write signals (sent to other chips on the board) for BASE, BASE+1, BASE+2 or BASE+3. The LS245 gates the eight-bit data bus on or off the board (the remainder of the diagrams refer to a "data bus" - this is the "printer board data bus" connected to the LS245).

Figure 3 shows the printer data port. This simply writes the data bus out to pins 2-9 on the DB-25 connector (what you plug your printer cable into) and reads it back again. The funny thing is, you can only read back what you've written. By grounding pin 1 on the LS374, the outputs of that chip are always enabled (i.e., always driving), so there isn't any way to use those pins to read data from the outside world.

Figure 4 shows the printer status and control lines. Notice the LS174 has an unused flip-flop which is nonetheless still connected to the data bus. If we cut the trace from pin 1 on the LS374 (Figure 3) and connect that pin to pin 7 of the LS174 (Figure 4), the printer data port becomes a general purpose I/O port.

To make it an input, we write a 1 to data bit 5 of BASE+2, which disables the outputs of the LS374 so they can read the outside world. This modification is about as simple as changing the BASE address.

But you don't need to make this mod to get input lines. The board comes with nine pins you can already read (so you can connect the A/D converter from issues #34 and #35). These are shown in Figure 4. Five of the pins (DB-25 pins 10, 11, 12, 13 and 15) are read-only pins, while four of them (1, 14, 16 and 17) can be read from or written to. The read and write addresses and data bits are shown in Figure 4.

(continued next page)

XT CLONE SYSTEMS
(One YEAR guarantee on system)

Turbo Mother Board 4.77 and 10 MHz
640 K Ram installed on board
Serial, Parallel, Game Ports
Clock/Calendar   AT Style Keyboard
Color Video Board (CGA)   Monochrome Opt
150 Watt Power Supply   Flip Top Case

<u>ABOVE</u> WITH 2 FLOPPY DISK DRIVES	**$ 699.00**
WITH 1 FLOPPY AND 20 MEG	**$ 950.00**
WITH 2 FLOPPY AND 20 MEG	**$1000.00**

Assembled and Tested for 24 Hours

AT TURBO SYSTEM

AT COMPATIBLE MOTHER BOARD WITH BIOS
6 MEG AND 10 MEG SWITCHABLE SPEED
512K RAM INSTALLED UP TO 1024 ON BOARD
WA2 HARD DISK/FLOPPY DISK CONTROLLER
MONOCHROME GRAPHICS VIDEO WITH PRINTER
1.2 MEG OR 360 K FLOPPY
220 WATT POWER SUPPLY   AT CASE
AT KEYBOARD   SET UP DISK

ONE YEAR WARRANTEE ON SYSTEM	**$1095.00**
EGA UPGRADE FOR ABOVE	$ 100.00
512K UPGRADE (1024 INSTALLED)	$ 50.00
5339 KEYBOARD UPGRADE	$ 30.00

HARD DISK DRIVES

20 Meg Seagate ST4026	(for AT)	$ 495.00
30 Meg Seagate ST4038	(for AT)	$ 595.00
30 Meg CDC Wren 1	(for AT)	$ 595.00

Does NOT include controller

Color Monitor RGB (CGA)	$ 300.00
Color Monitor RGB (EGA)	$ 410.00
Monochrome TTL (Green)	$ 110.00
Monochrome TTL (Amber)	$ 120.00
EGA Color Video Card	$ 195.00
MS DOS 3.1	$ 50.00

CITIZEN PRINTERS

MODEL 120D	120 CPS	9"	$ 200.00
MODEL MSP-10	160 CPS	9"	$ 300.00
MODEL MSP-15	160 CPS	15"	$ 400.00
MODEL MSP-20	200 CPS	9"	$ 350.00
MODEL MSP-25	200 CPS	15"	$ 500.00
MODEL 35	35 CPS LETTER QUALITY		$ 500.00

ALL PRINTERS COME WITH CABLE

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

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

(continued from page 33)

The read/write pins in Figure 4 are rather interesting. They use LS05 inverter gates, which have "open collector outputs." This means when the output is 0, it pulls down; but when the output is 1, it just lets go of the line. You'll note each LS05 has a "pull-up" resistor on the output - this pulls the line up to 1 (5V, in this case) when the output lets go.

Open-collector outputs used to be very common. They allowed many devices to share a bus (the bus had pull-ups on every line), but you had to make sure the outputs of a device went to 1 when it was finished with the bus, or else that line would be permanently pulled to 0 and you'd have problems. Tri-state logic, which releases the bus whenever the outputs aren't enabled (regardless of their value), is much more idiot-proof, more common, and doesn't require pull-up resistors. (The three states are: pull the line high, pull the line low, and don't do anything with the line.)

The pull-up resistors come in handy, though. To read pins 1, 14, 16 and 17, the outputs of the LS05s must be at 1. To do this, we write to BASE+2 with data bits 0, 1 and 3 at 0 (because of the inverters) and bit 2 at 1 (because of the extra inverter, which makes sense if you stare at Figure 4 long enough). This causes the LS05s to disconnect, so the pull-up resistors take over. You now have four "read" pins with internal pull-up resistors.

Why is this nice? Normal TTL inputs tend to float high, but you can't just put a switch between that input and ground and read the line to see if the switch is open or closed. It would work okay most of the time, but it's susceptible to noise and I wouldn't recommend it. The best practice is to pull the input up to +5V with a pull-up resistor, and then use a switch to pull it to ground. This, however, requires an external power supply or some way to get power out of the PC (which is entirely possible).

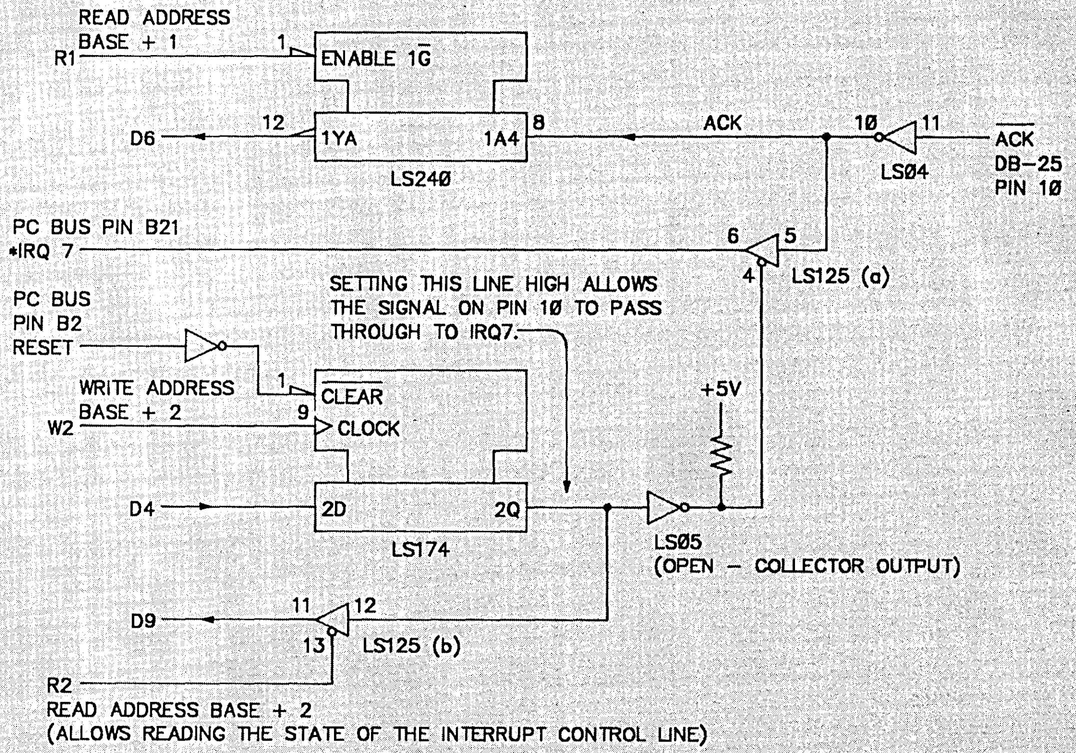
The four read/write lines already have pull-ups on them. So if you can get away with reading only four switches, your circuit is very simple. The other lines, of course, may be driven with TTL or CMOS outputs without worrying about pull-up resistors. (You must, however, make sure the grounds from both systems are connected together.)

Otherwise, a 1 to the first system might look like a 0 to the second. Everything is relative, so you must give them both the same starting point from which to compare 0s and 1s. (Using 4000-series CMOS to drive the pins with pull-ups on them is out of spec for that type of CMOS, so I wouldn't recommend it.)

Generating Printer Interrupt

Pin 10 (-ACK. Signals with a bar over them are sometimes typographically rep-

Figure 5 - Parallel Port Interrupt Circuit


READ ADDRESS BASE + 1
R1
ENABLE 1G
D6
1YA
1A4
LS240
ACK
LS04
ACK DB-25 PIN 10
PC BUS PIN B21 *IRQ 7
LS125 (a)
SETTING THIS LINE HIGH ALLOWS THE SIGNAL ON PIN 10 TO PASS THROUGH TO IRQ7.
PC BUS PIN B2 RESET
WRITE ADDRESS BASE + 2
W2
CLEAR
CLOCK
+5V
D4
2D
2Q
LS174
LS05 (OPEN - COLLECTOR OUTPUT)
D9
LS125 (b)
R2
READ ADDRESS BASE + 2 (ALLOWS READING THE STATE OF THE INTERRUPT CONTROL LINE)

* SINCE ALL THE PINS ON THE MICROSPHERE CARD ARE ETCHED ON THE EDGE CONNECTOR (UNLIKE SOME EXPENSIVE CARDS), IT IS POSSIBLE TO CHANGE THIS WIRE TO AN UNUSED INTERRUPT IF YOU DO IT CAREFULLY (IRQ7 IS USED BY PRINT SPOOLERS). UNUSED INTERRUPTS ARE:

INTERRUPT	SLOT PIN
IRQ2	B4
IRQ3	B25
IRQ5	B23

resented with leading '-' signs) serves double duty: it can simply be read, or if you initialize the proper circuitry, it will generate IRQ7 when pulled to ground. You can test the circuit with a push-button switch connecting pin 10 to ground, but the "production" circuit should have a pull-up resistor (a few k-ohms) to +5V on the input.

Figure 5 shows the interrupt circuit. If you write a 1 to data bit 4 of BASE+2, the inverted signal from pin 10 will pass through LS125 'a' to pin B21 on the PC bus, which is IRQ7. Reading BASE+2 shows the status of this interrupt control line on data bit 4. All the gold fingers are on the card, so with careful soldering you can move to a different interrupt (see Figure 5).

Simply passing the interrupt through to the PC bus isn't enough. You also have to configure the PC's 8259A interrupt controller to accept the interrupt (see Larry Fogg's article, "Hardware Interrupts On The PC," in issue #36), create an interrupt handler, and put the address of the handler in the interrupt vector table. I'll do this in the code.

Cabling

To get the wires where you want them, you can either cut the Centronics end off a printer cable (they're cheap enough to make this feasible) and figure out which wires are which with a continuity tester (multimeter, or light bulb, battery and piece of wire) or buy a male DB-25S to ribbon cable connector at your local electronics store. The advantage of the latter method is the ribbon cable keeps track of the pins for you.

The DB-25 on the printer board has tiny numbers to help you discover which pins are which.

The Code

The Turbo C program (see Figures 6A-6D, beginning next page) is a "terminate-and-stay-ready" (TSR) interrupt handler which installs itself as interrupt 15 (which is where IRQ7 goes - I was very disappointed when I tried installing the handler for interrupt 7 and nothing happened). It enables the printer board to pass pin 10 through to IRQ7 and reconfigures the 8259A to accept the hardware interrupt. This, I believe, is exactly the way background print routines work; the routine is reawakened when the printer is finished with what it is doing and lowers the -ACK line.

What makes building an interrupt handler simple is Turbo C's "interrupt" compiler directive and the keep() function. When you declare a function as "interrupt," the compiler creates code which saves the stack at the beginning and restores it at the end, and uses the "return from interrupt" assembly instruction to leave instead of the usual "return from subroutine." Because of this, you don't need to write any assembly language to create an interrupt handler.

Personally, my eyes glaze over whenever I see long (or short, even) assembly listings in magazines - it's just too much work to extract information, and I'm too mistake-prone to type the thing in and test it. I love anything which, like Turbo C, allows you to do more with less effort. Tools like this let us be artists instead of just technicians!

TSR.C is the definition and installation of the interrupt routine int_handler(). Main() calls Turbo C's setvect() function to place the address of the interrupt handler into the interrupt vector table. It then configures the printer board, the interrupt controller, and calls keep() to create a TSR.

The keep function is a call to the MS-DOS TSR facility, with most of the dirty work taken care of. All we do is supply the size of the program we want to keep. To determine the size, use Turbo C's options:linker:mapfile set to "segments." Then look at the last address in TSR.MAP (which is generated during compilation), and that's the program size.

This is my first time writing a TSR so everything may not be kosher. For instance, I didn't re-enable interrupts inside the handler, so if (for instance) you're accessing the disk when the interrupt occurs, the disk transfer is held up. I haven't found any problems because of this, but who knows - my clock may be slowed down or something.

The interrupt handler creates a Sidekick-like window on the screen. To do this without crashing into DOS, I created a windows package (in WINDOWS.C) to directly access screen memory. There are functions to define windows, paint them on the screen and write to them, as well as a "save_screen()" and "restore_screen()" which preserve whatever was interrupted.

The file WINDOWS.H (the header file for WINDOWS.C) uses a new ANSI extension to C called function prototyping. This makes function declarations much more familiar to Pascal types, and it allows the compiler to check for errors when you make a function call. I like this feature a lot.

To control screen colors, I created a header file called COLORS.H, which defines all the possible colors on the CGA. These tell the window which character and background colors to use.

EGA cards are appearing which emulate Hercules along with CGA, but I jumped into the clone fray too early for those so I'm just assuming you have CGA.

A Gotcha

One problem I encountered, and haven't yet figured out, is passing pointers to functions inside of the interrupt routine. You'll notice a strange function called window_put_binary(), which puts a literal string "1" or "0" into a window depending on the bottom bit of an integer.

When I tried passing a pointer to either of these strings, it worked but corrupted the array where the interrupted screen was saved. This seems odd and I haven't figured out whether it's a compiler bug or my fault, but things seem to work okay if I put everything in as literals. Ugly, but operational.

Some Details

When you compile this code, make sure you set the BASE '#define' at the beginning to your card's address.

You should also create a "project" file. This is like the UNIX "makefile" but much simpler. Use the "Project" option on Turbo C's main menu to tell the compiler what project file you are using. File TSR.PRJ looks like this: TSR (WINDOWS.H COLORS.H) WINDOWS (COLORS.H).

This means: there are two C files involved in creating this file: TSR.C and WINDOWS.C. If you change a C file, that file should be recompiled and the whole system re-linked. TSR.C depends on two header files: WINDOWS.H and COLORS.H. If you change either of those files, recompile and re-link TSR.C. WINDOWS.C depends on COLORS.H.

The Future

Next time I plan to use the printer port and Turbo C to create simple remote data-acquisition and control modules which use a four-wire synchronous serial interface.

Have any ideas for projects you'd like to see or problems you need solved? Please send them (to me and Abby).

■■■

CIRCUIT BOARDS

PCB-Edit... creates multi-layered PCB's with ease. Included are solder mask and legend ink support, plotter -- printer drivers MUCH MORE **$99.95**

PCB-Shop... will build double sided, plated thru holes, circuit boards from your artwork or PCB-Edit files for only $1.00 per square inch (single quantities) with no set up charges.

ANALOGIC... the 32 channel logic analyzer for the IBM PC. Has 16 bit trigger word, 80 nano second sample time, for only.

FULL UNIT	BARE BRD
$399.95	**$99.95**

MOVER... the two axis stepper motor driver controlled by your printer port. only:

FULL UNIT	BARE BRD
$179.95	**$49.95**

ANALOGIC
PHONE(602) 458-4065
BOX 3228, Sierra Vista, AZ 85636

68000
and 6800/2/8/9

SOFTWARE

SK*DOS - a powerful DOS for the 6809 ($75) or the 68000 ($140, incl. an editor, assembler, Basic, utilities, code for a boot ROM, etc.)
HUMBUG - a monitor/boot ROM, $50 - $75.
OTHER SYSTEM SOFTWARE including assemblers, text formatters, editors, spell checkers, languages, etc., all very reasonable.

HARDWARE

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

COMBINATIONS

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

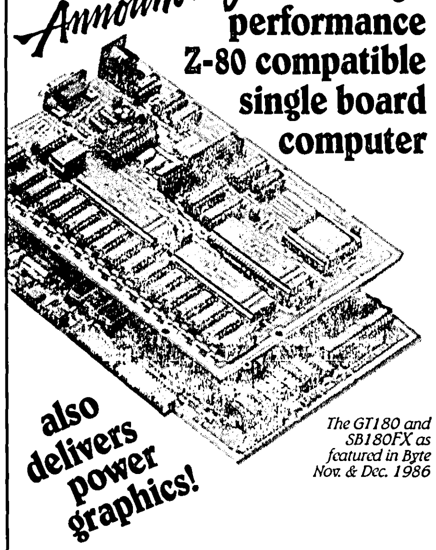
STAR-K
SOFTWARE SYSTEMS CORP.
BOX 209 • MT. KISCO, NY 10549
914/241-0287

Figure 6 - TSR, Windows, & Color Routines In C

```
/* Figure 6a -- TSR.C */

/* TSR.C: Terminate and Stay Resident (TSR) program written entirely in Turbo C
*  ("look, ma, no assembly language!"). Bruce Eckel, Eisys Consulting,
*  1009 N. 36th St., Seattle, WA 98103.  7/87.
*      The interrupt line on the parallel printer card (-ACK: pin 10 on the
*  DB-25 connector) is allowed through to IRQ7, and the 8259A is configured to
*  service the interrupt when -ACK is pulled to ground using a simple switch,
*  TTL or CMOS logic.
*      This was tested with an EGA monitor in color mode, so it probably works
*  with a color monitor or a monochrome monitor (for mono, change SCREEN_BASE
*  to 0xB000.  Note I have gotten around using any DOS or BIOS calls by writing
*  direct screen driver routines -- this prevents any possible collision
*  with the program being interrupted.
*      I tried using the interrupt while running an EGA program (Dr Halo).  The
*  results were...interesting.  Someday maybe I'll figure that one out
*  (Sidekick worked, but it left garbage on the canvas).
*/
/* #include <dos.h> */
#define INT_NUMBER 15  /* interrupt number to install this function into.
                          Note IRQ7 on the PC card bus corresponds to interrupt
                       handler 15 in the interrupt vector table.  */
#define PROG_SIZE 0x1E10  /* Run the Turbo C compiler with the options:
                        linker:mapfile set to "segments."  Look at the
                        mapfile generated for this program.  The "stop"
                        address for the stack is the highest address
                        used -- set PROG_SIZE to this value for use with
                        the "keep()" command */
/*  I tried using the "tiny" model instead of the "small" model, since it is
    supposed to be more size-efficient.  The map showed the size to be LARGER,
    even though it didn't create a stack and the small model did.  Seems odd. */

#include "colors.h"      /* color definitions */
#include "windows.h"     /* prototypes for window function definitions.
 These tell the compiler: (A) not to panic if a function is called for which
 it hasn't seen the definition, and (B) what the function's calling convention
 is, so the compiler can tell you if you're doing it right (helps find errors).*/

#define BASE 0x378        /*  Parallel port board base address, established for
                             LPT1.  Change this if you're using LPT2 or you
                             changed the jumpers on the cheap card.  */

#define PIC_OCW1 0x21     /*  8259A Programmable Interrupt Controller
                              Operation Control Word 1 (see issue 36, page 36) */

/* macro to read value of ACK line */
#define ACK (inportb(BASE + 1) & BIT6)

/*************************************************************************/
/*  Macro (with parameters) to read values at any pin location.  Returns  */
/*  "1" if a logical one comes back from the masked pin, and "0" if a     */
/*  logical zero comes back.  Note inversions due to hardware logic are   */
/*  not considered, so some pins may be seeing a TTL "1" and return a     */
/*  zero (see circuit).    */
#define READ_BIT(port_address,bit) (inportb(port_address) & bit ? 1 : 0)

/*************************************************************************/
/*  The interrupt handler.  Notice the 'interrupt' compiler directive,    */
/*  which tells the compiler to save and restore all the registers and    */
/*  use interrupt code instead of normal subroutine code.                 */
void interrupt int_handler()
{
#define LEFT_X   5    /* Window boundaries */
#define RIGHT_X  75
#define TOP_Y    5
#define BOTTOM_Y 20
/* Function prototype, to allow compiler to flag improper function calls: */
void window_put_binary (int window_number, int value, unsigned char attribute);

/* save the user's screen and put ours up instead */
save_screen();
define_window(0,LEFT_X,TOP_Y,RIGHT_X,BOTTOM_Y, YELLOW_CHAR | BROWN_BACK);
draw_window(0);
title_window(0,"Process Monitor Interrupt", RED_CHAR | GREEN_BACK);
/* If you want to add sound here, see the Turbo C user's guide, page 275 */

/* wait for the -ACK line to rise before returning.  Meanwhile, read all
    the input lines (for example) with pullup resistors on them.   */
    while (!ACK) {    /* see how much neater the macro makes things? */
         window_gotoXY(0,LEFT_X+3,TOP_Y+2);
         window_puts (0,"pin 1 : ", BLACK_CHAR | BROWN_BACK);
         window_put_binary (0,READ_BIT(BASE+2,BIT0),
                            BLUE_CHAR | BROWN_BACK);
         window_gotoXY(0,LEFT_X+35,TOP_Y+2);
         window_puts (0,"pin 14 : ", BLACK_CHAR | BROWN_BACK);
         window_put_binary (0,READ_BIT(BASE+2,BIT1),
                            BLUE_CHAR | BROWN_BACK);
         window_gotoXY(0,LEFT_X+3,TOP_Y+7);
         window_puts (0,"pin 16 : ", BLACK_CHAR | BROWN_BACK);
         window_put_binary (0,READ_BIT(BASE+2,BIT2),
                            BLUE_CHAR | BROWN_BACK);
         window_gotoXY(0,LEFT_X+35,TOP_Y+7);
         window_puts (0,"pin 17 : ", BLACK_CHAR | BROWN_BACK);
         window_put_binary (0,READ_BIT(BASE+2,BIT3),
                            BLUE_CHAR | BROWN_BACK);
   }

   /* restore the user's previous screen */
   restore_screen();

   /* tell the 8259A Interrupt Controller we are finished executing IRQ7 */
   outportb(0x20,0x67); /* specific EOI for IRQ7 */
}

/*************************************************************************/
/*  main() for TSR.  This installs the interrupt, sets up the hardware,   */
/*  and exits leaving the program resident.  Main is never used again.    */
main()
{
    setvect(INT_NUMBER,int_handler);  /* passes the ADDRESS of the beginning
                                         of the int_handler() function.
                                         setvect() is a Borland function. */

   /* change the bit on the parallel board to allow the -ACK interrupt to pass
      through to IRQ7 on the PC card bus.  BIT4 allows the interrupt to pass;
      BIT2 allows pin 16 to be read.  See circuit diagram. */
      outportb(BASE + 2, BIT4 | BIT2);

   /* zero top bit of OCW1 to allow IRQ7 to be serviced.  Note we get the
      current OCW1, force the top bit to 0 and put it back out -- this retains
      the rest of the word (which affects other aspects of the machine) to
      prevent undesirable side effects. */
      outportb(PIC_OCW1, inportb(PIC_OCW1) & 0x7f);

      keep(0,PROG_SIZE);  /* first parameter is exit status. See Ref. Manual. */
}

/*************************************************************************/
/*  Takes the bottom bit of an integer and prints it as an ascii string   */
/*  either "0" or "1".  For some reason, I had trouble passing string     */
/*  pointers inside the TSR -- it would modify the save_buf[] array -- so */
/*  it seems I can only use literals.  Be aware of this quirk -- I spent  */
/*  awhile chasing it and couldn't find out what it was.                  */
void
window_put_binary (int window_number, int value, unsigned char attribute)
{
 if (value & 1)
    window_puts (window_number,"1 ", attribute);
 else
    window_puts (window_number,"0 ", attribute);
}

/* Figure 6b -- WINDOWS.C */

/* WINDOWS.C: custom screen functions to prevent DOS collisions inside the TSR.
*  Bruce Eckel, Eisys Consulting, 1009 N. 36th St. Seattle, WA 98103  7/87
* "DOS calls!?  We don't need no steenkin DOS calls!..."  You can bet if
*  Borland had made DOS, it would have been re-entrant and relocatable, and we
*  wouldn't have this problem, or be limited to 640K on an AT.
```

(WINDOWS listing continued next page)

Announcing **A high performance Z-80 compatible single board computer**



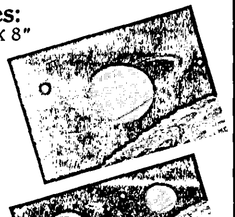
The GT180 and SB180FX as featured in Byte Nov. & Dec. 1986

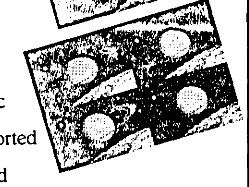
The SB180 FX

- *Small, fast, memory-packed single board computer*
- *add the Micromint GT180 for high resolution graphics*

SB180 FX features:
- Measures only 5.75" x 8"
- 64180 CPU running at 6, 9 or 12 Mhz
- Up to 512K bytes RAM and 32K bytes ROM
- Two 38.4 baud serial ports
- A parallel printer port
- Peripheral expansion bus
- Three bi-directional parallel ports
- Industry standard 765A - compatible disk controller
- NCR 53C80-SCSI bus controller for hard disk or network communications

GT180 FX features:
- Measures only 5.75" x 8"
- Designed to piggy-back on top of the SB180 or SB180 FX
- High resolution of 640x480x16 colors from a palette 4096
- Advanced HD63484 CRT controller
- 38 commands including 23 graphic drawing commands
- Fully software supported by Borland's GT180 Graphix Toolbox and Modula-2
- 2 million pixels per second





SB180 FX . . . . . . .as low as $409.00
GT180. . . . . .as low as $395.00

Turbo Modula-2. . . . .$69.00
Turbo Modula-2 w/GT180 Graphix Toolbox. $89.00

To order call
1-800-635-3355
TELEX: 643331
For technical information call 1-(203)-871-6170

MICROMINT, INC.
4 Park St., Vernon, CT 06066

FORTHkit

5 Mips computer kit

$400

Includes:

Novix NC4000 micro
160x100mm Fk3 board
Press-fit sockets
2 4K PROMs

Instructions:

Easy assembly
cmFORTH listing
shadows
Application Notes
Brodie on NC4000

You provide:

6 Static RAMs
4 or 5 MHz oscillator
Misc. parts
250mA @ 5V
Serial line to host

Supports:

8 Pin/socket slots
Eurocard connector
Floppy, printer,
video I/O
272K on-board memory
Maxim RS-232 chip

Inquire:

Chuck Moore's

Computer Cowboys

410 Star Hill Road
Woodside, CA 94062
(415) 851-4362

(WINDOWS listing continued)

```
*  Note I haven't put anything in to change the screen modes -- you are assumed
*  to already be in CGA.  I also haven't turned the cursor off and on.
*/

#undef TEST      /* '#define' this to make a stand-alone test program */
#include "colors.h"  /* CGA color #defines */
/* global place to save the screen so we can restore it when we're done: */
unsigned char save_buf[SCREEN_CHARS];

/* Now, a place for all the global attributes for each window: */
struct window_def {  /* coordinates start at upper left corner as 0,0 */
       int left_x;   /* left-most extent of window */
       int top_y;    /* upper limit of window */
       int right_x;  /* right-most extent of window */
       int bottom_y; /* lower limit of window */
       unsigned char attributes;  /* default char and background colors */
       int cursor_x; /* for functions which need cursors in the window */
       int cursor_y;
       } window[3];  /* To add more windows, increase the array size */

#define WINDW window[window_number]  /* saves typing and makes things clearer */

/*************************************************************************/
/*  This function simply initializes a window[] structure, so all other   */
/*  functions told to do something to that window can look up all the    */
/*  necessary information about it.                                       */
void define_window(int window_number, int left_x, int top_y, int right_x,
                int bottom_y, unsigned char attribute)
{
WINDW.left_x = left_x;  /* establish global window values */
WINDW.top_y = top_y;
WINDW.right_x = right_x;
WINDW.bottom_y = bottom_y;
WINDW.attributes = attribute;
WINDW.cursor_x = left_x + 1; /* put cursor inside box */
WINDW.cursor_y = top_y + 1;
}

/*************************************************************************/
/* Puts a character and its attribute anywhere on the screen.            */
void putc_at_location(char ch, int x, int y, unsigned char attribute)
{
pokeb(SCREEN_BASE,((y * SCREEN_WIDTH) + x) * 2,ch);
pokeb (SCREEN_BASE,(((y * SCREEN_WIDTH) + x) * 2) + 1, attribute);
}

/*************************************************************************/
/* Clears a window (including the border), retaining it's attributes.    */
void clear_window(int window_number)
{
unsigned int x,y;
   for (y = WINDW.top_y; y <= WINDW.bottom_y ; y++){
       for (x = WINDW.left_x; x <= WINDW.right_x; x++){
       putc_at_location(' ',x,y,WINDW.attributes);
      }
   }
}

/*************************************************************************/
/* Puts a string in a window, wrapping if it hits a border and refusing  */
/* to go past the lower right corner.  For some reason, the string must  */
/* be a literal (i.e. '"a string"', rather than a pointer you pass)      */
/* embedded in the function call or bad things happen when the           */
/* interrupted screen is restored.                                       */
window_puts(int window_number, char *string, unsigned char attribute)
{
int cursor_offset;
do {
    if (*string == 10) { /* check for newline */
       WINDW.cursor_x = WINDW.left_x + 1;
       WINDW.cursor_y += 1;
       if (WINDW.cursor_y > WINDW.bottom_y - 1)
          WINDW.cursor_y -= 1; /* just bump against the bottom if you run out */
    }
    else
```

```
     {
      putc_at_location(*string, WINDW.cursor_x, WINDW.cursor_y, attribute);
      /* Move cursor ahead, but keep it inside window */
      if (++WINDW.cursor_x > WINDW.right_x -1) {
         WINDW.cursor_x = WINDW.left_x + 1;
         if (++WINDW.cursor_y > WINDW.bottom_y -1)
             WINDW.cursor_y--;
      }}}
while (*++string); /* stops at the string's null terminator */
}

/***************************************************************************/
/* Saves the screen we just interrupted into a global array.             */
void
save_screen()
{
int scr_offset;
   for(scr_offset = 0; scr_offset < SCREEN_CHARS; scr_offset++) {
      save_buf[scr_offset] = peekb(SCREEN_BASE,scr_offset);
   }
}

/***************************************************************************/
/* Restores the interrupted screen from the global array.                */
void
restore_screen()
{
int scr_offset;
   for(scr_offset = 0; scr_offset < SCREEN_CHARS; scr_offset++) {
      pokeb(SCREEN_BASE,scr_offset,save_buf[scr_offset]);
   }
}

/***************************************************************************/
/* Puts a box of double bars (like Sidekick) around the window, using the */
/* window's pre-defined character and background colors.                 */
void make_box(int window_number)
{
int x,y;

for (x=WINDW.left_x, y=WINDW.top_y; x++ < WINDW.right_x; )      /* top bar */
    putc_at_location(0xCD,x,y,WINDW.attributes);
for (x=WINDW.left_x, y=WINDW.bottom_y; x++ < WINDW.right_x; )  /* bottom bar */
    putc_at_location(0xCD,x,y,WINDW.attributes);
for (x=WINDW.left_x, y=WINDW.top_y; y++ < WINDW.bottom_y; )     /* left bar */
    putc_at_location(0xBA,x,y,WINDW.attributes);
for (x=WINDW.right_x, y=WINDW.top_y; y++ < WINDW.bottom_y; )   /* right bar */
    putc_at_location(0xBA,x,y,WINDW.attributes);

    /* bottom left corner */
putc_at_location(0xC8, WINDW.left_x, WINDW.bottom_y, WINDW.attributes);
   /* top left corner */
putc_at_location(0xC9, WINDW.left_x, WINDW.top_y, WINDW.attributes);
    /* top right corner */
putc_at_location(0xBB, WINDW.right_x, WINDW.top_y, WINDW.attributes);
   /* bottom right corner */
putc_at_location(0xBC, WINDW.right_x, WINDW.bottom_y, WINDW.attributes);
}

/***************************************************************************/
/*  Puts window up if it isn't already; clears it if it is.               */
void draw_window(int window_number)
{
clear_window(window_number);
make_box(window_number);
}

/***************************************************************************/
/*  Centers a title in the foreground and background colors of your choce.*/
/*  Title is placed in the top bar of the window.                         */
void title_window(int window_number, char *title, unsigned char attribute)
{
char *title_ptr;
int title_count, x;
make_box(window_number); /* redraw box if new title is smaller than old one */
```

(WINDOWS listing continued on next page)

Now it's easier than ever to order from Micro Cornucopia!

Use your Visa or Mastercard and call:

1-800-888-8087

ORDERS ONLY

9 AM - 5 PM, Monday - Friday - Pacific Time

XEROX 820-1 AND 820-2 ITEMS
Reconditioned, Assembled and Tested

820-1 8" COMPUTER SYSTEM	$330.00
5 1/4" COMPUTER SYSTEM	$350.00
820-2 8" COMPUTER SYSTEM	$395.00
5 1/4" COMPUTER SYSTEM	$415.00
820-1 COMPUTER MONITOR (COMPLETE)	$125.00
820-2 COMPUTER MONITOR (COMPLETE W/CONTROLLER)	$195.00
820 COMPUTER MONITOR (NO MAIN BOARD)	$ 85.00
HIGH PROFILE KEYBOARD (COMPLETE)	$ 45.00
820-1 MAIN COMPUTER BOARD	$ 50.00
FULLY POPULATED BOARDS, AS IS (NEED REPAIR)	$ 20.00
820-2 MAIN COMPUTER BOARD	$ 70.00
FULLY POPULATED BOARDS, AS IS (NEED REPAIR)	$ 30.00
820-2 FLOPPY CONTROLLER BOARD	$ 95.00
DUAL 8" SSDD DISK DRIVES/ENCLOSURE (COMPLETE)	$175.00
DUAL 8" DISK DRIVE CABINET (NO DRIVES)	$ 75.00
5 1/4" DUAL DISK DRIVE CABLE	$ 20.00
8" DUAL DISK DRIVE CABLE	$ 35.00
RS-232 CABLES	$ 10.00

LINE CORDS	ea.$3.00
Z80-B 6MHz	ea.$3.00
Z80-H 8MHz	ea.$9.50
5 1/4" DSDD DISKETTES	ea.$ .60
8" SSSD DISKETTES	ea.$1.25
DC300A DATA CART..USED	2/$5.00

E^{2}I COMPUTER PRODUCTS
2273 AMERICAN AVE. #8
HAYWARD, CA 94545
(415) 786-9203

TERMS: Pre-payment, COD, Visa/Mastercard. California residents add sales tax. Orders are FOB Hayward, CA.. Shipments by UPS Ground unless otherwise requested. Prices and availability are subject to change without notice. All products are assembled and tested and have a 30 day warranty unless otherwise stated. Call or write for current product and price listing. Xerox is a trademark of Xerox Corporation. CP/M is a trademark of Digital Research.

ICs PROMPT DELIVERY!!!
SAME DAY SHIPPING (USUALLY)
QUANTITY ONE PRICES SHOWN for AUG. 23, 1987

OUTSIDE OKLAHOMA: NO SALES TAX

640K MOTHERBD UPGRADE: Zenith 150, IBM PC/XT, Compaq Portable & Plus; hp Vectra

DYNAMIC RAM			
1Mbit	1000Kx1	100 ns	$26.50
1Mbit	256Kx4	120 ns	32.00
51258	*256Kx1	100 ns	6.95
4464	64Kx4	150 ns	3.50
41256	256Kx1	80 ns	4.95
41256	256Kx1	100 ns	4.40
41256	256Kx1	120 ns	3.40
41256	256Kx1	150 ns	3.20
41264	2-PORT	120 ns	5.25
EPROM			
27512	64Kx8	200 ns	$11.25
27C256	32Kx8	250 ns	6.65
27256	32Kx8	250 ns	5.50
27128	16Kx8	250 ns	4.95
STATIC RAM			
43256L-12	32Kx8	120 ns	$11.95
5565PL-15	8Kx8	150 ns	3.25

* STATIC COLUMN DRAM FOR 386 COMPAQ

80387-16 $540.00
80287-8 $250.00
8087-2 $160.00

OPEN 6½ DAYS, 7:30 AM-10 PM: SHIP VIA FED-EX ON SAT.

SUNDAYS & HOLIDAYS: SHIPMENT OR DELIVERY, VIA U.S. EXPRESS MAIL

SAT DELIVERY INCLUDED ON FED-EX ORDERS RECEIVED BY:
Th: Std Air $4/1 lb
Fr: P-1 $10.50/2 lbs

MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts μP∞
MICROPROCESSORS UNLIMITED, INC.
24,000 S. Peoria Ave.,
BEGGS, OK. 74421
(918) 267-4961
No minimum order. Please note that prices are subject to change. Shipping & insurance extra, & up to $1 for packing materials. **Orders received by 9 PM CST can usually be delivered the next morning, via Federal Express Standard Air @ $4.00, or guaranteed next day Priority One @ $10.50!** All parts guaranteed.

(WINDOWS listing continued)

```
for (title_ptr = title, title_count = 1; *++title_ptr;
title_count++)
    ; /* count number of chars in string (stops when
*title_ptr == '\0') */
   /* starting x value to center the title */
x = (WINDW.right_x - WINDW.left_x - title_count)/2 +
WINDW.left_x + 1;
while (*title)  /* stops when *title == '\0' */

putc_at_location(*title++,x++,WINDW.top_y,attribute);
}

/*******************************************************/
/* Move the window cursor within the window, repecting
the boundaries.    */
void window_gotoXY(int window_number, int X, int Y)
{
/* If X isn't outside of the window bounds, set cursor
   to X, else set it to just inside the window bounds */
if (X > WINDW.left_x && X < WINDW.right_x )
   WINDW.cursor_x = X;
else
   if (X <= WINDW.left_x)
      WINDW.cursor_x = WINDW.left_x +1;
   else
      WINDW.cursor_x = WINDW.right_x -1;

/* same for y */
if (Y > WINDW.top_y && Y < WINDW.bottom_y )
   WINDW.cursor_y = Y;
else
   if (Y <= WINDW.top_y)
      WINDW.cursor_y = WINDW.top_y +1;
   else
      WINDW.cursor_y = WINDW.bottom_y -1;
}

#ifdef TEST
main()
{
/* Here's where you put calls to window routines when
performing stand-alone
*  tests.  Of course, just because they work here
doesn't mean they will work
*  in a TSR.  You also need a '#define TEST' at the
beginning of this file.
*/
}
#endif

/* Figure 6c -- COLORS.H */

/* COLORS.H: definitions for CGA screen characteristics
and colors */
#define SCREEN_BASE 0xb800   /* base address of color
graphics card (and EGA
                                    in color graphics mode
*/
#define SCREEN_HEIGHT 25
#define SCREEN_WIDTH 80
#define SCREEN_CHARS (SCREEN_WIDTH * SCREEN_HEIGHT * 2)
       /* number of chars and attributes in a screen */

#define BIT0 0x01      /* bit masks */
#define BIT1 0x02
#define BIT2 0x04
#define BIT3 0x08
#define BIT4 0x10
#define BIT5 0x20
#define BIT6 0x40
#define BIT7 0x80
```

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

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

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

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

Your programs will be using the most efficient searching techniques.
B+trees use efficient search techniques that require fewer disk seeks than other methods. CBTREE guarantees an optimized maximum search path and always remains balanced. CBTREE is optimized for speed. You will be using a commercial quality, reliable and powerful tool. CBTREE is a full function implementation of the industry standard B+tree access method and is proven in applications since 1984.

Access any record or group of records by:
- its absolute position in the index (GETFRST and GETLAST),
- its relative location in the index (GETPRV, GETNXT and GETSEQ),
- an exact match to a key (GETREC),
- a partial match to a key (GETPAR, GETALL and GETKEYS)
- a lexical relation to a key (GETLT, GETLE, GETGT and GETGE).
- You may also add, delete and update any record without the need to reorganize the index (INSERT, ISRTKY, DELETE, DELTKY and NEWLOC).
- Block retrieval calls speed up sequential processing.

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

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

CBTREE IS TROUBLE-FREE, BUT IF YOU NEED HELP WE PROVIDE FREE PHONE SUPPORT.
ONE CALL GETS YOU THE ANSWER TO ANY QUESTION!
CBTREE compares favorably with other software selling at 2,3 and 4 times our price.
Sold on unconditional money-back guarantee.
YOU PAY ONLY $99.00 - A MONEY-SAVING PRICE!
TO ORDER OR FOR ADDITIONAL INFORMATION
CALL (703) 356-7029 or (703) 847-1743
OR WRITE

PEACOCK SYSTEMS, INC.

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

```
/* Make a complete attribute by ORing a CHARacter
   type with a BACKground type */

#define BLUE_CHAR  BIT0
#define GREEN_CHAR BIT1
#define RED_CHAR   BIT2
#define INTENSE    BIT3
#define BLUE_BACK  BIT4
#define GREEN_BACK BIT5
#define RED_BACK   BIT6
#define BLINKING   BIT7

#define BLACK_CHAR 0
#define CYAN_CHAR (GREEN_CHAR | BLUE_CHAR)
#define MAGENTA_CHAR (RED_CHAR | BLUE_CHAR)
#define BROWN_CHAR (RED_CHAR | GREEN_CHAR)
#define WHITE_CHAR (RED_CHAR | GREEN_CHAR | BLUE_CHAR)
#define GRAY_CHAR (INTENSE | BLACK_CHAR)
#define LIGHT_BLUE_CHAR (INTENSE | BLUE_CHAR)
#define LIGHT_GREEN_CHAR (INTENSE | GREEN_CHAR)
#define LIGHT_CYAN_CHAR (INTENSE | CYAN_CHAR)
#define LIGHT_RED_CHAR (INTENSE | RED_CHAR)
#define LIGHT_MAGENTA_CHAR (INTENSE | MAGENTA_CHAR)
#define YELLOW_CHAR (INTENSE | BROWN_CHAR)
#define BRIGHT_WHITE_CHAR ( INTENSE | WHITE_CHAR)

#define BLACK_BACK 0
#define CYAN_BACK (GREEN_BACK | BLUE_BACK)
#define MAGENTA_BACK (RED_BACK | BLUE_BACK)
#define BROWN_BACK (RED_BACK | GREEN_BACK)
#define WHITE_BACK (RED_BACK | GREEN_BACK | BLUE_BACK)
#define GRAY_BACK (INTENSE | BLACK_BACK)
#define LIGHT_BLUE_BACK (INTENSE | BLUE_BACK)
#define LIGHT_GREEN_BACK (INTENSE | GREEN_BACK)
#define LIGHT_CYAN_BACK (INTENSE | CYAN_BACK)
#define LIGHT_RED_BACK (INTENSE | RED_CHAR)
#define LIGHT_MAGENTA_BACK (INTENSE | MAGENTA_CHAR)
#define YELLOW_BACK (INTENSE | BROWN_BACK)
#define BRIGHT_WHITE_BACK ( INTENSE | WHITE_BACK)

/* Figure 6d -- WINDOWS.H */

/*  WINDOWS.H: #include these function prototypes at
 *  the beginning of any file where you use windows
 *  functions.  Turbo C will then catch errors if you
 *  call the functions improperly.  (to "make" properly,
 *  you also need to mention windows.c in your
 *  "project" file).
 */

extern void define_window(int window_number,
      int left_x, int top_y, int right_x, int bottom_y,
      unsigned char attribute);

extern void putc_at_location(char ch, int x, int y,
                              nsigned char attribute);
extern void clear_window(int window_number);
window_puts(int window_number, char *string,
            unsigned char attribute);
extern void save_screen();
extern void restore_screen();
extern void make_box(int window_number);
extern void draw_window(int window_number);
extern void title_window(int window_number,
                         char *title, unsigned char attribute);
extern void window_gotoXY(int window_number, int X, int Y);


       /* End of Figure 6 */
```

PASCAL PROCEDURES

Image Scanner Part II Construction Tips

By John Paul Jones
6245 Columbia Ave.
St. Louis, MO 63139

I'm as excited about this project as you are. A $6 scanner is just the ticket for those of us with desktop or other graphic packages. Herein John covers scanner construction and calibration as well as some details about XT graphics displays.

This time I'll spend the entire column on my $6 scanner. (You'll need to have the schematic in issue #37 handy so you can check component values and identifiers.)

Construction

I've constructed my scanner on two small pieces of perf board using point-to-point wiring.

One board (the one mounted on the print head carrier), should have only R1, R2, R3 and the sensor. Before you start, however, remove the print head from the printer (power OFF please) and see what you've got to work with.

Plan ahead, the board should not come near any other part of the printer through the entire platen length. After cutting the board to size, drill for the print head mounting screws, the sensor mounting hole, and anything else needed. (I had to allow for a pair of locating pins on the head carrier.) Now you can figure where the parts go and begin construction.

Do NOT solder the sensor to the board; connect with wires and leave some slack to allow for later sensor to platen gap adjustment. Be prepared for some "gotchas." On my assembly the head mounting screws have oversize heads and the sensor needed to be trimmed to allow for them. A FLEXIBLE three-wire cable goes from this board to the remaining circuit. I used a piece of ribbon cable.

Test mount the sensor assembly on the head carrier. For best results, the center of the sensor should be at the same level as the center pin of the print head. If necessary, use washers to raise the board.

While the sensor is mounted, manually move the carriage back and forth. Be absolutely sure that the board and cable will not touch, snag, rub or otherwise fondle any part of the printer. You can DESTROY your printer with a mechanical jam!!!

For neatness, mount the second board in a small project box, but leave the lid off for now. Try to use less than six feet of cable between this board and the joystick input connector.

Preliminary Calibration

Adjust R1 to maximum (minimum LED current). Set up a calibration target with both black and white areas. Black permanent marker on white paper is okay for this. Power up the circuit, and with a voltmeter check the voltage at pin 1 of the LM324 when the sensor is aimed at each area.

For mine, I got 0.38 V for black and 1.3 V for white. Uh oh, too much gain on the op amp! The minimum gain can be reduced by reducing R4; if R4 = 0 there is no gain. Don't replace R4, use a parallel R to reduce its value. We may need the gain later on if we have to reduce the aperture of the sensor.

Adjust R7 so that the voltage at LM339 pin 5 is a little below the "white" voltage at LM324 pin 7. Now, with the program in Figure 1, you should be able to manually scan between light and dark areas and see the values change be-

Figure 1 - Routine to Calibrate Scanner

```
Program calibrate;
const
      game = $201; { Joystick Port }
var
      was , is : byte;
begin
 clrscr;
 was := 0;
 repeat
  is := port[game] shr 4;
             {data in upper 4 bits}
  if is <> was then {been a change?}
  begin {if yes, show the new value}
was := is;
write(^M,was:3);
  end;
 until keypressed;   {graceful exit}
end.
```

tween 15 and 0. Remember that as each brightness level is reached, the lower level bits remain on so the values you will see are 0, 1, 3, 7 and 15. "Tweak" R1, R5 (if you have any gain on the op amp), and R7 so that you get full range.

Now put it all away until we get the software under control.

Of Rasters and Pixels

We know where we're coming from, a picture on paper we want on the graphics screen. This is done with a process called rasterization, which converts a continuous image into a series of lines (rasters) of dots (pixels). By repeatedly scanning horizontally and stepping vertically, we will build an array of pixels which represent the original image.

We then want to display the image on a "standard" graphics display. Unfortunately, there are currently three standards, and depending on how well PS/2 sells there may be a fourth. The three current standards are Color Graphics Adapter (CGA), Hercules Graphics Adapter (HGA), and Enhanced Graphics Adapter (EGA). Let's look at how these actually display an image on the screen.

All three are memory mapped. A portion of the computer's address space is shared between the processor and the display circuitry, each memory location makes up a portion of the displayed image.

CGA is the most common, since it can feed either a composite monitor (either color or monochrome) or an RGB monitor. Sixteen K bytes at B000:0000 (segment:offset) are assigned to the CGA. In its maximum resolution (640 pixels X 200 lines), it supports only monochrome.

The pixels in each scan line are contained in 80 sequential memory locations, each bit of which controls one pixel.

(continued on next page)

68000
Single Board Computer
$249.95

☐ 8 Mhz 68000 ☐ 256 K bytes RAM (0 wait state) (expandable to 512 K bytes) ☐ up to 128 K bytes ROM
☐ 2 Serial Ports ☐ Parallel Port
☐ Floppy Controller
☐ SCSI Controller (optional)
☐ Buffered full-speed expansion bus
☐ +5 volts only ☐ 5¾" x 8"
☐ ROM Monitor includes Assembler, Disassembler, full memory test, etc.

Companion Memory Board
$249.95

☐ 512 K bytes (expandable to 8 megabytes)
☐ 0 wait state ☐ Interfaces to Single Board Computer

Marion Systems Corporation
1317 5th St. Suite 301, Santa Monica, CA 90401
(213) 451-8910

OMNIVID™

WINDOWS! SPRITES! ROW TABLE SCREENS!
FOR THE '83 KAYPRO II OR 4, XEROX 820-1, AND BIG BOARD-1!

THE OMNIVID DEVICE DRIVERS

Blazing fast assembly language device drivers make impressive screens quick and easy! No complicated code to write. Just send simple escape sequences to your Console Output routine. Multiple screens, each with its own windows and sprites, may be held in memory for instant use! OMNIVID™ makes it simple. And, with FLEXUTIL.COM (provided) it's quick and easy to create an executable .COM file which loads your selection of the OMNIVID™ device drivers in protected upper TPA, above the BIOS, or at any absolute address. Information on how to write and include your own device drivers is also provided!

Easy to use with most High Level Languages. Toolboxes provided for Sbasic® and Turbo Pascal®. TURTLE.BOX (also included for use with Turbo Pascal®) brings TURTLE GRAPHICS ability.

ALSO INCLUDED

* A device driver which supports a MICROSOFT compatable serial (RS-232) mouse. (Can be configured to support the popular LOGITECH C7 serial mouse.) Now you can write mouse driven programs with pop-ups and pull-downs for your Z80 computer!!
* A device driver for dumping the screen to your printer.
* SLOTS.COM - A realistic slot machine game demonstrates ROW TABLE DRIVEN SCREENS.
* JAWS - A game to demonstrate the ability to animate using the OMNIVID™ device drivers.

Available now for the '83 Kaypro II, 4, Xerox 820-1, Big Board-1.
Soon for the '84 series Kaypros and PC compatables.

ONLY 49.95 (includes shipping)
KY residents add 5% Sales Tax
VISA, MC, or COD accepted

Call or write for info. Or order documentation on disk for only $8.00 (includes shipping).

ALSO AVAILABLE

A GRAPHICS CHARACTER ROM for your 83 Kaypro II or 4, Xerox 820-1, or Big Board-1 supplements the OMNIVID™ device driver package with a thin-line and pixel graphic (160 x 48) character set. As well as providing better looking window borders and an improved alphanumeric character set, the pixel graphic character set is highly desirable for use with TURTLE.BOX (above). ROM PRICE 19.95 (includes shipping).

TEL (606) 325-3736 9AM-6PM EST

FLEXISOFT · 3987 VALLEY VIEW DRIVE · ASHLAND, KY 41101 TM

(continued from page 43)

Memory to screen mapping is interleaved - even scan lines in the lower 8 K, odd lines in the upper. This means that although bytes within a scan line are contiguous, the scan lines are not. For example: in screen coordinates, the upper left corner is X = 0, Y = 0. This pixel is located at B000:0000, bit 7. The pixel immediately below it, at X = 0, Y = 1, is at address B000:2000, bit 7. It's critical to keep track of where we are!

HGA (Hercules) is the only "non-IBM" graphics mode to become popular. It provides monochrome only, at 720 X 348. The adapter has 64 K bytes of memory, organized as two display "pages." Memory to screen mapping is interleaved by four. Each page is accessed by the display circuit as four 8 K banks, one each for scans whose number MOD 4 are 0, 1, 2 and 3. Again, within each line, displayed bytes are sequential.

Pixel	Mapped to (page 0)
0,0	B000:0000 bit 7
0,1	B000:2000 bit 7
0,2	B000:4000 bit 7
0,3	B000:6000 bit 7

This is a little more complex than CGA, but not too bad.

The EGA has all of the CGA's modes and adds several of its own. For the first run software, we'll use the 640 X 350 monochrome mode. This is the simplest of the three since there is no interleave. The screen is represented by a linear array at address A000:0000. It gets more difficult to use the attribute plane for this mode, so we may get into that later.

The Software Plan

Figure 2 is the overall outline for the software portion of the project. As you can see, certain areas are preliminary and will need to be expanded as we get farther along. There will not be a direct relationship between the organization of the outline and the program; I use the outline more to define what needs to be done, not how to do it.

For now, I'll discuss two low-level definition modules and in the process show how you can isolate portions of a project while at the same time generalize the program.

Since we want the program to run on all three display adapters with as little customization as possible, we want to keep the hardware-specific code isolated

(continued next page)

Figure 2 - Pseudo Code For Scanner Program

```
A. Initialization

    1) Clear data areas
        a) Global variables
            1> Image pixel array
            2> Scan line temporaries
            3> Option flags

    2) Image border
        a) Query white or black
        b) Set border pixels

    3) Calibrate sensor (separate program)

    4) Initialize printer
        a) Query printer ready
        b) Reset string to printer
        c) ? one line of null graphic data ?

    5) Take over clock interrupt


B. Capture - Display

    1) Scan line of image
        a) Output line of null graphic data
        b) <CR> to begin printhead movement
        c) Delay (empirically determined amount)
        d) Activate capture on clock interrupt
        e) Capture n data points
        f) De-activate capture on clock interrupt
        g) Fractional <LF>

    2) If enough data, preliminary processing
        a) Straight monochrome (two color) mode
            1> 'Dithering' of captured pixels based on analog value
                a> '0' - pixel quad all 0
                b> '1' - weight corners based on adjoining 3 pixels
                    a: if > 1 weight = 3, random placement
                    b: if none > 3 AND > 1 = 2, random placement
                    c: if none >= 2, random among 1's
                    d: if all = 0, random placement
                c> '2' - weight pixel pairs based on 6 adjoining pixels
                d> '3' - weight corners based on adjoining 3 pixels
                    logic as for '1', with reverse weights
                e> '4' - pixel quad all 1
        b) Gray scale mode
            1> No processing - pixel value = input value

    3) Display processed pixels


C. Store to Disk

    1) Query filename

    2) Save image to file

    3) Query exit/repeat
```

ERAC CO.

8280 Clairemont Mesa Blvd., Suite 117
San Diego, California 92111
(619) 569-1864 *Call for our Test Equipment Mailer!*

★ SPECIAL ★
4 COLOR PLOTTER

11"x17" Max. Apple III or IBM.
Brand new with manuals, pens, paper driver RS232C ONLY $225

KAYPRO EQUIPMENT

9" Green Monitor	$35.00
Keyboard	75.00
Hard Disk Cable Set (4)	15.00
PRO-8 Mod. to your board	149.00
Host Interface Board	15.00

KAYPRO ICS

81-189 Video Pal	$15.00
81-194 RAM Pal	15.00
81-Series Character Gen. ROMs	10.00
81-Series Monitor ROMs	10.00

CPU & SUPPORT CHIPS

MC68000-8 CPU	$10.00
Z80 CPU	.75
Z80A CPU	1.50
Z80 CTC	1.50
Z80A PIO	2.00
Z80A SIO	5.00
8088	6.50
8089-3	6.50
D8284A	2.50
4164-15	.90
4164-12	1.00
1793	6.00
1797	7.00
ICL7107 LCD Driver	7.00
ICL7140-14 14 Bit A/D	7.50
VC3524 Switching Regulators	5.00
1458 Dual Op-AMP	.70
LM2877P 4W Stereo Amp Dual	2.50
MB81464-15	2.75
2716	3.00
2732	3.25
2764	3.50
27C128-1	9.00
74HC00	.38
74LS125	.30
74LS373	.50
74LS174	.30

HOURS: Mon. - Fri. 9 - 6 — Sat. 10 - 4
MINIMUM ORDER — $15.00
TERMS: VISA, MasterCard, Certified Checks, Money Order, NO COD. Visa and MasterCard add 3%. *Personal checks must clear BEFORE we ship.* Include shipping charges. California residents add 6% Sales Tax. For more information please write (or call).

IBM/PC COMPATIBLES

Mainboard, 8 Slot, Case, Power Supply $225
To make this a complete system, add A) Memory B) Floppy Controller C) Drive D) Keyboard E) Video Card F) Video Monitor G) Multifunction Card

A) MEMORY

256K 150 NS	$19
512K 150 NS	38
640K 150 NS	54

B) FLOPPY DISK CONTROLLER

Card for 2 Floppy Drives	$36
Card for 4 Floppy Drives	42

C) 5¼" FLOPPY DISK DRIVES

Mitsubishi M4853 DSDD 80 Tr	$119
Fujitsu M2551A DSDD 40 Tr	99
Shugart 475 DS Quad 1.2Mb	159

D) KEYBOARDS

Cherry Keyboard (no case)	$38
XT Style Keyboard	47
AT Style Keyboard	69

E) VIDEO CARDS

Tomcat with Parallel and Lightpen Port	$43
Hercules compatible Video Board	75
Color Graphics Adapter	69
Enhanced Graphics Adptr-(EGA)	275

F) VIDEO MONITORS

Roland MB-122G, 12" (no case)	$39
New flat screen Samsungs!	
Samsung SM-12SFG, 12" Grn	96
Samsung SM-13SFA, 12" Ambr	96

G) MULTI FUNCTION CARD
Parallel & Serial Port, Game Port Floppy Controller, Clock & Cal . $96

EGA PACKAGE DEAL

Package consists of Intergraph + 4 EGA Card and the Autoseek 2000 EGA Monitor by Int'l Graphics. No software patches necessary. 1 yr. guar. ...$795

SYSTEM EXAMPLE #1
For the Hacker (Cheap)

Mainboard, Case, Power Supply	$225
256K Memory 150 NS	19
Floppy Controller (2 Drives)	36
Floppy Drive ½ Ht DSDD	99
Keyboard Cherry (no case)	38
Video Board with Parallel and Lightpen Port	43
Roland MB-122G, 12" Green Monitor (no case)	39
	$499

SYSTEM EXAMPLE #2
FCC Approved (Not Cheap)

Mainboard, Case, Power Supply	$225
640K Memory 150 NS	54
Multi I/O, Parallel, Serial, Floppy, Clock/Cal.	96
2 DSDD Floppy Drives (minimum)	198
EGA Package	795
AT Style Keyboard	69
	$1437
Oh, you wanted a turbo board	40
and a 20M Hard Drive & Controller	410

Now how much would you pay? JUST $1887

HARD DISK DRIVES

10M Seagate 212	$200
10M Rodin RO-252, RO352	230
20M Miniscribe	385
20M ST-225	385
20M Tandon TM252	350
20M Tulin (Oki)	345
20M Half Height with Controller	410
32M Half Height with Controller	667
40M Quantum Q540 (Factory Rebuilt)	665
60M with Controller	1150
70M Vertex V170	856

HARD DISK CONTROLLERS

WD-1002-WX2 with Cable	$125
Omni-5510	105
Adaptec 2070A (Get 15M on 10M Drive)	129
Konan KXP230 (Get 15M on 10M Drive)	145
Konan KXP230Z (Get double the density)	164

POWER SUPPLIES

Elgar 400W Unint. Power Sup.	$126.00
5V/1A, −5V/.2A, 12V/1A, −12V/.2A, −24V/.05A	15.00
24V/2.2A	8.00

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/75A, 12V/8A, 24V/5A	55.00

MISCELLANEOUS

Headset/Boom Microphone	$3.95
Z80 Controller Card w/8-bit A to D Converter	15.95
Nicad Pack 12V/.5AH	6.50
Joystick 4 Switches 1" Knob	5.50

TEST EQUIPMENT

OSCILLOSCOPES

Phillips 3260E 120 MHz Dual	$975
TEK 7403N/7A18/7B50A 60 MHz	750

ANALYZERS

Nicolet 500A 1 Hz-100 KHz	$1800

DBASE BOOK OF BUSINESS APPLICATIONS *by Michael J. Clifford*
Reg. $19.95 ***NOW ONLY* $3.95**

(continued from page 44)

and small. Unless I've missed something in my planning, the module in Figure 3 should be the only one specific to the adapter in use.

The constants defined in module Config are imported into the module in Figure 4. This module defines some higher-level constants and types, as well as the headers for a few low-level subroutines. The types exported from this module are transparent - probably not absolutely necessary, but since we'll be diddling bits and bytes, it'll be easier.

I've compiled these modules with both FTL Modula-2 and Logitech's Modula-2/86. Both SHOULD handle the project - we'll see how well as we get deeper into the code.

The procedures GrabClock, FastClock and SlowClock need a bit more explanation. The printer I'm using is rated at 160 CPS, but in reality prints at 120-130 CPS. For an 80-character line, this means we have only 615-670 mS to capture a line of up to 720 pixels, less than 1 mS per pixel.

Modula-2 is fast, but let's be reasonable! GrabClock will take over the real time clock interrupt and re-program the timer chip for a faster rate. When pixel capture is activated by FastClock, we'll get a pixel at each clock tick and store it for later processing (during the print head's return trip). At the appropriate multiple of this faster clock tick, we'll long call the normal real time clock routine. SlowClock will disable the pixel capture portion of the fast routine to let us get the other processing done.

You might want to think about how you would continue the development of the project, and compare it with what I'll have finished by next time.

Editor's note: the code found in this article is also available on the Micro C RBBS, (503) 382-7643, and (with the rest of the code in this issue) on the issue #38 disk. To order the disk, send $6 (if you're a U.S. subscriber) or $8 (non-subscriber or foreign) to Micro Cornucopia, PO Box 223, Bend, OR 97709. Specify MS-DOS or Kaypro 5 1/4" format.

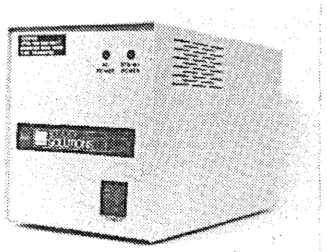
Also, Digi-Key stocks the parts for this project. 1-800-344-4539. Mention the $6.00 scanner.

■□□

Figure 3 - Display Adaptor Def. Module

```
DEFINITION MODULE Config;

(* This module provides the basic constants which define the
graphics screen. Xsize is in pixels, Ysize is in rows, Unused is
in Bytes (the interleaved formats have a few bytes left over in
each interleaved array) and ScrSegment is the segment address of
the screen memory. Use the values for the type of your display
adapter. This definition module is the only one which needs to be
different for the different adapters.*)

(* Depending on the compiler, you may need this EXPORT EXPORT
QUALIFIED Xsize, Ysize, Interleave, Unused, ScrSegment; *)


CONST       (*   HGA CGA  EGA  *)
  Xsize =720;   (*  640  640  *)
  Ysize =348;   (*  200  350  *)
  Interleave =  4;   (*   21  *)
  Unused =   362;   (*  1920  *)
  ScrSegment = 0b000h; (* 0b800h 0a000h *)

END Config.
```

UNINTERRUPTABLE POWER SOURCE



MICRO SOLUTIONS protects your equipment and your data from **power outages** and **brownouts.** Our power systems provide the **fastest switching speed** in the industry (2 ms± 1).

EMI/RFI filtering and surge/spike protection all in one affordable unit. 1 year warranty on all units. Available in a size to suit your needs -

200 watts	$290.00
350 watts	$360.00
550 watts	$410.00
800 watts	$610.00
1000 watts	$710.00

Includes shipping to your door in the continental U.S.. As specialists in overseas systems, we can supply 220 volt units. Call or write for details.

SOFTWARE SPECIAL

MINDREADER $65

"Mindreader's artificial intelligence approach to word processing is the sort of breakthrough we don't normally expect in this business."
"Editor's Choice" PC Magazine, January 1986.

MICRO SOLUTIONS gives you a **no risk** 30 day trial period - call for details.

LOGITECH

Get into **desktop publishing** for under $180.00! **LOGITECH MOUSE** and **PFS:FIRST PUBLISHER** will make your existing PC/XT/AT compatible and dot matrix or laser printer a flexible page layout system.

Limited quantities at an introductory price - Call today!

MICRO SOLUTIONS

P.O. Box 166 Riner, VA 24149
call 24 hours - 7 days a week
1-800-323-4829 (703) 382-6624
Visa MasterCard C.O.D.

We Ship Worldwide
Dealers Supported

Figure 4 - Definition of High Level Constants and Types

```
DEFINITION MODULE ScrnStuff;
  (* This module has the basic screen data definitions and provides the header for some of the lower level subroutines. *)

FROM SYSTEM IMPORT BYTE, ADDRESS;

FROM Config IMPORT Xsize, Ysize, Interleave, Unused;
  (* Depending on the compiler, you may need this EXPORT EXPORT QUALIFIED Raster, Screen, ArrayLen, Lines, ClrScr, GrabClock,
  FastClock, SlowClock, Scan, GraphMode, PixAddress,
  SetBit, ClrBit, InvertBit;*)

CONST
  Lines = Ysize DIV Interleave; (*# of rasters/interleaved array*)
  ArrayLen = (((Xsize DIV 8 * Ysize) DIV Interleave)-1)+Unused;
   (* Full size -1 of interleaved array *)

TYPE
  Raster = ARRAY [0..(Xsize DIV 8)-1] OF BYTE;
  Screen = ARRAY [0..Interleave-1], [0..ArrayLen] OF BYTE;
  BitPos = [0..7]; (* Ordinal value of a bit position in a byte *)
  Xpos = [0..Xsize-1]; (* Allowed range of X pixel position values *)
  Ypos = [0..Ysize-1]; (* Allowed range of Y pixel position values *)

PROCEDURE ClrScr (VAR S:Screen);
  (* Clear the Graphics mode screen, usually just requires filling the memory on the adapter with zeroes *)

PROCEDURE GrabClock (IntNum : BYTE; TickLen : CARDINAL);
  (* The data rate needed to capture pixels is faster than the standard real time clock, and an untimed software loop will be somewhat
  unpredictable. This routine re-programs the DOS real time clock interrupt device to generate interrupts at about 1 mS intervals. When
  enabled, the interrupt service routine will capture a pixel at each clock tick. *)

PROCEDURE FastClock;
  (* Enable pixel capture at fast clock interrupt rate *)

PROCEDURE SlowClock;
  (* Disable pixel capture at fast interrupt rate. *)

PROCEDURE Scan (VAR R : Raster);
  (* Capture one scan line of pixel data, no processing done. *)

PROCEDURE GraphMode;
  (* Put the video adapter in graphics mode. For CGA and EGA we can use BIOS services. For the HGA (since it was never an official IBM
  product) we will have to re-program the hardware directly. *)

PROCEDURE PixAddress (X : Xpos; Y : Ypos; VAR B:BitPos):ADDRESS;
  (* From X and Y pixel coordinates, calculate a physical memory address and bit position within that byte. *)

PROCEDURE SetBit (SrcByte : BYTE; BitNum : BitPos): BYTE;
  (* Set one bit in a byte *)

PROCEDURE ClrBit (SrcByte:BYTE; BitNum:BitPos): BYTE;
  (* Clear one bit in a byte *)

PROCEDURE InvertBit (SrcByte:BYTE; BitNum:BitPos): BYTE;
  (* Toggle one bit in a byte *)

END ScrnStuff.
```

Data Abstraction And Dynamic Allocation

How To Cut And Paste Text Screens In Modula 2

Pasting bits and pieces of screens together must be great fun. Look at all those pull-down menus people are creating. However, they can be very wasteful of memory. Unless, of course, you're a very dynamic programmer.

Large programs which use interrupt handlers, manipulate screens, or need to be optimized for speed are good candidates for a programming language which allows access to a computer's low levels.

We can reach these low levels, in some languages, by calling assembly language subroutines. Unfortunately, these embedded subroutines are expensive to maintain, don't allow data abstraction (the organization of information into objects we can manipulate without knowing the internal structure of the data), and lack portability.

Fortunately, some languages (such as Modula 2 and ADA) are versatile enough to handle the whole job themselves.

Data abstraction and dynamic memory allocation are two areas where these high-level/low-level languages shine. I'll show you how they work by developing a module (in Modula 2) for storing and manipulating sections of a text screen.

Defining A Module

Here's the specification -

- We'll store a rectangular part of a screen in as small an area of memory as possible, and then call it back to any other location on the screen.
- The screen blocks will retain text attributes, such as blinking or reverse video. We'll address them by name.
- All of the details involved in the operations will be hidden (abstracted) so the user can refer to the operations in a very high-level way.

Our first abstraction will be the definition module called Screenblocks. See Figure 1.

The first definition in the module is a block of screen data (which we'll address by name). The details of the block will be hidden (or abstracted). I'll call it "handle" because it will manipulate (or handle) the screen block.

We want to "cut" blocks of screen out (defining them by their row and column positions) and save them for future "pasting." We'll retrieve them (for pasting) by referring to their "handles."

So let's define two screen operations -

- CutBlock
- PasteBlock

Our screen will consist of 25 rows, numbered from 0 to 24, with 0 beginning at the top of the screen. The columns are numbered 0 to 79, beginning at the left. Thus we can define rectangular blocks

Figure 1 - ScreenBlocks Definition Module

```
DEFINITION MODULE ScreenBlocks;
(* This module is system specific. This version is written for
the     IBM-PC and clones using MS-DOS. *)

EXPORT QUALIFIED CutBlock,PasteBlock;

PROCEDURE CutBlock(  FirstRow,LastRow,FirstCol,LastCol : CARDINAL;
                         Handle : ARRAY OF CHAR;
                     VAR done : BOOLEAN);
(* Cuts a block of screen characters and their attributes
   and saves them for later retrieval. *)

PROCEDURE PasteBlock(     Handle : ARRAY OF CHAR ;
                          UpperLeftX,UpperLeftY : CARDINAL;
                          NewPosition : BOOLEAN;
                      VAR done : BOOLEAN);
(* Retrieves & pastes a block in a new position if new position is
   true or replaces it in its old position if new position is
 false.  *)

END ScreenBlocks.
```

Figure 2 - ScreenBlock RECORD Definition

```
TYPE NameArray    = ARRAY[0..24] OF CHAR;
     ScreenBlock = RECORD
                     Handle : NameArray;
                     FirstRow,LastRow,FirstCol,LastCol: CARDINAL;
                     Row : BlockType;
                   END;
```

By Thomas L. Ochs
Structured Scientific Software
1509 Queen Ave. SW
Albany, OR 97321

Data abstraction and dynamic memory allocation are two areas where Modula-2 and Ada shine.

easily, and move them about the screen.

No use wasting memory after we've thrown away a block, so we'll allocate and deallocate memory dynamically. (Since we want our program to be portable, we won't be able to move blocks from display memory, since the locations may be hardware dependent.)

We have a DEFINITION MODULE of procedures, so we can start building the low-level tools we need to implement this set of procedures.

The first of these tools is a data structure for storing a screen block. This structure must know where the block came from, its size, where its characters lie, and their attributes.

In The Low Levels

The most primitive unit in this data structure is the character and its attribute - a pair of bytes.

The Modula 2 module called SYSTEM has a data type BYTE with which we'll set up a two-byte (character and attribute) data structure.

```
TYPE CA = ARRAY[0..1] OF BYTE;
```

(continued next page)

Figure 3 - ScreenBlocks Implementation Module

```
IMPLEMENTATION MODULE ScreenBlocks;
(* This module is system specific. This version is written for
   the IBM-PC and clones using MS-DOS. *)

FROM Storage IMPORT ALLOCATE,DEALLOCATE,Available;
FROM Strings IMPORT Assign,CompareStr;
FROM SYSTEM IMPORT AX,BX,CX,DX,SETREG,GETREG,CODE,
                   SWI,TSIZE,BYTE,WORD;

CONST  rows    =  25;
       cols    =  80;
       NumBlocks = 10;
       PUSHBP  =  0055H;
       POPBP   =  005DH;
       INT10   =  0010H;
       READCH  =  0800H;
       WRITECH =  0900H;
       GETMODE =  0F00H;
       CURSOR  =  0200H;
       ROWINC  =  0100H;
       NAMELENGTH = 24;

TYPE   CA  = ARRAY[0..1] OF BYTE; (* Contains char value and
                                     attribute. *)
       (* CA[0] is the character and CA[1] is the attribute. *)
       R = ARRAY[0..cols - 1] OF CA; (* Each line of the
                                        80 col display. *)
       RowPointer = POINTER TO R;
       BlockType = ARRAY[0..rows - 1] OF RowPointer;
       ScreenBlock = RECORD
                Handle : ARRAY[0..NAMELENGTH] OF CHAR;
                FirstRow,LastRow,FirstCol,LastCol : CARDINAL;
                Row : BlockType;
                    END;
       BlockPointer = POINTER TO ScreenBlock;
       BlockArray = ARRAY[0..NumBlocks - 1] OF BlockPointer;

VAR BlockSpace : BlockArray;

PROCEDURE CutBlock(FirstRow,LastRow,FirstCol,LastCol : CARDINAL;
                   Handle : ARRAY OF CHAR ; VAR done : BOOLEAN);

VAR I,J,K,NumCols,Position : CARDINAL;
    A : BlockPointer;
    MODE,PAGE,TEMP : WORD;

BEGIN
    done := FALSE;
(* Test for legitimate input. *)
    IF (((FirstRow <= LastRow) AND (FirstCol <= LastCol))
    AND ((LastRow < rows) AND (LastCol < cols))) THEN
```

(continued from page 49)

Then we can define a row (R) of characters as an array of CA's (characters and attributes):

TYPE R = ARRAY[0..79] OF CA;

We could make this into a two-dimensional array (rows and columns), but it would be wasteful, since memory is automatically allocated for the entire structure when the variables are declared. Let's use only as much memory as we need.

An entire screen requires (25*80*2) = 4000 bytes, so let's define a pointer to type R, and an array of pointers to represent the number of rows:

```
TYPE RowPointer = POINTER TO R;
     BlockType = ARRAY[0..24]
                 OF RowPointer;
```

"BlockType" will hold up to a full screen of information, but will consume only 50 bytes (as 25 pointers).

Next, we define a record of ScreenBlock, including its handle and location (rows and columns). See Figure 2. When empty, ScreenBlock will require 83 bytes.

In order to keep track of all the ScreenBlocks, we define a pointer to the ScreenBlock type -

TYPE BlockPointer = POINTER TO ScreenBlock;

Then, we create an array of these pointers for storing the various screen blocks that may be "cut" out:

```
TYPE BlockArray = ARRAY
   [0..NumBlocks - 1] OF BlockPointer;
```

This is a fast data structure, since we're using array indices (instead of searching through long lists of pointers) to locate elements.

The savings in memory, however, might not be immediately apparent, so let's "cut" out a block of screen text from row 11 to row 15 and column 21 to column 50. This block will contain 150 characters out of the screen's 2000.

Dynamic Allocation

In Modula 2 there are two ways to reserve memory for dynamic structures -

- The NEW command: which is similar to Pascal's
- The ALLOCATE command

```
(* Calculate the number of rows and the number of columns. *)
      NumCols := LastCol - FirstCol + 1;

(* Now allocate the minimum space for the screen block. *)
      IF Available(TSIZE(ScreenBlock)) THEN
        NEW(A);
  (* Initialize the screen block. *)
        A^.FirstRow := FirstRow;
        A^.LastRow := LastRow;
        A^.FirstCol := FirstCol;
        A^.LastCol := LastCol;
        Assign(Handle,A^.Handle);
        FOR I := 0 TO (rows - 1) DO A^.Row[I] := NIL; END;

  (* Calculate the needed space. *)
        J := TSIZE(CA) * NumCols;

  (* Now allocate the needed space. *)
        WITH A^ DO
          FOR I := FirstRow TO LastRow DO
            IF Available(J) THEN
              ALLOCATE(Row[I],J);
            ELSE
              FOR K := I TO FirstRow BY -1 DO
                DEALLOCATE(Row[K],J);
              END;
              DISPOSE(A);
              RETURN;
            END;  (* FOR K *)
          END;  (* For I *)
        END; (* With *)

  (* Now read the screen blocks *)
        CODE(PUSHBP);   (* Save the Base Pointer. *)
  (* First find the currently displayed page and mode. *)
        SETREG(AX,GETMODE);
        SWI(INT10);
        GETREG(AX,MODE);
        GETREG(BX,PAGE);

  (* Now read each location. *)
        FOR I := FirstRow TO LastRow DO
          Position := (I * ROWINC) + FirstCol;
          FOR J := 0 TO NumCols - 1 DO
  (* First the cursor must be positioned. *)
            SETREG(AX,CURSOR);
            SETREG(BX,PAGE);
            SETREG(DX,Position);
            SWI(INT10);

  (* Now the character must be read. *)
            SETREG(AX,READCH);
            SETREG(BX,PAGE);
            SWI(INT10);
            GETREG(AX,TEMP);
(**** Warning the next statement is word size sensitive. *****)
            A^.Row[I]^[J] := CA(TEMP);
            INC(Position);
          END;
        END;
        CODE(POPBP);
  (* Now try to store the block *)
        I := 0;
  (* Find an open storage space. *)
        WHILE ((I < NumBlocks) AND (BlockSpace[I] # NIL)) DO
          INC(I);
        END;
  (* If one was open then store the block. *)
        IF I < NumBlocks THEN BlockSpace[I] := A; done :=
TRUE; END;
      END;  (* IF *)
    ELSE
      done := FALSE;
    END;
END CutBlock;
```

```
PROCEDURE FindBlock(Handle : ARRAY OF CHAR;
                    VAR INDEX : CARDINAL;
                    VAR A : BlockPointer;
                    VAR found : BOOLEAN);
BEGIN
  found := FALSE;
  INDEX := 0;
  WHILE INDEX < NumBlocks DO
    IF BlockSpace[INDEX] # NIL THEN
      IF CompareStr(BlockSpace[INDEX]^.Handle,Handle) = 0 THEN
        found := TRUE;
        A := BlockSpace[INDEX];
        RETURN;
      END;  (* IF CompareStr *)
    END;  (* If BlockSpace *)
    INDEX := INDEX + 1;
  END;  (* WHILE *)
END FindBlock;

PROCEDURE PasteBlock(    Handle : ARRAY OF CHAR;
                         UpperLeftX,UpperLeftY : CARDINAL;
                         NewPosition : BOOLEAN;
                     VAR done : BOOLEAN);
(* This can either paste the block in a new position if new
  position is true or replace it in its old position if new
  position is false. *)

VAR I,J,K,NumRows,NumCols,Position,CH,PC,
    FirstCol,LastCol,FirstRow,LastRow : CARDINAL;
    A : BlockPointer;
    MODE,PAGE : WORD;
    found,checked : BOOLEAN;
    chr : CHAR;
    MASK,TEMP : BITSET;

BEGIN
(* Find the Handle *)
  done := FALSE;
  found := FALSE;
  checked := FALSE;
  MASK := {15,14,13,12,11,10,9,8};
  FindBlock(Handle,I,A,found);
  IF found THEN
(* Calculate the number of rows and the number of columns. *)
    NumRows := A^.LastRow - A^.FirstRow + 1;
    NumCols := A^.LastCol - A^.FirstCol + 1;
    IF NewPosition THEN
(* Check to see if the new position will fit *)
      IF (((UpperLeftX + NumCols) < cols) AND
          ((UpperLeftY + NumRows) < rows)) THEN
          FirstCol := UpperLeftX;
          FirstRow := UpperLeftY;
          LastCol  := UpperLeftX + NumCols -1;
          LastRow  := UpperLeftY + NumRows -1;
          checked  := TRUE;
      END;
    ELSE
      FirstRow := A^.FirstRow;
      LastRow := A^.LastRow;
      FirstCol := A^.FirstCol;
      LastCol := A^.LastCol;
      checked := TRUE;
    END;
    IF checked THEN
      CODE(PUSHBP);    (* Save the Base Pointer. *)
(* First find the currently displayed page and mode. *)
      SETREG(AX,GETMODE);
      SWI(INT10);
      GETREG(AX,MODE);
      GETREG(BX,PAGE);
(* Now clear out the low byte in page. *)
      TEMP := BITSET(PAGE)*MASK;
      PAGE := WORD(TEMP);
 (* Now write each location. *)
      FOR I := FirstRow TO LastRow DO
```

We can minimize storage consumption by using the BlockPointer type and the ALLOCATE command to reserve memory on the heap instead of the NEW command.

NEW(A) reserves memory for the pointer A according to the type it points to. A RowPointer has 160 bytes reserved.

ALLOCATE lets us decide how much memory to reserve for a variable, regardless of type. So, ALLOCATE(A,N) reserves (or allocates) N bytes and returns the address of this allocated memory in the ADDRESS variable A.

The ADDRESS type is defined as a pointer to WORD, and is assignment compatible with other pointers. So we can use it to dynamically allocate part of the declared memory for a variable. If we declare:

```
VAR A,B,C : RowPointer;
ALLOCATE(A,10);
```

I hope it's clear that ALLOCATE saves a lot of memory while retaining the ease of array indexing.

only 10 bytes are reserved for the R type pointed to by A.

ALLOCATE(B,10); then allocates the next ten free bytes for the R type pointed to by B, and ALLOCATE(C,10) reserves the next ten bytes for C.

I hope it's clear that ALLOCATE saves a lot of memory while retaining the ease of array indexing.

If these variables (A, B, and C) represent three partial rows of five characters (and attributes) per row, then we've used only the 30 bytes we need instead of the 480 that would be reserved using the NEW command.

Details

We can still access characters by referring to their relative position in the

(continued next page)

```
        Position := (I * ROWINC) + FirstCol;
        FOR J := 0 TO NumCols - 1 DO
    (* First the cursor must be positioned. *)
          SETREG(AX,CURSOR);
          SETREG(BX,PAGE);
          SETREG(DX,Position);
          SWI(INT10);
    (* Now write a character. *)
          chr := CHAR(A^.Row[I]^[J][0]);
          CH := WRITECH + ORD(chr);
          chr := CHAR(A^.Row[I]^[J][1]);
          PC := CARDINAL(PAGE) + ORD(chr);
          SETREG(AX,CH);
          SETREG(BX,PC);
          SETREG(CX,1);  (* Number of char to repeat. *)
          SWI(INT10);
          INC(Position);
        END; (* FOR J *)
      END;   (* FOR I *)
      CODE(POPBP);
      done := TRUE;
    END; (* IF checked. *)
  END; (* IF found. *)
END PasteBlock;

END ScreenBlocks.
```

Figure 4 - MODULE Which Swaps Screen Blocks

```
MODULE SwapBlocks;
(* Swaps two blocks of the screen. *)
IMPORT Break;
FROM InOut IMPORT WriteString,WriteCard,WriteLn;
FROM ScreenBlocks IMPORT CutBlock,PasteBlock;

VAR done,NewPosition : BOOLEAN;
    I,J,K : CARDINAL;

BEGIN
  WriteLn;
  FOR K := 0 TO 11 DO
    FOR I := 0 TO 79 DO
      WriteCard(1,1);
    END;
  END;
  FOR K := 12 TO 24 DO
    FOR I := 0 TO 79 DO
      WriteCard(2,1);
    END;
  END;
  NewPosition := TRUE;
  CutBlock(5,10,22,42,'First',done);
  IF NOT done THEN
    WriteString('First block not cut.');
  END;
  CutBlock(15,20,10,30,'Second',done);
  IF NOT done THEN
    WriteString('Second block not cut.');
  END;
  PasteBlock('Second',5,7,NewPosition,done);
  IF NOT done THEN
    WriteString('Second block not pasted.');
  END;
  PasteBlock('First',15,10,NewPosition,done);
  IF NOT done THEN
    WriteString('First block not pasted.');
  END;
END SwapBlocks.
```

(continued from page 51)

array, but we need to be careful. A[6], for example, contains the same information as B[1], and A[12] contains the same information as C[2], even though we only allocated enough memory for A[0..4]. So we'll need some way to check array boundaries.

Also, if we use the assignment statement with an allocated data structure, we might have problems.

For example, if we say C^ := A^ (assigning the de-referenced values pointed to by A to the locations pointed to by C), even though only room for five of the index values has been reserved, all 80 locations will be assigned. This can lead to disastrous results since we can't be sure what's being written over.

We can now access the resulting data structure by A^.Row[I]^[J], where A points to the ScreenBlock, and Row[I]^[J] refers to the Ith RowPointer's Jth element.

In order to efficiently manipulate the blocks of screen images, the system uses the row pointer index to indicate the rows that are populated. The element index is started from zero, the relative column index in the block.

In order to remain machine independent, the primitive TSIZE is imported from SYSTEM to determine the storage size for our data types. If the size of the stored types changes, the system will automatically increase or decrease memory allocation.

Wrap Up

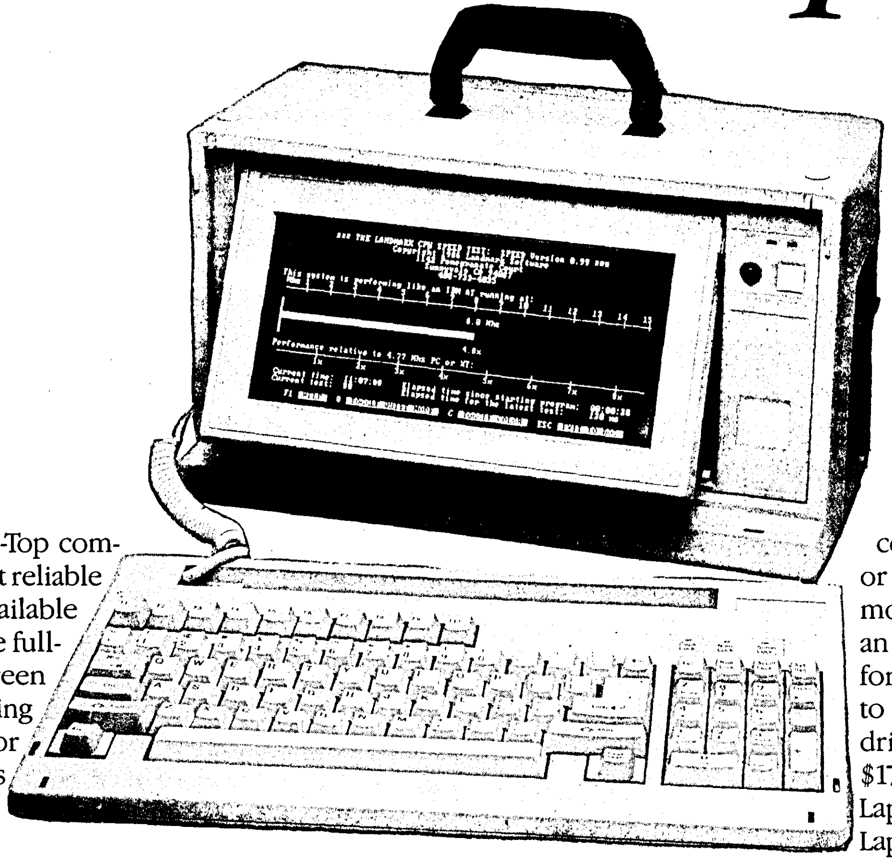
Now that we've described a method for dynamic allocation, we need to read the screen characters and their attributes, and write them back to the screen.

Hello BIOS software interrupt 10H (and Goodbye Mary Lou?). We can use the commands CODE, SETREG, and GETREG (in KModule SYSTEM) to include machine instructions, and the command SWI to generate software interrupts. See the implementation of both CutBlock and PasteBlock (See Figure 3).

The resulting module, "SwapBlocks," copies two areas of the screen and moves them to new positions. (See Figure 4).

■■■

The Ultimate Lap-Top



The McTek286B Lap-Top combines the fastest, most reliable AT motherboard available with the most visible full-size LCD lap-top screen on the market. Running at a switchable 8 or 10 MHz, it includes a 20MB hard disk, 3½″ floppy drive, parallel & serial ports, Award 3.01 bios, 640k, turbo indicator LCD & mouse interface. The screen is a fantastically readable, electroluminescently backlit, 80-column by 25-line LCD with adjustable intensity and screen-angle. It's as readable as a CRT. You can also plug in a digital or analog color monitor or a digital or composite monochrome monitor. Included also is an external 5¼″ floppy port for reading and converting to 3½″ disks (5¼″ external drive w/case, power supply: $179 when purchased with Lap-Top). The McTek 286B Lap-Top comes fully assembled with our one-year parts & labor guarantee, and sells for an amazing, complete price of only ***$1999!***

3 MB On-Board AT!

Our McTek 286A is the most integrated AT-compatible to date. It utilizes the highly regarded Chips & Technology chip set, and includes memory upgradable *on board* to 3 megabytes. No more worries about speed compatibility with expanded memory cards! The 8/10MHz, Ø-wait state McTek 286A runs at 11.5 Norton SI, and an effective 13.2MHz on the Landmark test. Serial, parallel & game ports are all standard on board. With Award 3.01 bios, 640k, 200W power supply, Samsung amber monitor with Hercules-compatible controller, locking case, AT-style keyboard, 1.2MB drive, 20MB Seagate. Assembled & fully tested, with a full one-year warranty. Get in on the most advanced AT-compatible on the market, at the lowest price ever offered! ***$1459!!***

XT Turbos & Supers

640k 4.77/8MHz and 4.77/10 switchable XT turboboards; two 360k floppy-disk drives with controller; one parallel, one serial and one game port; AT-style keyboard; clock, FCC-approved slide-case; eight slots; Hercules-compatible graphics card; amber monitor w/base; fully assembled and tested; one-year parts *and* labor warranty.

$649 **XT Turbo 4.77/8MHz Complete**

Superturbo 4.77/10MHz Complete **$699**

McTek Systems, Inc. • 2316 4TH Street • Berkeley, CA 94710 • 415-843-0714

McTek

DISK DRIVES	
Fujitsu 360k	$ 75
Fujitsu 1.2MB	109
Teac 360k	85
Teac 1.2MB	119
Toshiba 3½″ 720k	135
Floppy controller	25
20MB Hard Disk Kit	299
30MB Hard Disk Kit	339
ST-225	249
ST-238	279
ST-4038	539
ST-4096	899

PRINTERS	
Citizen CD120	$179
Citizen MSP10	265
Citizen MSP20	359
Citizen CTZ-224	699
Toshiba 321XL	559
Call for prices of other brands	

MODEMS	
Easydata int. 300/1200	$ 95
Taiheho external 3/12	105
Everex 2400 external	195

MONITORS	
Samsung amber	$ 85
TVM EGA color	399
TVM RGB color	295
NEC Multisync	559
Sony Multiscan	650
HGC-compat. mono card	55
Color graphic card	49
EGA color card	159

MOUSE	
Logimouse C7	$ 75

PC/XT	
640k TurboMothrbrd	$ 85
10MHz TurboMothrbrd	99
Multi I/O w/disk contrlr.	62
640K RAM card	39
2MB Expansion card	115
RS232 2-port card	35
4-serial port card	95
Game I/O card	15
384k Multifunction card	69
FCC-app. slide XT case	28
150W power supply	59
XT keyboard	52

PC/AT	
McTek286 6/10 1-wait	$334
Baby McTek 286B-AT 8/10 Ø-wait	369
McTek 286A Ø-wait 3MB, 4 ports on board	459
3MB Multifunction card	145
2MB Expansion card	125
Multi I/O card	65
Locking slide case	65
200W power supply	85
AT-style keyboard	59
WD HD/floppy controller	159

MISC.	
Kingtech Portable Computer Kits: XT/AT (power supply, case, keyboard, monitor)	$380/410
Eprom burner 4-socket	139
Multicenter	45
AC power center	25
AC power strips	10
Diskette file box	9
Printer or serial cable	8
Ext. case & powr supply w/Fujitsu 5¼ drive	120
	199

Review By David Thompson

The Konan Hard Drive Card

40 Megabytes Of Data On A 20 Meg Drive

This review comes to you by popular demand. I mentioned the Konan drive controller two issues ago, and since then I've heard from a number of you who are considering purchasing the card.

When I originally heard about the Konan drive controller, it seemed at first too good to be true. Then I wondered why no one else was doing it. Finally I remembered that this sounded like one of the projects that George Morrow was thinking about before his latest company went bust. Ah well, with this auspicious beginning, let's get on with it.

Essentially the Konan card is a fancy multi-function hard drive controller and resident driver which:

- Does disk caching in RAM.
- Packs data to optimize disk I/O, no fragmentation.
- Squeezes data going to the disk. There are three levels of compression, high speed (no squeezing), normal speed (data file compression only), and archive (squeezing on all files, with heavier compression on data files). If you have lots of data, this card will let you store the better part of 40 megs on your 20 meg drive.
- Generates an error correction code which can reconstruct a totally bad sector.
- Moves data off bad sectors and locks them out. (The process is not supposed to be visible to the user.)

Well, with all these features and a price that's not too much more than the standard controller it replaces ($169 for the super deluxe model), it would be hard to resist.

However, there are some tradeoffs. Buffering FATs and directories in RAM plus a 64K file buffer, for instance, left me with only 387K usable on a 640K system. I had to remove all the optional buffers to get the program space back to 541K. (I was using MS-DOS 3.10.)

The Konan card lets you initialize the disk four ways:

(1) As a standard drive. The first logical part of the drive had to be set up this way so the system could boot MS-DOS.

(2) As a high-speed drive. This gives you the error correction and data organization so there aren't empty clusters scattered about, plus RAM buffering if it's used. This mode appeared to be a bit slower than (1).

(3) Normal data compression. You get all the features of (2) plus a compression scheme that's supposed to reduce the size of data files by about one-third. This mode appeared to be about as fast as (2).

(4) Archival data compression. You get all the features of (3) plus more substantial compression on data files and a bit of compression on object files. This mode was noticeably slower than any other.

Test Configuration

First, I went over to MicroSphere and borrowed a drive. I divided up the MiniScribe 20 meg 65 ms unit into four logical drives. Logical drive C had to be normal so it was bootable. Konan recommended that the boot drive be 1 meg so I gave it the requisite 30 tracks.

I divided up the remainder of the disk into three equal parts. Drives D, E, and F each got 194 tracks. (See Figure 1.)

I set up drive D as a high-speed drive

Figure 1 - Timing The Konan Card

```
  drive c           drive  d          drive  e          drive  f

  16.41 sec         18.72 sec         19.04 sec         23.77 sec
CRC 184K of .com files
(standard card, Lapine 65ms drive, nothing in memory, took 16.75 sec.)
-----------------|-----------------|-----------------|-------------------
space consumed:

  356,352          303,116            311,296           229,376

copied 309,248 bytes of object code from a floppy to the hard drives
-----------------|-----------------|-----------------|-------------------

   24.34 sec        28.40             28.49              33.79

CRC time on 309,248 byte floppy. The floppy took 133.29 seconds
-----------------|-----------------|-----------------|-------------------
total bytes         4,954,904         4,954,904         4,954,904
space used          4,988,928         3,219,456         2,121,728
These were dBASE data files
-----------------|-----------------|-----------------|-------------------
total bytes         6,825,060 (full)  8,298,437         8,298,437
space used          6,864,896         6,144,000         4,587,520
mixed data and program files
-----------------|-----------------|-----------------|-------------------
first time 7.33   (couldn't load)     8.11 sec          12.79 sec
subsequent 7.33   (couldn't load)     7.36 sec          11.79 sec
time to load 130K program using 128K buffer space
-----------------|-----------------|-----------------|-------------------

(Note: Konan card was run in PC-Tech 80186 board.)
```

SCIENTIFIC SOFTWARE

SCI-GRAF *$99.95*	Create huge hi-res plots with log or linear scaling. Screen and printer output. Automatic legends and labels. Flexible ASCII input. Works with CGA, EGA, Hercules, and mono cards.
SCI-GRAF MODULES *$250.00*	Create custom hi-res graphs from within your own programs by linking to our object code. Supports all *SCI-GRAF* features plus plotter output, and more! Microsoft C, Turbo C, and Aztec C versions. No royalties.
FONTEDIT *$49.95*	Create custom Greek, math, or other symbols for use with *SCI-GRAF* or *SCI-GRAF MODULES.* Requires IBM compatibility and CGA or EGA.
SCI-DATA *$59.95*	Perform least squares fits (linear, parabolic, and exponential) and normal curve approximations. Convert between polar and rectangular coordinates. Also supports a variety of scaling transformations. Great complement to *SCI-GRAF!*
SCI-EVAL *$49.95*	Pop-up scientific expression evaluator, more powerful than other pop-up calculators. Complete expression editing facility. Full range of functions: scientific, statistical, logic. Requires IBM compatibility.

Free shipping on prepaid orders. No credit cards.

MSC Microcomputer Systems Consultants

P.O. Box 747, Santa Barbara, CA 93102
(805) 963-3412

ALL SALES SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. FREE COPY UPON REQUEST.

CALL FOR FREE CATALOG

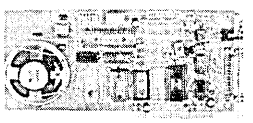
TEXT TO SPEECH BOARD!

PC/XT COMPATIBLE. MAKE YOUR COMPUTER TALK!

A VERY POWERFUL AND AMAZING SPEECH CARD. USES THE NEW GENERAL INSTRUMENTS SPO256-AL2 SPEECH CHIP AND THE CTS256A-AL2 TEXT TO SPEECH CONVERTER.

THIS BOARD USES ONE SLOT ON THE MOTHERBOARD AND REQUIRES A COM SERIAL PORT. BOARD MAY ALSO BE USED IN A STAND ALONE ENVIRONMENT WITH ALMOST ANY COMPUTER THAT HAS A RS232 SERIAL PORT. FEATURES ON BOARD AUDIO AMP OR MAY BE USED WITH EXTERNAL AMPS.

DEMONSTRATION SOFTWARE AND A LIBRARY BUILDING PROGRAM ARE INCLUDED ON A 5¼ INCH PC/XT DISKETTE. FULL DOCUMENTATION AND SCHEMATICS ARE ALSO INCLUDED.

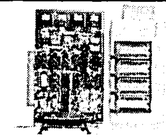


NEW! PRICE CUT! $69.95 ASSEMBLED & TESTED

CANON 80 COLUMN PRINTER - $39.95

ORIGINALLY MANUFACTURED FOR THE PC JR. BUT WITH OPTIONAL CONNECTOR WILL WORK WITH PC, XT, OR AT. REQUIRES SERIAL I/O. THIS THERMAL PRINTER IS QUIET AND USES EASY TO GET 8½ IN. ROLLS OF PAPER. 50 C.P.S., UPPER AND LOWER CASE, PLUS GRAPHICS. ORIGINAL LIST PRICE $199.00. ADD $3.00 FOR PC/XT CONNECTOR. ADD $5.00 UPS.

PC/XT EPROM PROGRAMMER $169



NEW!

★ LATEST DESIGN ★ PROGRAMS UP TO 4 DEVICES AT ONE TIME ★ FEATURES EASY TO USE MENU DRIVEN SOFTWARE THAT RUNS UNDER PC OR MS-DOS. ★ USES AN INTELLIGENT PROGRAMMING ALGORITHM FOR SUPER FAST (8X) EPROM BURNING. ★ THIS PLUG-IN BOARD ATTACHES TO AN EXTERNAL MINI CHASSIS CONTAINING 4 TEXTOOL Z.I.F. SOCKETS. ★ NO PERSONALITY MODULES REQUIRED ★ AUTOMATIC VPP SELECTION: 12.5V, 21V, OR 25V. ★ EPROM DATA CAN ALSO BE LOADED FROM OR SAVED TO A DISKETTE. ★ PROGRAMMING SOFTWARE SUPPORTS: 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 27256A, 27512, AND 27512A. ★ ASSEMBLED AND TESTED, BURNED. IN WITH MANUAL. $169 WITH SOFTWARE.

JUST RECEIVED SAME AS ABOVE PROGRAMMER, BUT PROGRAMS 10 UNITS AT ONE TIME - $299.

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

TERMS: Add $3.00 postage. We pay balance. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCard. Texas Res. add 6-1/4% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance.

(no compression). Drive E got normal compression. Drive F got archival (super-duper) compression. I set them each up with the same number of tracks so I could easily compare how much space was saved by the compression.

Bugs

I couldn't run programs from drive D (the fast, unsqueezed portion of the hard drive) if they approached about 40K. Also, I couldn't copy any files of that size (or larger) from drive D to any other drive. If I tried either, I'd get the message: "General Drive Error..." Smaller programs and files worked without a whimper.

A file-by-file test using Norton's DISKTEST reported that the files were bad. However when I asked DISKTEST to do a track-by-track search for errors, it would find none. Also, CRC (cyclic redundancy check) would read every byte of these files and generate the correct CRC without complaining.

Wow!

So I reformatted the MiniScribe, this time as three logical drives. Drive C was still the normal, MS-DOS boot. I made drives D and E both fast (no compression, but data correction, buffering, etc.). Maybe the software didn't like to shift from compression to non-compression.

Unfortunately still no go.

However, the speedy version wasn't any faster than the logical C drive (or any faster than a standard controller), so there might not be much need for it.

Normal and archival compression did well. I left the system running a .BAT file for half a day just copying back and forth between logical drives E (normal compression) and F (archival). There wasn't an error.

Normal compression increased file access time just slightly. Archival storage increased file access time by about 50%. But copying large quantities of data (overwhelming the RAM buffer) was slow. Copying 6.8 meg from F to D took 11 minutes 14 seconds. Copying 6.8 meg from one directory to another on the same system using a standard controller (and very full hard drive) took 6 minutes 10 seconds.

ROM Versions

I purchased the Konan Board about five months ago. After I got it, I called the company and mentioned that I would be reviewing it. Since that conversation they've sent me two new ROMs. The first was to substantially speed up compression and decompression. They didn't say why they sent the second. I used the latest ROM for these tests.

Conclusion

I'm glad I bought the Konan card. Because of the effective compression and the error correction, it should be a great way to archive lots of data. I had no problems with data stored using either of the compression modes.

Unless you have RAM to burn, however, I wouldn't let the Konan software buffer anything (memory use is pretty thoroughly covered in their documentation). And, I'd avoid the fast mode, at least until I'd checked it out on my system. (I used my PC-Tech X-16 for these tests. The board wouldn't work at all with our Challenger 186 boards.)

I haven't had enough experience with the card to know whether there will be long-term problems or if it is system or drive sensitive. ■■■

C'ING CLEARLY

Turbo C

Ron Miller
1157 Ellison Dr.
Pensacola, FL 32503

Everything you wanted to know about Borland's Turbo C - the compiler, editor, library, debugging environment... isn't here. It's not all in the C manual either. Between here and the manual, however, you should get pretty close. Another great C piece from Ron.

Like over 100,000 other hobbyists, professionals, and dreamers, I finally received my copy of Borland's new Turbo C. That's 100,000!!! - one times ten to the fifth. The very thought of 100,000 clones out there somewhere chock full of uninitialized pointers, unterminating loops, and unchecked type conversions makes one wonder whether it really was a good thing for Philippe Kahn to bluff his ads into those computer journals a few years ago.

Will C become the BASIC of the late 80's? Golly, I hope so. I think. You spouses of computer junkies may find this to be the last blow to what remains of the American marriage. Now you'll never get him/her to turn off the computer and come to bed.

The real question is whether Borland deserves to take over the C market the way it deservedly took over, and redefined, the Pascal market. After about a month-and-a-half of working with Turbo C some six hours a day, I'd give it a qualified yes. Version 2 or 3 ought to remove my qualifications.

The Delights

First of all, as a VALUE, Turbo C is even more of a breathtaking bargain than Turbo Pascal. It's the Borland Revolution at the very edge of sanity.

For $64 from the Programmer's Connection (they even paid UPS), I received four diskettes stuffed with extraordinarily sophisticated software - a full-fledged stand-alone compiler and an integrated package. The package came equipped with a beautiful programming editor blessed with almost all the goodies I dreamed of when using the Pascal editor.

Turbo C came with support for six memory models, all the latest ANSI extras, plus delightful extensions which I'll discuss shortly. Huge libraries. A zippy linker. A stand-alone make utility with a wonderfully elegant syntax. Two books that by themselves would sell for $49.90 at B. Dalton, one of them containing as useful a short introduction to C as can be had anywhere. Enough include files to clog the White House paper shredder. I felt like a software pirate just copying my very own disks onto my very own Seagate.

In conception, the integrated package is a programmer's dream. Since C is inherently messier than Pascal, things aren't as idiot-proof as in Pascal, which does a miraculous job of hiding the mechanics of compiling from the user.

Though the integrated package does everything in a single sweep of linked stages, even the beginner must deal with libraries, linkages, memory models, and the like. Borland puts all your options into SideKick-style pop-up windows.

What is gained from the extra machinery, of course, is the ability to build and control multi-file projects - but beginners are going to be a bit intimidated. The quick and dirty piece of code will perhaps never be quite as quick and dirty in C as in Pascal. It will, however, be smaller, thanks to linkers that don't load functions if your programs don't call for them.

Even if the rest of the package were only so-so, the error-flagging utility of the integrated environment would be worth the cost of the package. I confess I've written lots of applications software in Pascal, cursing ord's and chr's and succ's all the way. I do it because the Turbo environment suits my style of programming.

I sit down in front of my Zenith, coffee mug in hand, shoes across the room, reference manuals spread wildly across my desk and the adjacent floor - and write code - 1500, 2000 lines of it sometimes. Then I see if it compiles.

Turbo Pascal lets me walk through the errors one at a time, correcting and recompiling and correcting and recompiling until things fall all the way through. In Turbo there is no leaving the editor, trying a compile, scrawling the line numbers on the back of an envelope, reentering the editor, etc., etc., etc. Just bouncing back and forth between compiling and my subsequent stupidity.

Well, Turbo C does it even better. You can set the number of errors you want to stop at. (The default is 25, but that's silly. After 5 good errors an honest compiler is so helplessly confused that the following error messages are worse than useless.)

After the poor compiler reaches the error limit, a "message" window appears containing a list of the errors and warnings. You can go to any one of them by moving your cursor to an error listing

As a value, Turbo C is even more of a breathtaking bargain than Turbo Pascal. It's the Borland Revolution at the very edge of sanity.

and whacking "enter." Pop! There's the cursor, at the offending point in the code.

If the project involves several files, the editor will call up the appropriate file automatically. Or else you can use F8 and F7 to move one error forward or backward in the list. An utter joy.

My productivity in C has surely doubled because I can charge forward and trust the compiler to flag the silly errors, misspellings, undeclared variables, and so forth. This process is aided by Borland's adoption of function prototypes to check function arguments.

Use their include files and put prototypes of your own functions at the head of your files, and you will get a level of insurance that approaches the safety of Pascal. Or, if you don't want to be nagged, then revert to what Borland calls "classic" (i.e., K&R) C and you are free to walk the rope without the net.

Sometimes I get tired of Turbo's ditherings about my fast and loose games with pointers and wish it would just shut up. As far as I'm concerned, "suspicious pointer..." is a redundancy. All pointers are suspicious. That's why they're fun.

On the other hand, error catching in Turbo is the best I've ever seen. They are spot-on with structure syntax, where lots of compilers seem to get overly indulgent. Incidentally, Borland allows multiple structures with identical field names - something I've always loved about Turbo Pascal and hated K&R C for not permitting.

Gripes

So what could I possibly complain about?

Well, the package seems a bit thrown together - rushed in a way that Turbo Pascal never was, even from the very beginning. The documentation, slick though it seems to somebody accustomed to C compilers shipped in Zip-Lock bags, is full of little errors and omissions that confuse and erode confidence in the package.

The keyboard interrupt is 16 hex, not 14, as the "bioskey" documentation assures us. The functions _creat() and creat() are confused in the function descriptions. The interrupt function documentation is darned near incomprehensible. The discussions of open() and _open() are so jumbled together that it took me six or eight rereadings to realize that the raw MS-DOS low-level _open() is NOT a direct translation of DOS function 3Dh, because it insists upon the Unix-based O_RDONLY/O_WRONLY/O_RDWR symbolic constants squirreled away in some arbitrary header file.

Any assembly language practitioner who out of habit uses a "2" for read and write updating will get no warning, just a mess that's perverse enough to work MOST of the time.

A good sign of the slapdash quality of the whole is the fact that the documentation preaches on and on (justifiably, I think) about the advantages of using "typedef" in declaring variables, while the headers given by the program itself all use "struct XXX" declarations and not typedefs. Mr. Kent Dolan writes to tell me that the search key functions are poorly documented and nonstandard. Though I haven't played with those functions, I'm not surprised.

Although the memory management descriptions look very nice, diagrams and all, I still haven't found a precise description of how programs handle memory accounting with MS-DOS. What functions - and in which models - merely dole out memory already owned by the program, and which actually make fresh calls to DOS to get new memory? And what about the various fetch-a-character-from-the-console functions? Which MS-DOS calls are used? Writers of resident programs and folks who hate ANSI.SYS need to know.

Yeah, I know I could spend almost $300 to get a copy of their source code. But that sort of nuts-and-bolts information should come with the package. Nor have I discovered yet how much stack space the various models give me to play with (it seems to be 0x1000 bytes for the large model) and whether programmers can change that. Many less elegant compilers are much more explicit about the actual ways their programs work.

Although Borland will let you set a switch to detect stack overflow, it apparently won't let you decide how much auto stack you wish to set aside. It's as if Borland was still thinking about relatively passive Pascal and BASIC programmers rather than the C hacker who jolly well wants to know and control what's going on.

As a matter of fact, that's what I find hardest to take about the Borland package. I'm used to compiler documentation that treats me like an equal, that lets me know the choices and where to patch if necessary. Borland sees itself as Big Brother who thinks that the masses ought to treat compilers as black boxes. The chapter on Advanced Programming begins, "We knew you'd get around to this chapter sooner or later." Notice the condescension in that turn of phrase. It's not unrepresentative.

And then there are twists that can only be termed undocumented mysteries. Take for example the matter of the system() call. K&R prescribes - and so Turbo naturally delivers - a function that will execute a string (blanks and all) just as if it had been typed on the command line.

In simple situations Turbo's system() works fine, though it adds over 5300 bytes to a file (large memory model). Apparently it's that massive because that one call is entangled with a long series of exec..() and spawn..() functions carried over from UNIX. That was okay - until I tried to call system() from within a resident program. Mysterious refusals to execute.

Naturally, I suspected myself first; but after hours of fruitless staring at the

(continued next page)

(continued from page 57)

code, I reminded myself that MS-DOS 4Bh calls *can* be made from residence. DOS just doles memory out from the unallocated memory at the top. It works in other Cs, why not in Turbo? So I started from scratch with an assembly language module that used the MS-DOS function directly. Turbo's inline coding made it a snap. It worked, right from the first. And my own version added 454 rather than 5344 bytes to the EXE file.

So what is going on? In almost 5000 extra bytes, a lot is possible. Some sort of error checking, maybe. All I know is that I immediately added my own exec() function to my private linking library. My function may not search the path automatically and may not remember my birthday and may not concatenate long lists of argument strings, but it is smaller and the error codes make sense.

I also discovered that stream I/O apparently doesn't work from residence, though I'm a skilled enough programmer to know how to reset the hardware and the psp's. It's not pleasant to be told that a file is not there - at least that was the DOS error message - when I could see the darned thing with my very own eyes in the directory. Again, such routines work with other Cs. Another foray into low-level DOS calls got me back into business, at the cost of another couple of hours of self-doubt.

The point is not that there could never be good reason for making stream I/O and system calls unavailable under certain conditions. However, I'd appreciate being told. Lots of expensive and potentially billable time was wasted because I innocently tried something done a half-dozen times before in other settings.

Is Borland trying to protect me from nasty side effects that MS-DOS allows? Possibly, though I'd argue that as a consenting adult anyone fool enough to do TSR programming deserves to make messes if he chooses. It took me months to discover what I could and couldn't do when DOS is interrupted, so let me play my hand.

And then there are the truly spooky times when in the middle of a compile the integrated environment would go off north-north-west and lock up or reboot the machine or give the dreaded message, "irreducible expression tree." Sometimes the glitch was reproducible; most often it wasn't.

(continued on page 60)

Figure 1 - In-Line Assembly Code for exec() Function

```
#pragma inline
/***************/
exec(char *program, char *argument)  /* new ANSI way of declaring args */
{
/*  The rules of this game are rather too involved to explain here.  See
    Ray Duncan's Advanced MSDOS, chapter 10, for a lucid explanation
    of the MSDOS's ''exec'' function. This is done in the ''large'' model so
    pointers will all be 4 bytes long and coding will be straightforward.*/
char comline[30];                     /* to construct command line */
struct{                             /* structure needed by DOS 0x4b service */
   unsigned envseg;
   char *command,*fcb1,*fcb2;
   } paramblock,*pbptr;

paramblock.envseg=0;                 /* keep the old environment */
comline[0] = strlen(argument);       /* first char: length of line */
strcpy(comline+1,argument);
strcat(comline,"\r");                /* terminate with 0xd */
paramblock.command = comline;
paramblock.fcb1=paramblock.fcb2=(char *)0xffffffff; /* Ignore fcb's */
pbptr = &paramblock;
asm push ds          /* DS & BP must be saved. Turbo saves SI & DI */
asm push bp
/* The next two routines work because this is the LARGE model and
   pointers are 32-bit entities and work with lds & les. Cute, no? */
asm lds dx,program                  /* DS:DX pointed to program path */
asm les bx,pbptr                    /* ES:BX pointed to parameter structure */
asm  mov word ptr cs:fill[0],ss     /* save SS & SP */
asm  mov word ptr cs:fill[2],sp
asm  jmp next
asm  fill dw 0,0                    /* store SS & SP in code segment */
next:
asm  mov al,0
asm  mov ah,4bh
asm  int 21h                        /* do exec call */
asm  cli
asm  mov ss,cs:fill[0]              /* restore SS & SP */
asm  mov sp,cs:fill[2]
asm  sti
asm  pop bp
asm  pop ds
}
```

C CODE FOR THE PC

source code, of course

C Source Code

FSP (screen manager)	$400
Barcode Generator (specify Code 39 (alphanumeric), Interleaved 2 of 5 (numeric), or UPC)	$300
GraphiC 4.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$275
Vitamin C (MacWindows)	$200
Essential C Utility Library (400 useful C functions)	$160
Essential Communications Library (C functions for RS-232-based communication systems)	$160
Panache C Program Generator (screen-based database management programs)	$150
PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
The Profiler (program execution profile tool)	$100
Entelekon C Function Library (screen, graphics, keyboard, string, printer, etc.)	$100
Entelekon Power Windows (menus, overlays, messages, alarms, file handling, etc.)	$100
QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
CBTree (B+tree ISAM driver, multiple variable-length keys)	$80
ME (programmer's editor with C-like macro language by Magma Software)	$75
Wendin PCNX Operating System Shell	$75
Wendin PCVMS Operating System Shell	$75
Wendin Operating System Construction Kit	$75
EZ_ASM (assembly language macros bridging C and MASM)	$60
Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
Make (macros, all languages, built-in rules)	$50
Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
Coder's Prolog (inference engine for use with C programs)	$45
PC/MPX (light-weight process manager; includes preemption and cooroutine packages)	$45
Biggerstaff's System Tools (multi-tasking window manager kit)	$40
TELE Kernel (Ken Berry's multi-tasking kernel)	$30
TELE Windows (Ken Berry's window package)	$30
Clisp (Lisp interpreter with extensive internals documentation)	$30
Translate Rules to C (YACC-like function generator for rule-based systems)	$30
6-Pack of Editors (six public domain editors for use, study & hacking)	$30
ICON (string and list processing language, Version 6 and update)	$25
LEX (lexical analyzer generator)	$25
Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
C Compiler Torture Test (checks a C compiler against K & R)	$20
PKG (task-to-task protocol package)	$20
A68 (68000 cross-assembler)	$20
Small-C (C subset compiler for 8080 and 8088)	$20
tiny-c (C subsubset interpreter including the tiny-c shell)	$20
Xlisp 1.5a (Lisp interpreter including tiny-Prolog in Lisp)	$20
List-Pac (C functions for lists, stacks, and queues)	$20
XLT Macro Processor (general purpose text translator)	$20
C Tools (exception macros, wc, pp, roff, grep, printf, hash, declare, banner, Pascal-to-C)	$15

Data

DNA Sequences (GenBank 48.0 of 10,913 sequences with fast similarity search program)	$150
Protein Sequences (roughly 4,000 protein sequences with similarity search program)	$60
Webster's Second Dictionary (234,932 words)	$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works

11100 Leafwood Lane

Austin, Texas USA 78750-3409

(512) 258-0785

Free surface shipping on prepaid orders **MasterCard/VISA**

(continued from page 58)

Once I spent hours staring at code and retesting the memory, only to solve my problem by moving the editor up in memory a K or so by adding a few more buffers to the 20 declared in my config.sys file. (Just why I tried that trick is too long a story to tell. Suffice it that memory placement was crucial although the memory chips were fine.)

The whole package feels "buggy" in a way Turbo Pascal's environment never was. Interestingly enough, the stand-alone compiler "tcc" has been solid as a rock. Of course, it's a more standard piece of software.

Conclusions Therefrom

All in all, these irritations are the sort of start-up difficulties that one might expect from version 1 of any highly complex program. The old rule about not buying an automobile engine from Detroit the first year it comes out remains a wise one. If you can wait for version 2, wait.

For those of you for whom programming productivity is money in the bank, go ahead and plunk down your $65. Just expect a little flakiness until you get the upgrade. The real problem is that many of the folks who have bought this package in such numbers are the very people least able to deal with phantom reboots and incomplete documentation.

And Yet, The Enhancements

To end on that note, however, is to leave the wrong impression. There are dozens of surprises that simply make you grin. The other side of that "system" tale above is that the in-line assembler needed to roll my own exec() function was ridiculously easy to write. I offer it in Figure 1.

Notice that there is no need for the usual [bp+N] addressing so dear to the heart of assembly hackers. The compiler takes care of that if we simply use symbol names - structure members and auto variables, even. The compiler does it for us, making adjustments for the memory model as it produces the ASM file that is passed directly to the assembler. Never was writing assembly language easier: all the segmenting and assuming and offsetting is automatic, so all you have to do is dive into the actual operations.

The one fly in the ointment - perhaps it's an entire buzzard in the ointment - is the requirement of MASM 3.0 or 4.0 for the assembly language system to work automatically. MASM, you may note, costs almost TWICE what Borland's entire package does. The breathless you-too-can-write-inline-assembler ads don't quite tell you that. I own MASM 1.27 and can do it by hand. But it is a little frustrating to buy the car for $2000 and pay $5000 for the windshield wipers.

Then there are the functions of type "interrupt." Declare a function with the reserved word "interrupt" and the generated machine code pushes and pops the registers, and saves and restores the DS register, so that the 32-bit function address can be placed directly in the interrupt table. Define the function arguments of such a routine properly, and you can write complete TSR programs without resorting to assembly language at all.

In other words, practically anybody can now toss off a resident program. Considering the oversupply of those darn things nowadays, that may turn out to be the greatest social cost of those 100,000 compilers out there. Think of it: thousands of routines in the public domain, all fighting for Alt-F1.

To demonstrate such operations, I offer in Figure 2 a TSR that deallocates TSRs above it if the alternate key and both shift keys are pressed down at once. It's hardly fun anymore; it's almost too easy.

The Bigger Picture

The appearance and overwhelming success of Turbo C mark a genuine revolution in the world of C programming. No longer will C be the possession of the select few who love to snort and say that C is obviously not something for the average hobbyist. It looks as if the average hobbyist is going to own a copy of a superb C compiler. Whether he will use it is still to be seen. It is a fine sign of the growing sophistication of the microcomputer world that something so esoteric as C can become a genuine fad.

I wish I could be as enthusiastic about the details of Borland's implementation as about its overall conception. It's just not as smooth a product as Turbo Pascal. Yet.

Perhaps the greatest benefit to those of us who already use C is that Borland just upped the ante for every maker of C compilers - in the area of cost (first of all!), in the area of documentation, in the area of ease of use. The day of the Zip-Lock bag and mimeo documentation is officially over.

Though there's nothing to match Microsoft's CodeView in Borland's package, surely Microsoft is going to be pressed hard to keep up its $450 list price. Microsoft must come forth with a QuickC and an integrated environment derived from QuickBasic. When this sort of competition occurs, everybody's the winner. God bless free enterprise.

A Bow

A little over two years ago, I wrote my first article for *Micro C* which compared Turbo Pascal to C. It seems appropriate, and nicely symmetrical, that my last should compare Turbo C to Pascal. Though techno-journalism has been fun and a refreshing change of pace, it takes time away from other kinds of writing.

My thanks to the staff of *Micro C* for giving me a soapbox to preach from. Their tolerance has been exemplary. They've never interfered with whatever obsession I happened to be chasing at the moment, no matter how arcane. It has become more and more alarming, however, to open up the magazine to discover what alien words and incomprehensible thoughts have been thrust into my mouth, but I guess that's just the culture shock of a stranger in a strange land.

Most of all I'd like to thank those of you who have corresponded with me. You are proof positive that the so-called "Hacker Ethic" of sharing and candor survives in the Cynical Eighties. The greatest appeal of computerdom is not those delightful machines or those even more delightful languages, but the genuinely open community of people who understand that by giving freely to one another we are all made richer. I at least know of no other human association whose architecture is half so open. *Micro Cornucopia* and its readers embody that ideal. Again, thanks.

Editor's note: Thanks for the memories Ron. Yes, the editing wasn't always as precise and careful as we would have liked, but it's been great having you. We'll all miss your carefully written prose and well-done code.

After we received your note, Ron, we got our heads together trying to come up with another C'er who could fill your shoes. We couldn't. We're open to C article ideas (call, write, or leave a message on the Micro C RBBS (503) 382-7643). After all, C is really hitting the big time.

■■■

Figure 2 - Getting Rid of Resident Programs

```
/* LARGE MODEL USED SO POINTERS ARE 32 BITS.
This compiles to 5.6K, which supports Borland's claim to relatively tight code.
Incidentally, I can do the same in 2.3K in DeSmet C and about 1K in assembler.
Neither coding is half so simple, however. This sort of direct manipulation of
MSDOS allocation headers is an example of ill-behaved programming at its most
incorrigible. But don't long C pointers make crime EASY??*/

#include <dos.h>
#include <stdio.h>
#define RSHIFT   1    /* corresponding bits on shift state byte @  00:417H */
#define LSHIFT   2
#define ALT      8
#define TESTIT   (RSHIFT | LSHIFT | ALT)
#define TICKINT  8
#define NEWTICK  0x80
#define FINI     0x5a
char intable[0x400],*shiftptr=(char *) 0x417L;
typedef struct {          /* MSDOS allocation header format */
    char flag;        /* Either 5Ah for end or 4Dh for not end */
    unsigned nextpsp,paragraphs;
   } HEAD;
HEAD *header;
extern unsigned _psp;
unsigned meminstall;    /* how much memory is installed in PC? */
/*******************/
void interrupt tickhandler()
{   /* check shift state for three shifts depressed */

if( (*shiftptr & TESTIT) == TESTIT) deallocate();
geninterrupt(NEWTICK);  /* chain the interrupt */
}
/***************/
deallocate()
{
HEAD *nextheader;

/* _AX is a "pseudovariable" which can be used to get or change
  the value stored in AX. The possibilities are mind-boggling. */
_AX=0xe00+7;              /* just a beep to show something happened  */
geninterrupt(0x10);
memcpy(NULL,intable,0x400);  /* restore old int table */
nextheader = MK_FP(_psp + header->paragraphs,0);
nextheader->flag = FINI;  /* tell MSDOS that everything's free above  */
nextheader->nextpsp = 0;
nextheader->paragraphs = meminstall - 1 - _psp - header->paragraphs;
}
/*********************/
main()
{
char **newtick,**oldtick; /*just so it's 32 bits. char * is a convenience.  */
header  = MK_FP(_psp - 1, 0); /* point to TSR's allocation header  */
newtick = MK_FP(0,4*NEWTICK);      /* swap interrupt vectors */
oldtick = MK_FP(0,4*TICKINT);
disable();                        /* stop hardware */
*newtick = *oldtick;
*oldtick = (char *) tickhandler;  /* put interrupt address into table  */
enable();
memcpy(intable,NULL,0x400);       /* save old interrupt table */
geninterrupt(0x12);               /*how much memory installed? */
meminstall = _AX*0x40;            /* store amount in paragraphs */
/* Note that TurboC concatenates strings for you. */
puts("\nDEINSTALLATION PROGRAM INSTALLED\n"
     "     Press Alt-Leftshift-Rightshift\n"
     "to deinstall subsequent resident programs.\n" );
keep(0,_SS - _psp +1);       /* lop off stack, terminate & stay...*/
}/* keep(0,n) uses DOS function 31h to set aside n paragraphs of memory  */
```

86 WORLD

New Toys, Modula And The Zenith 181

By Laine Stump
Redhouse Press
Merkez PK 142
34432 Sirkeci
Istanbul, Turkey

Our boy is back in Turkey after participating in the Micro C technical forum in Bend (SOG VI). This issue he's hot on the trail of the new Modula and the greatest new laptop. (Well, almost the greatest.)

By the time this issue has been printed and mailed, I will have a new address. No, no, that's not it. Let's try that again. By the time this issue has been printed and mailed, I will not have an address. There. That's what I should have said.

My two year contract with the DFT is now officially finished, and I'm officially moving on to something else. Or nothing else. Or none of the above. I will be completely unreachable during the month of October (unless you send a note to "Switzerland - General Delivery"); but starting November 1, 1987, I will be in Istanbul for awhile. Even then I don't know what my address will be, but I can give you one that will eventually get things to me:

Redhouse Press
Merkez P.K. 142
34432 Sirkeci
Istanbul, TURKEY
tel: [011] (90)-1-527-8100

I have no idea how long I will be at this address. I am writing this in mid-August and don't even want to think about where I'll be or what I'll be doing in six months.

As those of you who were at SOG this year know, I am writing this just after returning to Turkey from the U.S. I came back to the land of the Ottomans with a few new toys (and promises of a few other new toys), so I guess that's what I'll spend most of my time talking about. Oh, yeah, and I have a bug fix in here somewhere, too (now where did I put that damn flashlight...).

So which course do you want first? The hardware? Or the software? How about the vaporware??? I guess that sounds good to me...

Modula

Remember my review last issue of EXE2LNK, the program that allows linking of assembly language modules with Logitech Modula 2? Well, I really jumped the gun on that one. Logitech has a new version of their Modula compiler which can compile to, and link with, standard MS-LINK modules (.OBJ files). Now you don't need EXE2LNK. Silly me. If I had just waited a few months.

Other features of the updated Modula include: new runtime debugger interface, new text editor, more code optimization (register tracing, mostly), intelligent DOS compatible linker which only links in those parts of a module actually referenced by a program (the old version just included the whole module, even if you only used one constant), and many other goodies that slipped through my mind during the conversation.

Since I am writing in August and the new compiler won't be out until September, I am telling you all this on faith; but I think (hope?) I can trust my sources.

All I know for sure is that I'm real anxious to get my hands on a copy. Now that I no longer have to worry about dodging pools of chicken blood, computing egg counts, and assembling clones, I will have more time to devote to my true addiction: programming. If I get my software care package in time, I may be able to tell you a few things about it next issue.

Then again, maybe I'll have to write about slug control or desalination plants.

Enough of vaporware. Let's get on to...

Software

Unfortunately, the only software I saw while I was in the U.S. that really impressed me was Earl's (Hinrichs of PC Tech) graphic Mandelbrot set calculation and display program. I picked up a copy of Turbo C, too, but it's already receiving so much undeserved press that I refuse to even mention it.

Not that I'm not impressed with Turbo C. It's just that there are so many other deserving products out there that are getting ignored just so that all the computer columnists can have an

"I can give more slobbery praise than you can" contest. *(Editor's note: I thank you for pointing that out Laine, Larry thanks you, Gary thanks you, Philippe thanks you...)*

Anyway, Earl's Mandelbrots will most likely be mentioned somewhere else in the magazine, so I won't repeat any of that here. But, talk of the Brots brings up the subject of the machine they were displayed on, which means...

Hardware

The hottest hardware at the SOG was PC Tech's hi-res color graphics board. This beast is based on the TI 34010 (pronounced "threefortyten" according to Earl) graphics processor running at approximately 5 MIPs (give or take a milli MIP) and capable of displaying 256 colors from a palette of 256K at a resolution of up to 1024 x 800 (800 x 512 on a standard MultiSync monitor). If you're sick of your EGA (or have too much money for your own good), this is definitely the stocking stuffer of the year.

At SOG time, the board was emulating CGA and EGA for use with standard software packages; if you wanted to take full advantage of its power, you had to write your own software. Maybe by the time you read this, there will be a PGA emulator and a Windows driver written for it so that any program running under Windows can take full advantage of all those freaked out colors and all five of those crazy little MIPs.

For those more into desktop publishing and less into cash depletion schemes, PC Tech also has a monochrome board with basically the same features (except it's not color, of course, just shades of grey). They had one of these boards hooked up to a vertically oriented "full page" type monitor, and it was quite nice to work with.

Presently, the mono board works under Windows (and Gem, etc.) by emulating a Genius monochrome graphics adaptor. Later they will write a native Windows driver to take full advantage of the 34010 (and increase display speed by several orders of magnitude).

No danger in purchasing now, either. Historically, software updates have been available from PC Tech at minimal charges (e.g., "Just send us some blank ROMs and a few stamps"). Contact PC Tech for details on either of these products.

Mainstream Hardware

A bit further away from the jagged edge of technology: Along with my new seven-man river raft (christened "Contraband," for obvious reasons), a PC Tech monochrome board, and two bottles of French wine, I came back from the U.S. with two new playthings that will make me a more portable kind of guy. But first some background...

About a year and a half ago while in Hong Kong, I purchased a little unit that has changed my life irreversibly. I bought one of the original Toshiba T1100s. The 1100 was an IBM compatible portable with 512K of RAM, one disk drive, and a two-thirds height LCD guaranteed to earn you a new pair of glasses in 30 days or your money back. For a year and a half, my 1100 went everywhere with me - from HK back to Turkey, on a bus across Europe, to the U.S. and back, and all across Turkey north to south. I had never made a more intelligent purchase. Until now.

The problem with the Toshiba was that it had only one drive. Fine for word processing, Turbo Pascal, and MASM, but I wanted to do Modula while sitting on a bus, too. That meant I needed a new machine. And while I was home this year, I bought it. A shiny new Zenith 181.

Zenith 181

My latest love has two drives, 640 K, and a full-height backlit LCD that even your grandmother would enjoy (even if she's legally blind). Not only that, but it runs at 8 Mhz (Tosh runs at 4.77), keeps time with a battery-backed clock, and has a connector on the back panel mysteriously labelled "Ext Bus." The batteries are rated for a shorter operation time (4.5 hours compared to 8), but it's not healthy to sit in front of a screen for longer than that, anyway. The real clincher: I paid $250 less for the Zenith than I did for the Toshiba. That's progress.

I used to think that my Toshiba was the greatest little machine in the world. Now I can't stand to even look at it. That's progress. To paraphrase Oliver Wendell Jones of Bloom County: "As a rule, we hackers take obsolescence quite hard."

Diconix 150

I didn't just get a computer, either. I also bought myself a little Diconix 150 portable inkjet printer. About the size of a hardback copy of "War and Peace," this 3.75 lb. wonder is conveniently concealed in carryon baggage for easy transportation across international borders (shhh! You didn't hear it from me!). Not only that, but it does a decent job of printing, too. And it's cheap.

The Diconix printer fits handily in my backpack, does 150 cps in draft mode, uses HP Thinkjet cartridges, recognizes the Epson standard command set, can print 150 pages on a full battery charge (so they say), and is quieter than an ambassador with a gas problem at a state dinner party.

Not only that, but it looks convincingly like an electric inflation pump for my new rubber raft. At least to anyone who doesn't know much about electronics. Customs agents outside of the Far East and the U.S. don't know much about electronics. So far I'm happy with my new inflation pump, especially with its NLQ mode.

My latest love has two drives, 640K, and a full-height backlit LCD that your grandmother would enjoy (even if she's legally blind).

But there are a few problems.

First - although the printer can handle full-size single sheet and tractor feed paper, it can't print across the full width of the page. About 1/2 inch on each side is unprintable. In practical terms this usually doesn't matter, since you set margins narrower than this most of the time, anyway. It still prints 80 characters across by making each character a bit smaller, which brings us to the second problem...

Each character is slightly smaller on the Diconix than on a standard Epson. As a matter of fact, each dot occupies 1/96 inch vertically instead of 1/72 inch. Everything still prints fine, it just comes out looking smaller (especially in graphics mode). On the other hand, it's higher resolution.

There also is no paper-turning knob on the printer. This was done to save space, and I appreciate that, but it is real-

(continued next page)

(continued from page 63)

ly inconvenient to be forced to grab the tractor rings and twist to feed in new paper. Loading paper is far from automatic or convenient (really, I guess it's okay now that I've got the method down, but I still can't get over the idea that it's a kludge).

Another possible problem for some programs is the lack of a few Epson features. Downloaded character sets are not supported (not surprising, since the Diconix uses a 12-jet printhead instead of the standard 9). Similarly, quad density graphics mode (<ESC> Z) is not supported. Neither is Elite pitch (<ESC> P) or reverse line feed (<ESC> j) or half speed printing (<ESC> s).

These are small points, though. The idea of a battery powered printer to accompany my battery powered computer, the ability to print out letters and lists and samples and drafts anywhere I like, the sheer fun of watching people's faces when they realize I wasn't joking when I said my printer was in my bookbag, these all make it worthwhile. I'm glad I bought it.

The Combo

Now that I have a Zenith 181 and a Diconix 150, I can truly say that I am a "Portable person." I can go anywhere in the world, take my entire computer system with me, and run any program I could possibly need or want to use. Except PageMaker...

I Screwed Up

Some people say it takes a big man to admit he was wrong. Personally, I think it just takes someone who has made a mistake to admit he was wrong. Anyway, I'm admitting that I was wrong. What's the danger in it? After all, somebody already wrote in and said I was wrong and it was published in the last issue of *Micro C*. What have I got to lose?

Anyway, the mistake I'm talking about was the crashing Exec procedure uncovered by George L. Florman on page 68 in issue #37 (the original Exec was published way back in issue #31, but its utility is timeless). My mistake was that I didn't save the stack pointer before calling the MS-DOS exec function. It isn't that I didn't read that part in the DOS manual saying that all registers were fair game, including SS and SP; it's just that I never dreamed that anyone would be so stupid as to actually destroy the basic unit of context storage (i.e., the stack). Also, I was lazy.

Why the stack pointer gets clobbered only when running Exec on certain ATs (but not on XTs running the same version of DOS), I'll never know. The fact is that it does. I would like to know, however, if they destroyed the stack, how did they ever get back to the calling program anyway???? Something smells here. No wonder DOS isn't reentrant.

I would have just let the subject drop after seeing Mr. Florman's note, but he only explained how he fixed the problem, leaving youse guys to go thrashing through DEBUG and MASM all on your little lonesomes. Because I happen to be a fan of "plug and go" code, because I think the ability to execute external programs is an essential feature in any serious program written these days, and because I'm sure you're just as lazy as me (I like to call it "efficiency conscious"), I decided I'd better fix the bug myself and publish the corrected routine.

By the way, Mr. Florman, thanks for pointing out my mistake. I had heard of the problem, but hadn't been motivated enough to figure it out for myself.

In order to fully understand what I'm talking about, by the way, you should equip yourself with copies of *Micro C* issues #31, #32, and #37, as well as Turbo Pascal and your favorite MS-DOS machine. If you don't have these issues already, you should get them; they'll do wonders for your Pascal programs (especially #31 and #32). If you don't have an MS-DOS machine, turn the page; you're not supposed to be reading my column anyway.

The Fix (And The Explanation)

Figure 1 is a correction of the Exec procedure I gave in my column in issue #31. It is functionally identical to the original. Internally, the only difference from the original is that I save SS and SP before calling DOS and restore them immediately after returning. In the original version, I saved myself the agony of writing inline assembly by using Turbo's built-in MS-DOS procedure. But because that procedure messes around with the stack before AND after doing INT 21h, I had to change my strategy and do the INT 21h myself, which means (yeucchh!) inline code.

Those who attempted to follow Mr. Florman's directions for fixing the bug will notice a bit of difference between my code and the code you arrived at. That is because I like to conserve code space. The MS-DOS procedure loads ALL the CPU registers from a record in memory before doing INT 21h, and puts the contents of all the registers back in the record after return from DOS. This is flexible and thorough. It works for any DOS call. But it would be a waste of time for me to duplicate it all.

The only registers that must be loaded for an exec system call are AX, DS:DX, and ES:BX, and the only registers that must be saved are DS, BP, and SS:SP (I save ES, too, just to be safe). By saving and modifying only those registers I need to use or save, I saved a lot of code space. And you get a smaller program, you lucky little dipstick.

You will probably wonder about the way I arrive at the addresses to give to DOS. Why do I put the SS register into ES and DS? And why do I add BP to both of the offsets? If you had only read your Turbo manual, you would understand, dear.

Both of the structures I want to point to are local dynamic variables. Local dynamic variables are located in the stack segment, and their offsets are relative to BP (the "frame pointer" as it is sometimes called). I also increment ComFile by one because I want to point at the text of the string, not at its length byte.

The only other notable notation I should note is that I declared several variables as typed constants. In Turbo, typed constants are actually initialized variables in the code segment. I declare the variables this way because CS is the only register I can assume will be unharmed after return from DOS.

Oh, yeah. That call to the function GetEnvironment is so that I look for COMMAND.COM in an intelligent manner. See my column in issue #32 for an explanation of this tactic, as well as a listing of the GetEnvironment procedure.

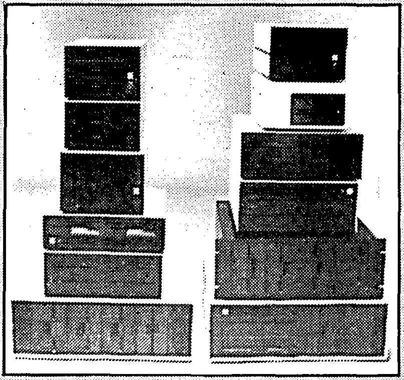
There! I think I have sufficiently atoned myself for this month. Be sure to tune in next time when I detail my thrilling adventures piloting "Contraband" down the mighty Goksu River.

■■■

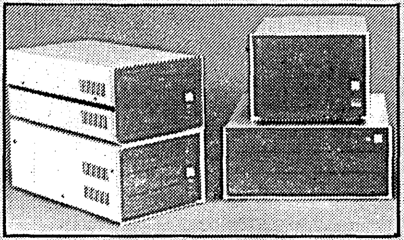
Figure 1 - Corrected Exec for Turbo Pascal

```
{--------------------------- Exec ----------------------------}
{    execute 'Command' as if it was typed at the A> prompt      }
{      requires Getenvironment function from Micro C. 32        }
{--------------------------------------------------------------}

FUNCTION Exec (Command : string128) : INTEGER;

CONST
    SSSave   : INTEGER = 0;
    SPSave   : INTEGER = 0;
    RetAX    : INTEGER = 0;
    RetFlags : INTEGER = 0;

TYPE
    ExecPacketRec = RECORD
        EnvironmentSeg : INTEGER;
        CommandPtr,
        FCB1,
        FCB2           : ^CHAR;
        end;    { ExecPacketRec }

VAR ComFile  : String20;
  ExecPack : ExecPacketRec;

  begin
  ComFile := GetEnvironment('COMSPEC')+chr(0);
                                {execute command.com}
  IF (length(command)  0) THEN    {sending it this line}
      Insert ('/c ',Command,1);   {'c' switch means 'do this'}
  command[length(command)+1] := ^M;

  ExecPack.EnvironmentSeg := $0000; {use parent's environment}
  ExecPack.CommandPtr := Ptr(seg(Command),ofs(Command));
  ExecPack.FCB1 := Ptr(0,0);
  ExecPack.FCB2 := Ptr(0,0);

    { START OF MODIFIED AREA }
  inline($06/$1E/$55/$9C/ {push es, ds, bp, flags    }
         $8C/$D0/$8E/$D8/ {mov ax,ss  mov ds,ax (seg(ComFile))}
         $BA/ComFile/         {mov dx,offset ComFile[0]    }
         $01/$EA/$42/         {add dx,bp  inc dx           }
         $8E/$C0/             {mov es,ax    (seg(ExecPack))}
         $BB/ExecPack/        {mov bx,offset ExecPack      }
         $01/$EB/             {add bx,bp                   }
         $B8/$00/$4B/         {mov ax,4B00h          }
         $2E/$8C/$16/SSSave/  {mov cs:[SSSave],ss <<-}
         $2E/$89/$26/SPSave/  {mov cs:[SPSave],sp <<-}
         $CD/$21/             {int 21h               }
         $2E/$8E/$16/SSSave/  {mov ss,cs:[SSSave] <<-}
         $2E/$8B/$26/SPSave/  {mov sp,cs:[SPSave] <<-}
         $2E/$A3/RetAX/       {mov cs:[RetAX],ax     }
         $9C/$58/             {pushf pop ax          }
         $2E/$A3/RetFlags/    {mov cs:[RetFlags],ax  }
         $9D/$5D/$1F/$07);    {pop flags, bp, ds, es }

  IF ((RetFlags and CARRY) = 0) THEN
      Exec := 0
  ELSE
      Exec := RetAX;
  { END OF MODIFIED AREA }
  end;   { Exec }
```

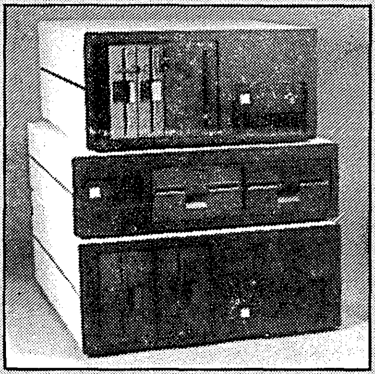
ENCLOSURES & POWER SUPPLIES



for FLOPPY WINCHESTER TAPE DRIVES from $100



for SINGLE BOARD COMPUTERS from $125



for IBM & S-100 SYSTEMS from $200

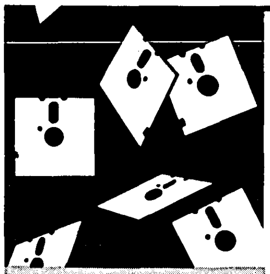
*1 piece; prices lower in quantity. (Disk drives not included)

Call or write for free catalog

INTEGRAND RESEARCH CORP.

8620 Roosevelt Ave. • Visalia, CA 93291
209/651-1203
TELEX 5106012830 (INTEGRAND UD)

*IBM TM International Business Machines



IN THE PUBLIC DOMAIN

Upstarts And Start-Ups

Anthony Barcellos
P.O. Box 2249
Davis, CA 95617-2249
(916) 756-4866

The feature this issue is Eric Isaacson's shareware assembler. It's fast, powerful, and should be real competition for Microsoft.

You have to admire these fellows. Imagine having the nerve to take on, for example, a giant corporation like Microsoft (ever heard of it?). Imagine doing it *alone*. Call your lawyer first and name your next-of-kin.

A86

Eric Isaacson is the fearless shareware author who is playing David to Microsoft's Goliath. His A86 macro assembler is now up to version 3.02 and has all the speed of a slingshot.

"A86 is blazingly fast," says Isaacson. "Don't believe the advertisements of that other, big company. *This* is the fastest MS-DOS macro assembler, bar none." Isaacson claims that on an 8 MHz AT with a RAM disk, a large program will assemble at better than a thousand lines per second.

What can you do with A86? Naturally you can assemble .COM files for stand-alone execution. And it's not surprising that you can generate .OBJ files to use with a linker. But you can also produce code that is suitable for burning into ROM. (That's what "ROMable" code is.) With several laptop computers now running software in ROM, this A86 feature should be attractive to professional programmers interested in this burgeoning market.

Isaacson offers a measure of human-friendliness with error messages written in English. At the user's option, these messages may appear at the appropriate points right in the source code file or be directed to a separate message file. If you let A86 embed the error messages in your source file, it will also strip them out when you reassemble your program.

It's For Real

Integers are nice and neat, but many applications insist on floating-point numbers. Floating-point routines are built into languages like BASIC, but assembly language programmers are accustomed to doing all of the dirty work themselves. (Explaining long division to a microchip is especially nasty.)

A86 comes to the rescue with four-function arithmetic routines for floating-point operations. Isaacson has already done a lot of the work for you. Since A86 also supports the mathematics co-processors (8087 and 80287), the floating-point operations can be extremely fast.

Credentials

The roster of A86 features looks pretty good. A companion program, D86, provides fancy symbolic debugging features. Overall, the package appears to offer great power and functionality.

So who is this Isaacson character that he thinks he can beat Microsoft at its own game? A former Intel employee, Isaacson has been working on chips since the Intel 8080. He was half of the development team for Intel's ASM86 assembler. His book on the architecture of the 80386/80387 was set for release by John Wiley & Sons during the summer of 1987.

Since A86 is shareware, it is readily available from software libraries and electronic bulletin board systems. To register your copy, send $40 to:

Eric Isaacson
416 E. University Street
Bloomington, IN 47401

If you send $50, Isaacson will send you an update disk that contains the A86LIB tool that is provided only to registered users. For $80, Isaacson provides a registered copy of the D86 symbolic debugger in addition to the update disk. Registered users are entitled to future updates of A86 for only $10 (in 1987 dollars, says Isaacson, hedging a bit for inflation). Visa and MasterCard holders can call (812) 339-1811.

User-supported software has given stiff competition to regular commercial packages in several areas. With the advent of A86 and D86, even the high-end low-level language field has a credible shareware alternative. You'd better duck, Microsoft!

BOYAN

Shareware got its start because the IBM PC didn't have any decent communications software. Andrew Fluegelman wrote PC-Talk and the rest is history.

Eric Isaacson is the fearless shareware author who is playing David to Microsoft's Goliath.

Eventually, powerful programs like SmartComm and CrossTalk appeared in the regular commercial channels, and ProComm and Qmodem shoved PC-Talk aside in the shareware market. With so many good programs to choose from, who needs any others?

Justin Boyan can be forgiven for not having noticed that the market for communications software was glutted. After all, he's only 17. If only someone had been kind enough to clue him in, Justin could have been spared the embarrassment of trying to squeeze in among the big boys, right?

Think again. The BOYAN communications program is not only highly competitive, it has the slick gloss of professional-level programming. From its subdued aural cues to its context-sensitive help, BOYAN is a program that belies its origins. This is not kid-ware, folks.

BOYAN is an "open" program that welcomes customization and additions. BOYAN's macro command language permits unattended, automatic operation. Its script facility is sophisticated enough to allow a PC to operate in host mode. (Boyan is reportedly at work on a host mode script for imminent release.)

Justin Boyan offers two reasons for his shareware debut: "I wanted many features that weren't available in the other programs; and second, because I really need money for college." Boyan graduated from high school this year and is enrolled at the University of Chicago. It will take plenty of $35 registration fees to pay his tuition.

Features

BOYAN supports communications rates from 300 to 9600 bps. The standard file transfer protocols are built-in (ASCII, Xmodem, Xmodem/CRC), and others can be added by the user. BOYAN also notices when you ask to download a file from a bulletin board and automatically captures the filename as the default name to which the transmitted data should be saved (instead of making you enter the name twice).

While other programs permit prompted ASCII upload for transmission of messages prepared beforehand, BOYAN automatically determines the prompt character and handles word-wrap problems during the upload. Boyan says that this is a BOYAN exclusive (and Boyan should know).

The "log" feature of most communications programs merely opens a capture buffer for all screen displays. BOYAN offers a usage log that records the lengths of all calls and appends such information as the efficiency of any file transfers.

Instead of requiring a separate directory maintenance utility, BOYAN can itself sort its dialing directories by name or number.

The "backspace editor" in BOYAN lets you correct errors in a line of text before it's sent to a BBS.

"Once you get used to the Backspace Editor," says Boyan, "you will find that you never post messages with typos again!"

Anyone who frequents the BBS world can appreciate the need for this feature.

BOYAN is presently shipping version D3. Send your $35 registration fee to:

Justin Boyan
9458 Two Hills Court
Columbia, MD 21045

For an additional $10 or a stamped disk mailer and diskette, Boyan will send you the latest version of his program. Anyone wishing to download the BOYAN program can call Bruce Felstein's BOYAN Support Board at (301) 495-7323; the board runs 24 hours and supports speeds up to 9600 bps.

Shareware Grows Up

User-supported software has been staking out new territory in its bid to be taken more seriously. The recently organized Association of Shareware Professionals (ASP) has announced its intention of assisting program authors who want to go the shareware route.

What's In A Name?

Since even the large-circulation PC magazines frequently get it wrong, one of ASP's highest priorities is to nail down a good definition of shareware.

First of all, "public domain" is not synonymous. Shareware programs bear their authors' copyrights and are distributed freely only by permission.

(continued next page)

(continued from page 67)

Public domain programs are not as protected. Shareware programs are commercial programs with an unorthodox distribution system - namely, users group software libraries, electronic bulletin boards, and friend-to-friend "sharing."

In fact, that is the nub of ASP's formal definition: "Shareware is a distribution method, not a type of software."

A Rose By Any Other Name

Shareware has had its rough spots since Fluegelman invented the concept for his PC-Talk communications program. Furthermore, such software escapes the Darwinian struggle for dealers' shelf-space that quickly banishes unsuccessful examples of standard software. That is, a weak shareware program can bounce about the bulletin board systems and molder in software libraries long after a similar program has entirely vanished from the normal marketplace. Thus the shareware landscape is littered with decaying corpses. Shareware critics seize upon these as examples that user-supported software is inferior to the standard stuff.

It is also easy to find shareware that was issued by authors who haven't quite gotten the idea. The anarchic opportunities afforded by the shareware concept spawned programs that earned such insulting nicknames as "crippleware," "demoware," and "beggarware."

The first two monikers refer to programs that have been deliberately incapacitated. In extreme cases, the result was no more than a canned demo. The sales pitch could be phrased as: "Take my word for it, I'm really wonderful. Send money and I'll prove it. But not before."

Other shareware contenders distinguished themselves by incessant pitches for payment. These could take the form of anything from flashing screen messages to "a word from our sponsor" that freezes all activity until the sales pitch is burned into your retinas.

Clearly, user-supported is not necessarily the same as user-friendly.

Shareware Standards

ASP hopes to establish guidelines for shareware authors that will professionalize the field, improve user satisfaction, and increase voluntary registration. As a software librarian, I have a few suggestions for shareware authors.

1. Always include a list and description of the files that belong to your program. Don't forget that shareware programs can get jumbled together. A file list makes it easier for users to preserve the integrity of a shareware package when making copies for friends.

2. Distribute a fully-functioning version of the program. It's all right to offer additional utilities or special features to those who register, but shareware should permit users to "try before you buy." Crippled software doesn't do that.

3. Keep the registration instructions simple. Complicated fee schedules and innumerable registration options just make it easier for users to put it off.

4. Provide genuine documentation. That doesn't mean you can't abridge it to fit on the distribution diskette, but omitting too much of the user's manual will prevent users from getting an accurate feel for your program. Certainly all basic program functions should be fully documented.

5. Use standard ASCII text files for your documentation. That means no WordStar documents (with high-bit characters), no embedded printer codes, and no "naked" line feeds. Why should you limit your market to people who happen to have one particular word processing program or one specific model of printer?

6. Include a "permission to copy" statement in your documentation. This encourages individuals and users groups to distribute your programs. (A conscientious software librarian looks for "permission to copy" statements to ensure that the program really is shareware.)

7. Acknowledge registration fees. When a user mails in a check, you should at least drop a postcard in response. If you offer program updates in return for registration, send the update promptly. Remember that shareware is a commercial product prepared by professionals, right?

The Association Of Shareware Professionals

Current and potential shareware authors should contact the Association of Shareware Professionals by writing to:

ASP Membership
11058 Main Street, Suite 225
Bellevue, WA 98006

Interested parties can also track down ASP on CompuServe's IBMNET.

The user community will be watching as ASP takes a stab at raising shareware out of its awkward youth, preparing it for maturity.

What Users Can Do

1. Pay for your programs. Unsupported shareware dies of starvation. If you use a shareware program regularly, then you must have found some value in it. Send in your fee and keep it alive.

2. Remember the "share" in "shareware." You can be part of the distribution system. Pass along programs you like to friends and associates. Make sure that your local users group gets a copy for its software library. (Be careful to keep all pertinent files together.) The more people who use your favorite shareware program, the more likely it is to thrive and survive.

3. Make it better. You're the user. How do you use it? What works? What doesn't? What's missing? No software gets updated more rapidly than shareware. Authors care what you think.

■■■

ANNOUNCING A
CALCULATED BREAKTHROUGH IN COMPUTING

If you use a SCIENTIFIC OR FINANCIAL CALCULATOR and a PERSONAL COMPUTER, you need the
PC HYPERCALCULATOR.

YOU NEED: A pop-up scientific/financial calculator for the IBM PC **integrated with every program you use.**

YOU NEED: A programmable calculator with 100 registers, 1000 program steps, and alphanumeric prompts.

YOU NEED: A faithful emulation of the Hewlett-Packard HP-11C and HP-12C that runs 20-40 times faster.

YOU NEED: Only $49.95 plus $3 for shipping (includes **free 8087 version and utility programs**).

Stop copying from calculator to computer **now!**
Call toll-free:
(800) 628-2828, ext. 502

Sunderland Software Associates
Post Office Box 7000-64
Redondo Beach, CA 90277

HP-11C, HP-12C and IBM PC are trademarks of Hewlett-Packard Co. and International Business Machines Corp., respectively.

EMERALD MICROWARE – Your CP/M and MS-DOS Connection

Hard Disks for your Z80 CP/M computer from Emerald Microware and MICROCode Consulting

No other upgrade improves your computer's productivity like a hard disk. We have all the hardware and software to install a hard drive on your Xerox 820, Kaypro, Zorba, or almost any Z80 CP/M 2.2 system.

HDS Host Board with Winchester Connection software ... $ 89.00
HDS Board with software and WD1002-05 board $250.00
WD1002-05 Hard Drive Controller Board $185.00
Rodime, LaPine, & Miniscribe hard drives, and XT controller cards – call for prices

The KayPLUS ROM Package by MICROCode Consulting

Get the performance of a Kaypro 10 and more, even on your Kaypro 2. Lets you install up to four floppies and two hard drives, with no software assembly required. Adds features such as automatic screen blanking, type-ahead buffer, boot from hard drive, and quad density support. Includes manual, standard utilities, AND hard disk utilities

KayPLUS ROM Set .... $ 69.95
KayPLUS ROM Set with QP/M ★★ **SPECIAL** ★★ $115.00

QP/M by MICROCode Consulting, CP/M 2.2 compatibility with outstanding performance.

QP/M adds features such as automatic disk relogging, drive/user selection from colon, 31 user areas, drive search path, and transparent time/date stamping; all in the same space as CP/M. Installs from a convenient customization menu, no software assembly required. Bootable disks available with CBIOS for Kaypro, Xerox, & BBI.

QP/M Operating System, complete ready to boot .... $ 80.00
QP/M without CBIOS (installs on any Z80 system) ...... $ 60.00

MicroSolutions - Software and hardware to link CP/M and MS-DOS

UniForm-PC by MicroSolutions .... $ 64.95
This program allows you to read, write, copy, and format diskettes for over a hundred CP/M and MSDOS computers on your PC, XT, or AT, including 8", 96 TPI, high density, and 3 1/2" formats (with optional hardware). Once installed, UniForm stays memory resident so you can use your standard DOS commands and other programs directly on your original diskettes.
Uniform for Kaypro and other machines .... $ 64.95

UniDOS by MicroSolutions .... $ 64.95
UniDOS program uses the NEC V20 CPU chips to actually RUN your favorite 8080 CP/M programs on your PC. Use UniDOS with UniForm-PC, and you can run them directly from your CP/M format diskettes. All standard CP/M BDOS calls are supported.
UniDOS w/UniForm and V20 chip $135.00

★★★ NEW ★★★ UniDOS Z80 Coprocessor Board ... $179.95
This 8Mhz. Z80H half-card runs your Z80 and 8080 code programs at LIGHTNING speed on your PC or AT. Functions just like the UniDOS program, except NO V20 or emulation mode is required to run your programs.

The CompatiCard by MicroSolutions $169.95
This half-card floppy controller allows you to run up to four 8", 5¼"(standard, 96 TPI, or high density), or 3½" disk drives on your PC/XT. With the Compaticard and the UniForm-PC program you can format, read, and write almost all CP/M and MSDOS di sk formats.
Compaticard with UniFORM-PC ★★★ SPECIAL ★★★ $225.00

MatchPoint-PC by MicroSolutions $169.95
This half-card allows you to read and write to **NorthStar** hard sector, **Apple DOS, PRODOS,** and **Apple CP/M** diskettes on your PC. INCLUDES a copy of the UniForm-PC program, as well as utilities to format disks, copy, delete, and view files.

Call or write for our complete catalog of parts and accessories for the Kaypro, Xerox 820, and PC/AT's. Full repair services available for Kaypro, Morrow, Xerox, disk drives, and most clones.

★★★★★★★ SUPER SPECIALS!! ★★★★★★★★

PC-Mastercard by Magnum Computer

Up to 1.5 Megabytes of RAMDISK and PRINT SPOOLER (or boost your system to 640k), with serial, parallel, game ports, and real time clock! This is one of the BEST multi-function cards on the market. Can use mixed banks of 64K and 256K chips. Comes with memory manager, ramdisk, spooler, and diagnostic software.
PC-MASTERCARD (0k installed) .... $ 69.95
call for pricing on 384k and 1.5M boards

Turbo Editor Toolbox by Borland .... $ 19.95
Ever wanted to add text editing to your Turbo Pascal application, or write a word processor that does things the way that YOU want? Comes with source for two sample editors, modules for windowing, multi-tasking, and many other options. Requires PC with Turbo Pascal 3.0.

COPY II PC by Central Point Software .... $ 19.95
Stop worrying about your copy protected disks. COPY II PC lets you back them up, so you can keep going when your master disk can't.

WordStar V3.3 Manual .... $ 12.00
Genuine MicroPro manuals in hard binder pack

InfoStar Manual Set .... $ 18.00
Genuine DataStar and ReportStar manuals in MicroPro three vol. set

8" Generic Diskettes - Ten Pack .... $ 7.50
8" Generic Diskettes - Hundred Pack .... $ 65.00
Single sided, SD or DD, with Tyvec sleeves

★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★★

Four Device Printer/Data Switch .... $ 49.95
Quality with economy. These four port boxes can be used with either RS232, or IBM parallel (DB25) printer cables.
IBM style Parallel Printer Cable .... $ 12.00

300/1200 Baud Internal Modem $117.00
Half-card, Hayes compatible, auto-answer, auto dial

CT-6260 MULTI I/O - FLOPPY DISK CONTROLLER .... $ 69.95
Half-card parallel, COM1, optional COM2, game port, floppy disk controller, real time clock, with manual and cables.

TWO DRIVE FLOPPY DISK CONTROLLER .... $ 29.95
Half-card floppy controller with cable.

Double Density for the Xerox 820-1 by Emerald Microware and MICROCode Consulting

Run up to four 5¼" (48 or 96 TPI) and 8" drives at once. Get support for all standard printers, mini-monitor functions, autoboot capability, 19 built in disk formats, and banked ROM-BIOS for more TPA. Software compatible with Kaypro and Xerox 820.
Plus2 ROM Set and X120 Board A&T $135.00
Plus2 ROM Set and X120 Bare Board .... $ 62.00
Plus2 ROM Set only .... $ 49.95
X120 Bare Board only ★★★ **CLOSE-OUT SPECIAL** ★★★$ 15.00
or two for $ 25.00, five for $ 50.00

VISA

EMERALD MICROWARE

MasterCard

P.O. Box 1726, Beaverton, OR 97075
(503) 641-0347

Prices subject to change without notice. Include $4.00 shipping and handling, $7.00 for COD, call for Blue Label charges. VISA and Mastercard accepted. 30 day money back guarantee on all products.

THE KAYPRO COLUMN

Undocumented Z80 Instructions

Walter Rottenkolber
Box 936
Visalia, CA 93279

I'd heard rumors about undocumented Z80 opcodes before I got my first Z80. Maybe that's why I got my first Z80. It had secrets. Herein Walter discovers some of these secret instructions hidden inside an otherwise innocent-looking piece of code. This is sleuthing at its most... (I'm thinking).

It was a dark and stormy night. A Friday night, as I recall. The day had been rough and I truly felt I'd spent it in the trenches, the bottom of the trenches. I sat down at the computer in the mood to disassemble - something or someone.

And so it was while disassembling one of my favorite programs that I came across strange code. Code that made no sense:

```
DB  OFDH      or     DB  ODDH
DEC H                LD  H,1
```

Not Satanic Messages

After many days puzzling over this, I decided these were not secret Satanic messages, the output of a demented disassembler, or random noise. After all, the program did work. And since ODDH and OFDH prefix the instructions of the index registers, IX and IY, it appeared likely that the codes were related to them. So I followed Sherlock Holmes' dictum: namely, that after eliminating the impossible, whatever remains, however improbable, must be the answer. In this case, I concluded that the code was undocumented Z80 instructions.

Figure 1 - Z80 Instructions

Reg. XH or YH		Reg. XL or YL	
ADC	A,H	ADC	A,L
ADD	H	ADD	L
AND	H	AND	L
CP	H	CP	L
DEC	H	DEC	L
INC	H	INC	L
LD	H,n	LD	L,n
LD	H,r	LD	L,r
LD	r,H	LD	r,L
OR	H	OR	L
SBC	A,H	SBC	A,L
SUB	H	SUB	L
XOR	H	XOR	L

```
''n'' = a Byte Value. (O - OFFH)
''r'' = reg A,B,C,D,E,H, or L.
H and L in this case refer to the
high or low byte of IX or IY reg.
```

I guessed that these codes somehow split index registers into low and high bytes much the same as you can with the HL register. When I wrote a routine to test my guess, it worked!

Believing these codes to be in the little known, rather than the dark secret category, I checked through my references. But only one book, Alan R. Miller's *8080/Z80 Assembly Language,* confirmed their existence. Ten years ago they were apparently well known, though only one assembler incorporated them - Allen Ashley's PDS assembler.

In that assembler, Allen assigned the names XH and XL, YH and YL for the high and low bytes of the IX and IY registers.

Figure 1 contains a list of working instructions. Notice that they are limited to the basic Intel 8080 code level for the H and L registers. Not functioning are the Zilog superset instructions: input and output, bit rotations and shifts; and bit testing, setting and resetting. Also not functioning are such full register HL instructions as EX DE,HL or SBC HL,DE.

Secret Advantages

A major advantage of these undocumented codes over the standard index registers is that arithmetic and logical operations effect all the appropriate flags. In contrast, the increment and decrement functions don't set the Zero flag in the standard index register. Also, the EXX instruction does not effect them, so they can hold data common to both the standard and alternate register sets.

Except for the prefix byte (ODDH or OFDH), these instruction codes are identical to those of the H and L registers. Therefore, direct interactions of the XH, XL, YH, and YL registers with the H and L registers are not possible. For example:

```
DB    ODDH
LD     L,H
```

will move the high byte of the IX register into the low byte of the IX register, not into the low byte of the HL register. So transferring data between the two register sets has to be done via other registers, or the stack.

Using these undocumented Z80 instructions is simple.

The easiest way is to precede the instruction for the H or L register, which in this case would stand for high or low byte, by a DB ODDH for the IX register instruction, or DB OFDH for the IY register instruction (see Figure 2). Define byte may be DB or DEFB, depending on your assembler.

Macroassemblers Take Note

If you have a Macroassembler, you can write a set of macros in the style of:

```
LDY    MACRO  REG1,REG2
       DEFB   ODDH
       LD     REG1,REG2
       ENDM
```

The macro call - LDY H,A - will then generate the code for you.

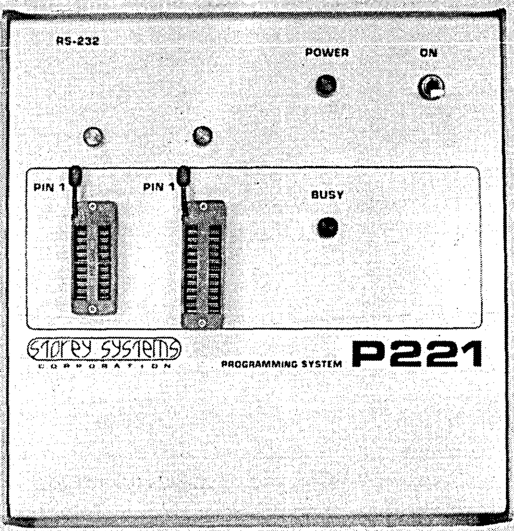
And, finally, if you are a Hacker, you could modify a Z80 assembler to respond to a set of XH, XL, YH, and YL register opcodes.

Clark Calkins of C.C. Software offers a listing of his great, much modified version of the Crowe Z80 assembler at a modest price. This is the same one he includes with his Turbo Pascal disassembly program. For rock bottom cost, though, you can't beat Micro Cornucopia's offer-

(continued next page)

Figure 2 - Undocumented Z80 Codes

```
;*****************************************
;        T3.ASM
;TEST OF UNDOC. Z80 CODES SPEED.
;*****************************************

        ORG     100H

START:
        LD      E,0FFH
        LD      C,6
        CALL    5
        OR      A
        JP      Z,START         ;WAIT FOR KEYPRESS
        CALL    SHOWX

        DB      0DDH            ;SET UP OUTER LOOP
        LD      H,0FFH
OUTLP:
        DB      0FDH            ;SET UP MIDDLE LOOP
        LD      H,0FFH
MIDLP:
        DB      0FDH            ;SET UP INNER LOOP
        LD      L,0FFH
INLP:
        DB      0FDH
        DEC     L
        JP      NZ,INLP

        DB      0FDH
        DEC     H
        JP      NZ,MIDLP

        DB      0DDH
        DEC     H
        JP      NZ,OUTLP
        CALL    SHOWX
        RET

SHOWX:  LD      E,7             ;BELL
        CALL    CONOUT
        LD      E,'X'
CONOUT: LD      C,6
        CALL    5
        RET

;*****************************************
        END             ;OF T3.ASM
;*****************************************
```



PLD PROGRAMMING SYSTEM **$995**

- LOWEST COST
- SUPPORTS MMI, NATIONAL, TI
- COMPLETE PLD DEVELOPMENT SYSTEM
- BUILT-IN COMPILER

CALL 1-800-852-2022

STOREY SYSTEMS
CORPORATION

3201 North Hwy. 67 Suite E Mesquite, Texas 75150 • (214) 270-4135

INTRODUCING:

PROBE-ONE

LOGIC ANALYZER

COMPACT 20 MHZ ANALYZER-IN-A-POD
INTERFACES DIRECTLY TO PC PARALLEL PORT

- EIGHT DATA CHANNELS
- MEMORY RESIDENT CONTROL/DISPLAY PROGRAM
- DATA CAPTURE RAM HOLDS 4096 SAMPLES
- 3 BIT MASKABLE TRIGGER OR EXTERNAL TRIGGER
- 5 INTERNAL CLOCK RATES UP TO 20 MHZ PLUS EXTERNAL CLOCK
- COMPLETE WITH REFERENCE MANUAL, TEST LEADS, AND 90 DAY WARRANTY

- PROBE-ONE (16 MHZ)...........$345
- PROBE-ONE (20 MHZ)...........$395

FOR MORE INFORMATION OR TO ORDER, PLEASE CONTACT:

LOGIC CONNECTION INTERNATIONAL
P.O. BOX 23852
PORTLAND, OR 97223
(503) 626-8468
MASTERCARD VISA

(continued from page 71)

ing. I'd get the later disk, K25, as it not only has the source of Jim Owen's fine modification of the Crowe Z80 assembler, but a Z80 Macroassembler as well. (No listing of it, however. Sigh.)

How well do these instructions work? References to the index registers abound with words like "slow," "sluggish," or even, "ponderous." Well, there's no use getting excited over turkey code, so to check it out I wrote a loop program (see Figure 2) which decrements 16,777,216 to 0, and modified it into five test programs.

1. Test A uses the undocumented codes.

2. Test B replaces the DBs with NOPs, and the XH with the B register. The NOPs simulate the extra fetch needed for the DBs.

3. Test C removes the NOPs and is the "Gold Standard" for speed.

4. Test D uses the alternate register set via the EXX instruction.

5. Test E uses the brute force method. You know: Push AF on the stack, fetch a variable from memory, etc.

Tests D and E try schemes which don't use the undocumented codes. Timing was done with a watch, and this proved accurate enough. Figure 3 summarizes the results.

Figure 3 - Z80 Instructions Timing Results

Test	Description	Code Size in bytes	Time/sec.	Time/DEC microsec	Clock cycles /DEC
A	Undoc. code	50	60	3.58	18
B	Reg. & NOP's	50	60	3.58	18
C	Std. Reg.	44	47	2.80	14
D	Use EXX	56	73	4.35	22
E	Use Mem. Var.	76	203	12.10	61

Tests were run on a 5 MHz. Kaypro II.
Accuracy +/- 1% (est.).

As expected, the undocumented instructions are indeed slower than using standard register codes. However, the speed reduction is due solely to the extra instruction byte fetch. Depending on your point of view, the undocumented codes are either 28% slower than the standard codes, or the standard codes are 22% faster. Compared to the alternative methods, however, the undocumented codes are speed demons. They ran 22% faster than switching to the alternate register set, and an incredible 238% faster than the brute force method.

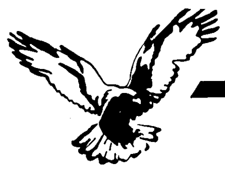
Simple to use and fast, these undocumented Z80 instruction codes can provide an elegant solution to finding room for one more byte of data when you push registers to the max.

References

Alan R. Miller: *8080/Z80 Assembly Language*, John Wiley & Sons, Inc., New York, 1981.

C.C. Software
1907 Alvarado Ave.
Walnut Creek, CA 94596

■■■



Z sets you free!

Who we are

Echelon is a unique company, oriented exclusively toward your CP/M-compatible computer. Echelon offers top quality software at extremely low prices; customers are overwhelmed at the amount of software they recieve when buying our products. For example, the Z-Com product comes with approximately 92 utility programs; and our TERM III communications package runs to a full megabyte of files. This is real value for your software dollar.

ZCPR 3.3

Echelon is famous for our operating systems products. ZCPR3, our CP/M enhancement, was written by a software professional who wanted to add features normally found in minicomputer and mainframe operating systems to his home computer. He succeeded wonderfully, and ZCPR3 has become the environment of choice for "power" CP/M-compatible users. Add the fine-tuning and enhancements of the now-available ZCPR 3.3 to the original ZCPR 3.0, and the result is truly flexible modern software technology, surpassing any disk operating system on the market today. Get our catalog for more information - there's four pages of discussion regarding ZCPR3, explaining the benefits available to you by using it.

Z-System

Z-System is Echelon's complete disk operating system, which includes ZCPR3 and ZRDOS. It is a complete 100% compatible replacement for CP/M 2.2. ZRDOS adds even more utility programs, and has the nice feature of no need to warm boot (^C) after changing a disk. Hard disk users can take advantage of ZRDOS "archive" status file handling to make incremental backup fast and easy. Because ZRDOS is written to take full advantage of the Z80, it executes faster than ordinary CP/M and can improve your system's performance by up to 10%.

Installing ZCPR3/Z-System

Echelon offers ZCPR3/Z-System in many different forms. For $49 you get the complete source code to ZCPR3 and the installation files. However, this takes some experience with assembly language programming to get running, as you must perform the installation yourself.

For users who are not qualified in assembly language programming, Echelon offers our "auto-install" products. Z-Com is our 100% complete Z-System which even a monkey can install, because it installs itself. We offer a money-back guarantee if it doesn't install properly on your system. Z-Com includes many interesting utility programs, like UNERASE, MENU, VFILER, and much more.

Echelon also offers "bootable" disks for some CP/M computers, which require absolutely no installation, and are capable of reconfiguration to change ZCPR3's memory requirements. Bootable disks are available for Kaypro Z80 and Morrow MD3 computers.

Z80 Turbo Modula-2

We are proud to offer the finest high-level language programming environment available for CP/M-compatible machines. Our Turbo Modula-2 package was created by a famous language developer, and allows you to create your own programs using the latest technology in computer languages - Modula-2. This package includes full-screen editor, compiler, linker, menu shell, library manager, installation program, module library, the 552 page user's guide, and more. Everything needed to produce useful programs is included.

"Turbo Modula-2 is fast...[Sieve benchmark] runs almost three times as fast as the same program compiled by Turbo Pascal...Turbo Modula-2 is well documented...Turbo's librarian is excellent". - Micro Cornucopia #35

BGii (Backgrounder 2)

BGii adds a new dimension to your Z-System or CP/M 2.2 computer system by creating a "non-concurrent multitasking extension" to your operating system. This means that you can actually have two programs active in your machine, one or both "suspended", and one currently executing. You may then swap back and forth between tasks as you see fit. For example, you can suspend your telecommunications session with a remote computer to compose a message with your full-screen editor. Or suspend your spreadsheet to look up information in your database. This is very handy in an office environment, where constant interruption of your work is to be expected. It's a significant enhancement to Z-System and an enormous enhancement to CP/M.

BGii adds much more than this swap capability. There's a background print spooler, keyboard "macro key" generator, built-in calculator, screen dump, the capability of cutting and pasting text between programs, and a host of other features.

For best results, we recommend BGii be used only on systems with hard disk or RAMdisk.

JetFind

A string search utility is indispensible for people who have built up a large collection of documents. Think of how difficult it could be to find the document to "Mr. Smith" in your collection of 500 files. Unless you have a string search utility, the only option is to examine them manually, one by one.

JetFind is a powerful string search utility which works under any CP/M-compatible operating system. It can search for strings in text files of all sorts - straight ASCII, WordStar, library (.LBR) file members, "squeezed" files, and "crunched" files. JetFind is very smart and very fast, faster than any other string searcher on the market or in the public domain (we know, we tested them).

Software Update Service

We were suprised when sales of our Software Update Service (SUS) subscriptions far exceeded expectations. SUS is intended for our customers who don't have easy access to our Z-Node network of remote access systems. At least nine times per year, we mail a disk of software collected from Z-Node Central to you. This covers non-proprietary programs and files discussed in our Z-NEWS newsletter. You can subscribe for one year, six months, or purchase individual SUS disks.

There's More

We couldn't fit all Echelon has to offer on a single page (you can see how small this typeface is already!). We haven't begun to talk about the many additional software packages and publications we offer. Send in the coupon below and just check the "Requesting Catalog" box for more information.

Item	Name	Price	
1	ZCPR3 Core Installation Package	$49.00	(3 disks)
2	ZCPR3 Utilities Package	$89.00	(10 disks)
5	Z-Com (Auto-Install Complete Z-System)	$119.00	(5 disks) *
6	Z-Com "Bare Minimum"	$69.95	(1 disks)
10	BGii Backgrounder 2	$75.00	(2 disks)
12	PUBLIC ZRDOS Plus (by itself)	$59.50	(1 disk)
13	Kaypro Z-System Bootable Disk	$69.95	(3 disks)
14	Morrow MD3 Z-System Bootable Disk	$69.95	(2 disks)
16	QUICK-TASK Realtime Executive	$249.00	(3 disks)
17	DateStamper file time/date stamping	$49.95	(1 disk)
18	Software Update Service	$85.00	(1 yr sub)
20	ZAS/ZLINK Macro Assembler and Linker	$69.00	(1 disk)
21	ZDM Debugger for 8080/Z80/HD64180 CPU's	$50.00	(1 disk)
22	Translators for Assembler Source code	$51.00	(1 disk)
23	REVAS3/4 Disassembler	$90.00	(1 disk)
24	Special Items 20 through 23	$169.00	(4 disks)
25	DSD-80 Full Screen Debugger	$129.95	(1 disk)
27	The Libraries.SYSLIB, Z3LIB, and VLIB	$99.00	(8 disks)
28	Graphics and Windows Libraries	$49.00	(1 disk)
29	Special Items 27, 28, and 82	$149.00	(9 disks)
30	Z80 Turbo Modula-2 Language System	$89.95	(1 disk)
40	Input/Output Recorder IOP (I/OR)	$39.95	(1 disk)
41	Background Printer IOP (BPrinter)	$39.95	(1 disk)
44	NuKey Key Redefiner IOP	$39.95	(1 disk)
45	Special Items 40 through 44	$89.95	(3 disks)
60	DISCAT Disk cataloging system	$39.99	(1 disk)
61	TERM3 Communications System	$99.00	(6 disks)
64	Z-Msg Message Handling System	$99.00	(1 disk)
66	JetFind String Search Utility	$49.95	(1 disk)
81	ZCPR3: The Manual bound, 350 pages	$19.95	
82	ZCPR3: The Libraries 310 pages	$29.95	
83	Z-NEWS Newsletter, 1 yr subscription	$24.00	
84	ZCPR3 and IOPs 50 pages	$9.95	
85	ZRDOS Programmer's Manual 35 pages	$8.95	
88	Z-System User's Guide 80 page tutorial	$14.95	

* Includes ZCPR3: The Manual

Echelon, Inc.

885 N. San Antonio Road, Los Altos, CA 94022 USA
415/948-3820 (order line and tech support)
Telex 4931646

NAME ______________________________

ADDRESS ______________________________

____

TELEPHONE ______________ DISK FORMAT ________

☐ REQUESTING CATALOG

ORDER FORM

Payment to be made by:

☐ Cash
☐ Check
☐ Money Order
☐ UPS COD
☐ Mastercard/Visa:

# ______________

Exp. Date ______________

California residents add 7% sales tax. Add $4.00 shipping/handling in North America, actual cost elsewhere.

ITEM	PRICE
________	________
________	________
________	________
Subtotal	________
Sales Tax	________
Shipping/Handling	________
Total	________

CP/M Notes

Fish Story

I am writing to you on my "new" Kaypro 8 (once a Kaypro II-83). Thanks to your magazine's wonderful instructions, I've survived the bewildering but rewarding odyssey through the soul of my machine. Among the rewards was the delightful surprise of calling Micro C's technical help line and being greeted by none other than the E&P himself.

One of my calls concerned a problem in which two disk drives were selected every time I turned on the system. You suggested I may have zapped the drive cable when I added additional drive connectors. A new cable worked with the drives out on the workbench, but not with everything stuffed back in the cabinet.

An engineer friend found the solution. Eyeing the tight fit of my two half-height Mitsubishis, he said the select jumpers were shorting out: A against the drive enclosure, and B against A. He gave me some stiff cardboard-like material called "fish paper" to insert between the drives as an insulator. He assured me that this paper really does find use in some electrical systems. I don't know about that, but at least the disk drives aren't acting fishy anymore.

Timothy R. Gaffney
433 S. Fifth St.
Miamisburg, OH 45342

Editor's note: Thanks for the tip on fish paper. We tried it on one of our systems. The only problem we're having is clearing out the cats when we want to use the system.

Z80MR Additions

Here are some additions and corrections to the documentation for the Z80MR macro assembler (Micro C User Disk #K25).

When naming labels and macros, the characters %, _, $, !, @, and ? count as letters. Numerals are allowed also except as the first character. Z80MR converts all characters outside of quoted strings to upper case.

COND is a synonym for IF.

ENDC is a synonym for ENDIF.

PUBLIC declares a list of symbols to be exported.

EXTRN declares a list of symbols to be imported. External symbols can be used in simple expressions. Misuse of external symbols results in an "E" error code.

ASEG makes a segment absolute (non-relocatable).

CSEG and DSEG both make a segment relocatable. They cause different record-type codes to be placed in the record headings of the OBJ file.

Use of PUBLIC, EXTRN, CSEG, or DSEG causes output of an OBJ file rather than a HEX file. OBJ files are arranged in records similar to the Intel hex format, but in binary form rather than ASCII.

NAME places a name of up to six characters into the OBJ file.

An expression in parentheses can be used as an operand. This expression may include conditional statements which give 0 for a false condition and -1 for true. Conditional statements treat any non-zero value as true.

Z80MR has some problems with:

```
LD HL,(N1+N2)/N3
```

Rewrite this line as:

```
LD HL,+(N1+N2)/N3
```

In this context, the leading "+" is not a unary operator. Instead it assumes a default first operand of 0. So:

```
LD A,-5 -> LD A,0-5
```

A single external label can be used in an expression as long as it adds its value to the expression.

```
LD HL,EXTRN1-N2 ;ok
LD HL,N2-EXTRN1 ;no good
```

An expression containing an external label may be used as an argument for either .LOW. or .HIGH.

Options for LIST and NLIST:

- B: List all relocatable symbols in OBJ file. All PUBLIC symbols are included whether or not B is set.
- Z: Fix this option by changing the byte at 2398h from 2Dh to 2Ch.
- X: Does not exist.
- G: Does not exist.
- O: Output code to OBJ or HEX file. Not related to printing code in the PRN file.

Specifying a non-existing function does not cause an error message or an effect of any kind.

Single character error codes:

- B: Out of range in destination for JR or DJNZ.
- C: IF..ENDIF imbalance.
- K: Pre-defined operand (register name or flag condition mnemonic) used in an expression.
- M: Macro formatting error.
- N: Does not exist. Nesting overflow results in an explicit message.
- S: Syntax error in expression.
- T: Does not exist. Full symbol table causes "Not enough memory" message.
- Z: Z80 opcode encountered while LIST Z option active. (See option Z above for bug fix.)

Declaring an EXTRN symbol to be PUBLIC is an error but goes unreported.

Neil Koozer
Kellogg Star Route Box 125
Oakland, OR 97462

CP/M™ Software: Saving The Best for Last

Plus: The Greatest **ConIX**™ Giveaway

CP/M: Some people love it, others love to hate it, but most still use it. Its users complain that most software companies have abandoned it. Very true, yet we haven't*!* We've been selling the ConIX software line for many years; we developed it, we market it, and we support it - completely*!* ***What?! You haven't tried it?*** Saving the best for last, eh*?* *Don't wait!* Support *your* CP/M software company - try ConIX for *as low as $10!* What's more, you could even get lucky and receive *your entire order* ***FREE!*** See details below.

ConIX™ Operating System

An extensive upgrade for 48K+ CP/M 2.2/3.0 and equivalent systems. Provides professional capabilities with blinding speed, as often found on high-end UNIX™ machines. Installs easily in just minutes to add over 100 new built-in commands and features while maintaining 100% compatibility with all your existing software*!* Includes I/O redirection, aliases, improved user area access, auto-searching, PF Keys, Screen Paging, Print Spooler, Archiver, New SysCalls, ... Eliminates many points of user frustration with CP/M. Uses only 1/2K TPA, 0-27K disk minimum.

Included *FREE* with commented source is the Pull-Down Menu System, a user-friendly interface to ConIX. Loads with a single keystroke*!*

ConIX is the greatest, most powerful 8-bit upgrade, with speed and capabilities that are so incredible it's bringing users *back* to CP/M*!*

ConIX™ Programming System

A structured programming language for ConIX extends CP/M SUBMIT capability. Adds conditionals, loops, subroutines, labels, nesting, interrupt processing, error traps, and debugging facilities. Design intricate menu systems and command-automation shells. Also includes a special source-code "compiler" that provides string and numeric variables. An absolute *must* for CP/M power-users and developers*!*

ConIX™ Library Vol. I XCC Utilities

Over 25 utilities for ConIX written in the shell language, including hierarchical directories with overlay - adds pathname capability to existing software, interactive debugger, move/copy/link multiple files, print files with pagination, review disk files for deletion, unerase disk with stats, full-screen TYPE, and more. Source code included*!*

ConIX™ Shareware Version

A new Shareware version of the ConIX O.S. includes our regular distribution software less the Archiver, On-Line Manual, Menu source code, and some satellite utilities. ConIX Shareware is available through CHI for just the cost of the diskette and shipping, or on-line via many popular bulletin board systems. Register by purchasing regular ConIX.

ConIX™ Disk Manual Version

To reduce the cost for those who want to purchase only the ConIX O.S., we are offering the complete software package with documentation provided on disk. The disk manual has each chapter stored in individual files, excluding the Chapter Summary, Chapter Reference, and Index sections that come standard in our regular typeset manuals.

The Great Giveaway: Every 100th ConIX Order *FREE!*

That's right*!* Every 100th order processed by our computer will be shipped with a Credit Certificate for the total purchase price or $100, whichever is lower. This credit may be used toward a future purchase from CHI, or may be redeemed for cash within ninety (90) days of receipt. Your odds are an incredible *1 in 100!*

Offer applies only to private individuals and non-profit institutions ordering directly from CHI. Orders placed by PO or purchased for commercial use are not eligible. To enter, certify eligibility by signing order form.

Product Trademarks - CP/M: Digital Research Inc., ConIX: Computer Helper Industries Inc., UNIX: AT&T Bell Labs.

Name: ____________________
Company: ____________________
Address: ____________________
____
Tel. #, Hours: ____________________

End-User Software Licenses:		
ConIX Operating System	$ 29.95	$____
Disk Manual Version	$ 19.95	$____
Limited Shareware Version	$ 0.00	$____
Printed Manual Only	$ 9.95	$____
ConIX Programming System	$ 29.95	$____
Printed Manual Only	$ 9.95	$____
ConIX Library I XCC Utilities	$ 24.95	$____
Printed Manual Only	$ 9.95	$____
All ConIX Packages Above	$ 69.95	$____

Computer Brand:
____

Software Distribution Disk Format: *		
8" SSSD Standard	$ 5.00	$____
5-1/4" DSDD 48 TPI Soft Sector	$ 5.00	$____
5-1/4" SSDD 48 TPI Soft Sector	$ 6.00	$____
5-1/4" _S_D__TPI____ Sector	$ 10.00	$____

* Add only one format charge per order.

Format Brand:
____

Shipping Information:		
UPS Ground USA	$ 4.50	$____
US Mail USA (For P.O. Boxes)	$ 6.50	$____
Air Mail Canada	$ 9.50	$____
Air Mail Foreign	$ 12.50	$____
Subtotal:		$____
Sales Tax (N.Y. Residents Only):	x____%	$____
Total (Thank You!):		$____

I Certify ConIX Giveaway Eligibility:

X____________________

POs and UPS CODs accepted by phone only. Sorry, credit card payment unavailable. Personal checks require 10 days to clear. Non-USA orders must be prepaid by bank draft in US $. Delivery in 2-4 weeks.

Computer Helper Industries Inc.
Post Office Box 680
Parkchester Station, N.Y. 10462
(212) 652-1786 9AM-5PM M-F

By Jim & Jack Dennon
microMethods, Inc.
P.O. Box G
Warrenton, OR 97146

ZRP/M2: Anticipating The Z280

A New Z80 Emulator For PC/XT/AT Clones

The new Z280 has definitely put some spark back into the CP/M world, but Zilog has withdrawn the chip. What to do? Read on as the Dennons describe their attempts to emulate the Z80 and the Z280 on a clone. (Where else?)

If you boot your V20-based PC/XT compatible with PC RP/M2 (it emulates CP/M on a clone), you can run most CP/M programs. The ones that won't run usually contain Z80 instructions, and the V20 supports only the 8080 subset.

Hardware & Software Solutions

Our first solution for this problem was a hardware-oriented one based on Decmation's Blue Thunder card. With this board plugged into the PC bus, booting the Blue Thunder version of PC RP/M2 brings up a Z80 CP/M 2.2 compatible system.

Since a Z80 is executing the code, the system is fast; and for computationally intensive applications, it remains the best solution. The disadvantages are: it's somewhat expensive, and it occupies an expansion slot.

We were thinking about releasing a software Z80 emulator-equipped version of PC RP/M2 when Zilog announced the Z280. The Z280 executes all the Z80 instructions and more. And some of its more interesting 16-bit instructions map closely into 8088 instructions.

So we decided to create a Z280 emulator and surround it with a new version of PC RP/M2. The result is a system we call ZRP/M2, which will boot on any PC/XT/AT clone (a clone with lots of memory).

Decmation designed the emulator. There's no instruction decoder so the emulator simply uses the operation code as the high byte of the 16-bit offset of the instruction processor. We invented a multi-segment version of this scheme for the Z280. This architecture achieves speed at the expense of memory space.

Unlike the Z80, which has an 8-bit I/O space, the Z280 has a 24-bit I/O space that maps easily down to the 16-bit I/O space of the 8088. We simply set the high order eight bits of the 24-bit port value to zero. On the Z280, the high order eight bits are provided by the I/O Page register. The Z280 clears the I/O Page register at reset.

On-chip peripheral devices - a UART and a DMA controller, for example - occupy nonzero I/O Page locations. Our present emulator supports only I/O Page zero. That is, we map Z280 I/O page zero directly onto the 16-bit 8088 I/O space.

On the IBM PC, the COM1 serial data register (for example) is located at I/O location 03F8H, so the code in Figure 1 is sufficient to read a character from the COM1 serial port.

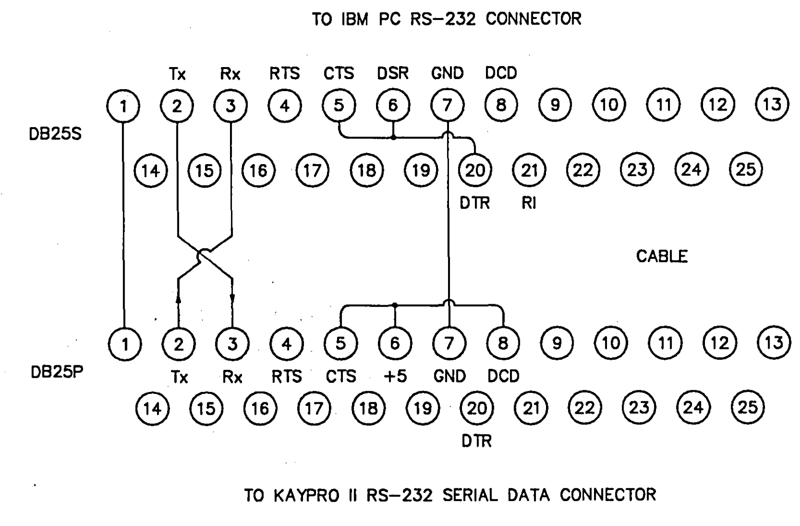
I/O

Using this new I/O capability, we've written a ZRP/M2 driver for Doug

Figure 1 - Fetching A Character From A Port

```
;
;       GSI -- Get COM1 serial character.
;       Exit    A = char
;
01F803  GSI:    LXI     B,03F8H
ED78            IN      A,BC
C9              RET
```

Figure 2 - Kaypro To PC Null Modem Cable


TO IBM PC RS-232 CONNECTOR
Tx
Rx
RTS
CTS
DSR
GND
DCD
DB25S
DTR
RI
CABLE
DB25P
Tx
Rx
RTS
CTS
+5
GND
DCD
DTR
TO KAYPRO II RS-232 SERIAL DATA CONNECTOR

ZRP/M™ creates

Z280®

CP/M©2.2 compatible

IBM PC

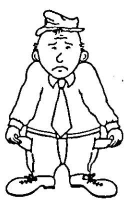
ZRP/M is an **operating system** combined with a Z280 emulator. Either standalone or with DOS present, ZRP/M provides the solid base of a genuine operating system reliably distinct from the facade created by an MSDOS interface. **All 2.2 system and CBIOS calls are supported,** 56.5k TPA, file date and time stamping, fast virtual disk, iobyte redirection, terminal emulation, color console display, auto relog, COM path, SAVE anywhere, single key phrase recall, built-in access to DOS drives. SETDISK redefines a drive to any of over 80 CP/M formats. System disk with manual $129. Shipping $5 ($10 nonUS)

Micro Methods, Inc.

118 SW First St. - Box G
Warrenton, OR 97146
(503)861-1765

POOR MAN'S NETWORK

Now includes FREE 2-user Database!



Poor Man's Network is a true Local Area Network for CP/M and Z-system computers. Uses RS-232C or bidirectional parallel ports; no extra hardware to buy. Share floppy disks, hard disks, RAM disks, and printers. Remote drives are accessed just by specifying a network drive letter. Send screen messages with a single key-stroke.

Not just a terminal program, **Poor Man's Network** is a loadable BIOS extension (takes 7K) that provides networking capabilities at minimum cost. Works with virtually all CP/M 2.2 (with or without ZCPRx) and Z-system computers, including Kaypro, Televideo, Columbia, North Star, Apple II, Xerox, Big Board I & II, Pied Piper, S-100, Ampro, Nabu, Bondwell, Micro Mint, and others. Works with IBM PC when used with Micro Methods' RP/M or Z-RP/M.

Special BDOS calls let you implement electronic messaging between computers. Free sample database, complete with Modula 2 source, illustrates how to write your own network programs and implement record locking. Write your own multi-user game or database, in any language that can make BDOS calls.

Best of all is the price: still only **$69!**

Many disk formats available, including 8" SSSD, Kaypro, IBM, Televideo, Apple, North Star, etc. Each disk contains Assembler source and hex drivers for all supported computers. 50 page manual included.

How to Order: We accept Visa, Mastercard, or money orders. No purchase orders or CODs. No personal cheques from outside Canada, please, as they take up to six weeks to clear. Phone orders accepted 10 AM - 5 PM, Tues - Sat. Canadian residents please pay in Canadian dollars; others in US dollars. Price includes disk, manual, and first class postage. Ontario residents add provincial sales tax. If you can't download to your other computer, we can provide extra disks for $10 each. For overnight shipping by Federal Express to most major cities in North America, add $20.

ANDERSON TECHNO-PRODUCTS INC.
947 Richmond Road, Dept C
Ottawa, Ontario K2B 6R1
Telephone **613-722-0690** for more information or to order.

Registered trademarks: CP/M: Digital Research, ZCPR: R. Conn, Z-system: Echelon.

Anderson's latest version of Poor Man's Network. Our network operates at 9600 bps and connects a Kaypro II to an XT clone. Figure 2 is a diagram of the serial cable we use.

Our earlier Poor Man's Network driver for the V20 version of PC RP/M2 was limited to 4800 bps. The problem was that there's no 16-bit I/O directly available to the V20's 8080 emulator. To make the 16-bit I/O space accessible to 8080 programs, we created an input and output redirection scheme based on the CP/M I/O-byte at location 0003H.

Evidently, the redirection overhead limits performance. Under the Z280 emulator there's no operating system overhead associated with doing input or output to a port, so in this unusual instance the software-based emulator outruns the hardware-based emulator.

Even though most IN and OUT instructions will execute correctly under the Z280 emulator, it's necessary in nearly all cases to modify and reassemble CP/M programs that do their own I/O. For example, with the I/O Page register set to 00, and the A register set to 42H, the 8080 instruction DB 66, or IN 66H, will input to the A register, one byte from I/O location 004266H. The original contents of the A register provide bits 8 through 15 of the port address.

Zilog's corresponding rendition of the 8080 OUT instruction is even less useful. With the same original register configuration, the instruction D3 66, or OUT 66H, will write the value 42H to I/O location 004266H! Evidently, these instructions are casualties of a grander scheme.

Useful I/O instructions are the Z80 type that now take port bits 0 through 15 from register pair BC. Examples of these instructions appear in our Poor Man's Network driver for ZRP/M2. The driver will be included with Poor Man's Network from Anderson Techno-Products.

Our Z280 emulator presently has all the Z80 instructions operational and enough of the new Z280 instructions to support simple programs such as the network driver.

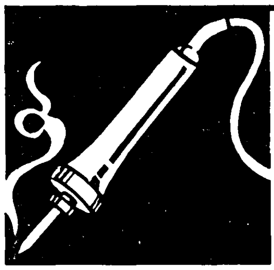
Operating speed on a 4.77 MHz PC isn't spectacular, but it's usable. On a turbo PC the system has approximately the feel of a V20 system. The structure of the emulator is such that its speed is dependent on the number of instructions implemented, so meaningful benchmarks can be run with the present system.

Unlike the Hitachi 64180, which has a multiply instruction but not a divide, the Z280 has signed and unsigned 16-bit multiply and divide instructions. These operations are important to the performance of servo systems such as those we use in our sawmill systems.

The Future

Our existing systems are based on either the 8085 or the Z80. If the Z280 actually becomes available, it should find a good market in the controls industry. The Beaverton Zilog rep says parts will be available by July. We were quoted $28 in small quantity. Hope it's real.

■■■



Technical Tips

Fixing Automatic Capitalization

I was so impressed with Mark Boyd's "Automatic Capitalization" article in issue #34 that I bought FTL Modula-2 and the Editor Toolbox. I made Mark's changes right away and dove in.

A subtle bug in his modification transforms "ReadWord" into "ReadWORD." Mark's code assumes that a symbol will consist entirely of lower-case letters. What's wrong with that assumption? It causes upper-case letters and digits (all valid symbol components) to be seen as delimiters of symbols. This means that "ReadWord" becomes two symbols, "ead" and "ord." The reserved word "ord" is then capitalized.

To fix the problem, just expand the case in the scan procedure to include upper-case letters and digits.

```
CASE ch OF

  'a'..'z','A'..'Z','0'..'9':
```

Make the same change to the initialization of symset in SCANNER.MOD, and you've solved the problem.

Bill Spees
710 W. Main St. #819
Arlington, TX 76013

More Noise

With regard to the recent Technical Tips item on high frequency noise from a monitor: Last year, Inter-Noise 86, the annual conference of the Institute of Noise Control Engineering, published an article by David S. Gaunt of British IBM.

He found that there are three noise sources: the switching power supply, the analog CRT card, and the CRT yoke. The analog card generated the most noise, while the CRT yoke generated the least.

The worst offender on the analog card was the line output transformer along with a power transistor. *(Editor's note: He's no doubt referring to the horizontal sweep transformer and its driver transistor.)*

The main transformer on the switching power supply also generated a lot of noise. After covering these components with acoustic chimneys (open, foam-lined boxes), isolating them from the card, and attaching a plastic cover to the non-component side of the card, noise was reduced about 20 percent.

Mike Fern
Box 1105
Covina, CA 91722

Fix For Turbo C

I agree with the comments made in the reviews of Borland's Turbo C in the Not-August issue (#37), but wish to pass on the details of a problem I had with it in the hope of saving others from falling into the trap.

The atan2(y,x) function must be of much more use to engineers than programmers as it seems to be poorly coded, and untested, in several compilers. Turbo C is one of them.

I got my Turbo C early in June and found that this function failed for zero values of x. Microsoft C, Desmet without 8087, and Greenhills C for the DSI-32 handle it, while Turbo C, Desmet for the 8087 (at least the version I have) and the C compiler in UNIX version 3 do not.

I sent a note with the registration card describing how the function would hang up the system and got a prompt acknowledgment from Borland. A few days later I received a complete replacement set of disks (still labeled Version 1.0) with the problem fixed. Could not ask for better support.

Figure 1 has been useful for exercising the function in various compilers, showing this and other problems.

John S. Innes
120 Macpherson St.
Cremorne, NSW 2090
Australia

Figure 1 - Program to Test ATAN2 Function

```
#include <math.h>
#include <stdio.h>

main()
  {
  double atan2(), x, y;
  int i;
  static double v[8] = {0, 1.0, 1.0, 1.0, 0, -1.0, -1.0, -1.0};

  for(i = 0; i < 8; ++i)
  {
  x = v[(i+2)%8];
  y = v[i];
  printf("%d times PI/4 = %.3f\n", i, atan2(y, x));
  }
}
```

BEAMED IN DIRECT FROM DEEP SPACE
WARP SPEED PROCESSING

The Computer Dream Come True, Breathe Fusion! Here's Technology that's at the Tip of the Cutting Edge!

INMOS™ TRANSPUTER MODULES

Interface Card installs into your PC (or QL). Low Cost TDS: 7.5 MIPS-Multiprocessing up to 60 MIPS! 1MB Hi-Speed Dram per Module. Cycle configuration linking, fully socketed 95mm x 75mm size.

Wide range of Development Software: OCCAM 1 (PC Cross Compiler), Assembler, MPE Modular Forth. From Microway: OCCAM 2, Fortran, Pascal, C, TDS Environment & board units.

The DOOR is OPEN into the Future of Computing!
Contact us for a Cutting Edge TEK PAK of Bit by Bit Details . . . .

ULTRATEK

Quantum Computing

Box 1280, Dover, NJ 07801

Call: (201) 328-8846

QLine BBS: 328-2919

Technical Info & Assistance - -
Telex: 9102500026
Compuserve ID # 76625,2214

EVER PROGRAM ON A SILVER PLATTER??

How much would you expect to pay for a 32 bit MC 68000 computer that's a mainframe condensed down into a keyboard? Thousands, right? No, just several hundred! But that's actually our price!!! The most powerful computer money can ever buy is now the most inexpensive computer money can buy!!! So don't buy the name! Buy the power!! The power is **NOT** in the name!

If **you** had the opportunity to work in a research laboratory amongst Machine Code ROM Designers, VAX & UNIX wizards, designing an MC 68000 architecture computer that's 2nd to none. . .

Just what would you come up with?? And what would you finally call it??

Well It's Already Been Done!!
They Called It The QL For The Quantum Leap That It Is!!

Absolutely a Quantum Leap beyond what you know & use - and it's truly like Programming on a Silver Platter!!

The QL Desktop Minicomputer: Designed by SRL Labs, manufactured by Samsung. An absolute Quantum Leap beyond all the rest! The phenomenal open architecture QDOS is a dimension beyond. It works from the protected mode, has Virtual Memory RAM, Multitasking Job Control, Multiuser Networking. It'll Cache Files into unused Memory and create/delete Directories Automatically! Even allows File Names up to 36 characters long! Everything is built into ROM here: QDOS, Networking, Windowing, & 32 Bit Turbo SuperBasic, all in a totally concurrent non-destructive environment. Unlimited quantities & lengths allowed with: Variables, Program Lines, CONsoles & Buffers. Dynamic non-destructive virtual RAM Disking & Networking buffers too! Even a System Variables Brain Page Screen! Built-in DCE & DTE Serial Ports. 68020 and Transputer parallel processing upgrades are now available too.

Language Environments:
Metacomco's "C", LISP, BCPL, 68000 Assembler, APL, Development Kits. Prospero's Pro Pascal & Pro Fortran 77. Digital Precision's Forth-83. QJUMP's 65C02 or 8048 Cross Assembly ROMs. Everything generates native 68000 Binary Code. ROM Firmware & Software Package is now available which will even bring it up in CPM!

Imagine working with a 32 bit Turbo SuperBasic that out-structures Turbo Pascal, in an interpretor always present with QDOS, all concurrently running in a built-in UNIX-like multitasking job controlled environment with access to 360 fully channeled windows, devices & files by EACH job! 3 Major Compilers exist for the SuperBasic source alone! TURBO, SUPERCHARGE, QLIBERATOR! The compiled SuperBasic code or ANY other language will multitask and control with QDOS and SuperBasic. The list of ALL the Superior Features would fill this entire publication!

The QL comes bundled WITH PSION Integrated Word Processor, Spreadsheet, Database and Presentation Graphics Programs. PLUS: Our FREEWARE Demos & Utilities with all purchases! Add $12 shipping and handling.

INMOS is registered to Inmos. Where Innovation Reigns Supreme!

CULTURE CORNER

No Headline

By

THIS PAGE INTENTIONALLY LEFT BLANK

Editor's note: This blank page was sent to us by Nothing On Pages (NOP), a memberless society whose unpublished goal is the preservation of the blank page.

After receiving this very nice blank page, I visited the society (based in a community which will, no doubt, remain nameless) but failed to get more than a blank stare from the missing receptionist. However, after a little poking around, I located a Mr. M. T. Page, a forgettable little man with no title.

"It used to be that we didn't do much here at NOP. After all, it wasn't until recently that anyone realized the significance of the blank page.

"Printers used to feel that a blank page lacked something. It wasn't until manuals became commonplace and people began seeing our motto: 'This Page Intentionally Left Blank' that public awareness picked up. Otherwise, the craft of printing blank pages (a craft which, I must say, predates Gutenberg) might have died out.

"Of course now people realize the contribution that blank pages make."

"Like?"

"Well, there's a special difference that makes a blank page stand out. Imagine a full-page ad which is entirely blank. Think of the impact.

"And, imagine that same blank page in full color. Wow!

"Also, if a subject's very important, it makes perfect sense to print nothing about it at all (on as many pages as seem appropriate). That way there's no chance of misinterpretation."

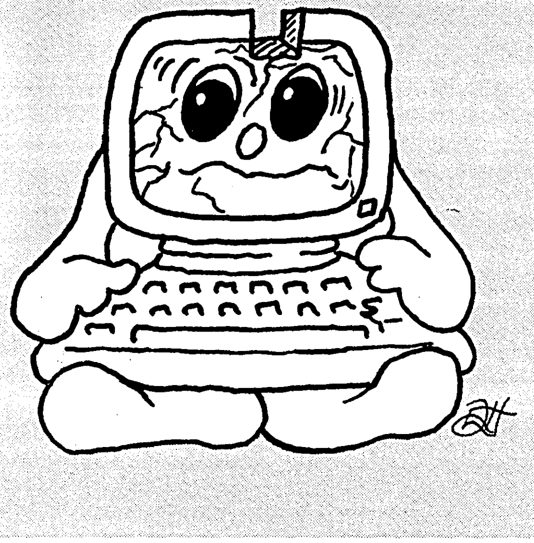
"Any Problems?"

"Our biggest problem has been readers. They expect something to read everywhere they look - bank buildings, billboards, hamburger wrappers, blimps, computer screens... Otherwise they're bored. Stick these people out in the woods and within minutes they're carving their names into trees. It's no doubt a hangover of the early literacy drives.

"On the other hand, curiously enough, we've had no problems at all with business executives."

For excellent coverage of the blank page, its past, its present, and its future, get:

Between The Blank Sheets
By Wood E. Pulp
Director Of Sales
LeBlanc Paper Company
300 pps.



Is There A Gap In Your Info?

Fill in your back issues of Micro C today

ISSUE #1 (8/81)
Power Supply
RAM Protection
Video Wiggle
½ PFM.PRN
16 pages

ISSUE #2 (10/81)
Parallel Print Driver
Drive Motor Control
Shugart Jumpers
Program Storage Above PFM
½ PFM.PRN
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
Cheap RAM Disk
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
5MHz 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
Interface to Electronic Typewriter
BBI Video Size
Video Jitter Fix
Slicer Column Begins
Kaypro Color Graphics Review
48 pages

ISSUE #15 (12/83)
Screen Dump Listing
Fixing Serial Ports
Playing Adventure
SBASIC Column Begins
Upgrading Kaypro II to 4
Upgrading Kaypro 4 to 8
48 pages

ISSUE #16 (2/84)
Xerox 820 Column Restarts
BBI Double Density
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 Driven Serial Printer
Low Cost EPROM Eraser
Smart Video Controller
Review: MicroSphere RAM Disk
Future Tense Begins
86 pages

ISSUE #24 (6/85)
C'ing Into Turbo Pascal
8" Drives On the Kaypro
48 Lines On a BBI
68000 Versus 80x86
Soldering: The First Steps
88 pages

ISSUE #25 (8/85)
Why I Wrote A Debugger
The 32-Bit Super Chips
Programming The 32032
Modula II
RS-232C: The Interface
104 pages

ISSUE #26 (10/85)
Inside ZCPR3
Two Megabytes On DSI-32
SOG IV
The Future Of Computing
MS-DOS In The Public Domain
Graphics In Turbo Pascal
104 pages

Issue #27 (12/85)
SOLD OUT

ISSUE #28 (2/86)
Pascal Runoff Winners
Rescuing Lost Text From Memory
Introduction To Modula-2
First Look At Amiga
Inside The PC
104 pages

ISSUE #29 (4/86)
Speeding Up Your XT
Importing Systems From Taiwan
Prototyping In C
C Interpreters Reviewed
Benchmarking The PCs
104 pages

ISSUE #30 (6/86)
PROLOG On The PC
Expert Systems
Logic Programming
Building Your Own Logic Analyzer
256 K RAM For Your 83 Kaypro
PC-DOS For Non-Clones
104 pages

ISSUE #31 (8/86)
RAM Resident PC Speedup
Practical Programming In Modula-2
Unblinking The PC's Blinkin' Cursor
Game Theory In PROLOG and C
104 pages

ISSUE #32 (10/86)
Public Domain 32000:
Hardware and Software
Writing A Printer Driver For MS-DOS
Recover A Directory By
Reading & Writing Disk Sectors
96 pages

ISSUE #33 (12/86)
Controlling Stepper Motors
From Your PC
Introduction To Fractals
The Secrets Of MS-DOS, From
Boots To Device Drivers
Poking About In The System
With Turbo Pascal
96 pages

ISSUE #34 (2/87)
Designing With The 80386
Build A Simple Oscilloscope
A Cheap 68000 Operating System
A Concurrent Operating System
Recovering Directories And FATs
96 pages

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

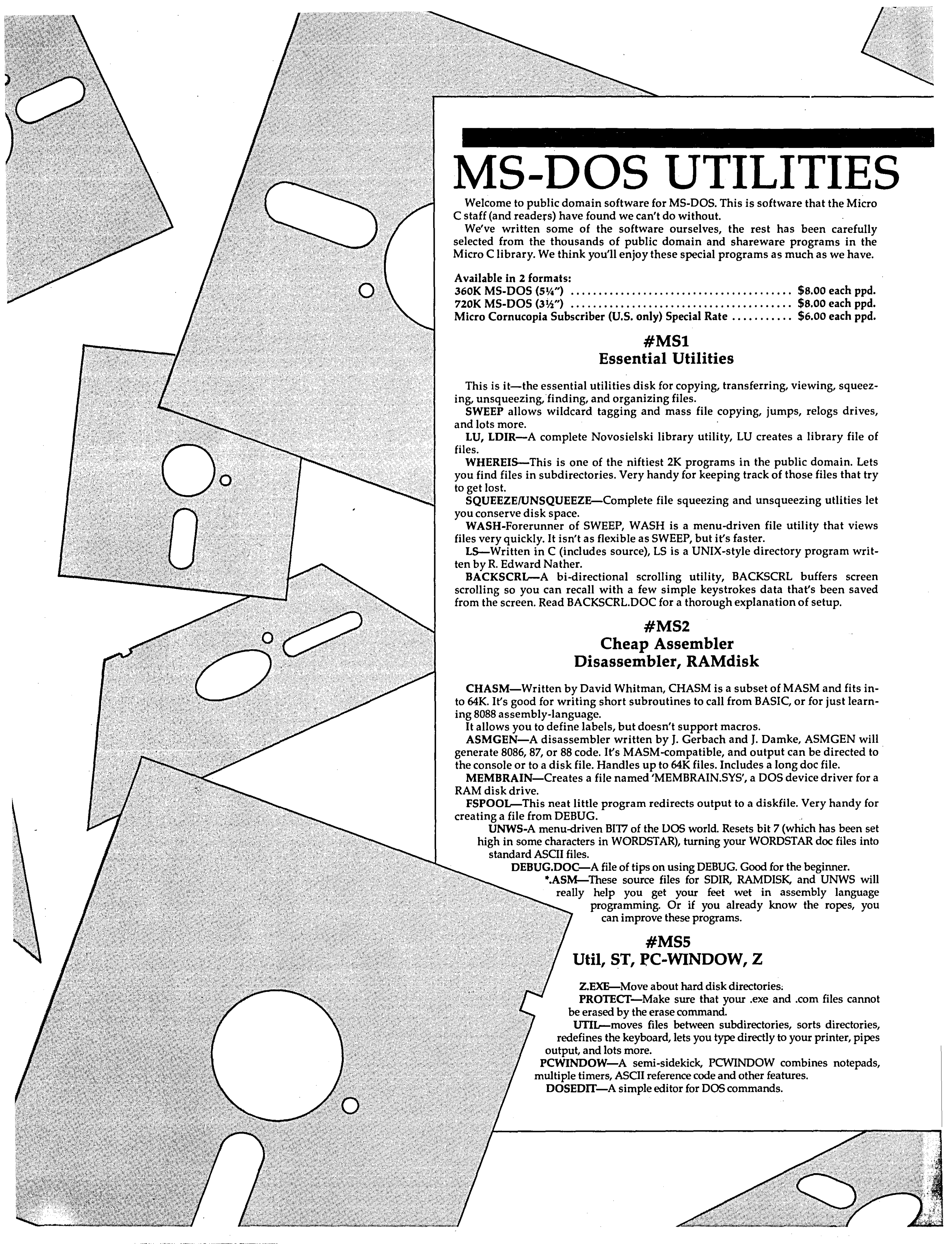
ISSUE #36 (6/87)
Build A Midi Interface For Your PC
Designing A Database, Part 2
Interrupts On The PC
Hacker's View Of MS-DOS Vs 3.X
Digital To Analog Conversion, A
Designer's View
96 pages

ISSUE #37 (9/87)
Desktop Publishing On A PC
Build Your Own Hi-Res Graphics
Scanner For $6.00, Part 1
Designing A Database, Part 3
Controlling AC Power From Your PC
Expanded Memory On The PC/XT/AT
Uninterruptible Power Supply For
RAMdisks
96 pages

BACK ISSUES OF MICRO C

U.S., Canada & Mexico
Issues #1-34 .... $3.00 each ppd.
Issues #35 through current issue ...... $3.95 each ppd.

Foreign (air mail)
Issues #1 through current issue ...... $5.00 each ppd.



MS-DOS UTILITIES

Welcome to public domain software for MS-DOS. This is software that the Micro C staff (and readers) have found we can't do without.

We've written some of the software ourselves, the rest has been carefully selected from the thousands of public domain and shareware programs in the Micro C library. We think you'll enjoy these special programs as much as we have.

Available in 2 formats:

360K MS-DOS (5¼")	**$8.00 each ppd.**
720K MS-DOS (3½")	**$8.00 each ppd.**
Micro Cornucopia Subscriber (U.S. only) Special Rate	**$6.00 each ppd.**

#MS1
Essential Utilities

This is it—the essential utilities disk for copying, transferring, viewing, squeezing, unsqueezing, finding, and organizing files.

SWEEP allows wildcard tagging and mass file copying, jumps, relogs drives, and lots more.

LU, LDIR—A complete Novosielski library utility, LU creates a library file of files.

WHEREIS—This is one of the niftiest 2K programs in the public domain. Lets you find files in subdirectories. Very handy for keeping track of those files that try to get lost.

SQUEEZE/UNSQUEEZE—Complete file squeezing and unsqueezing utlities let you conserve disk space.

WASH-Forerunner of SWEEP, WASH is a menu-driven file utility that views files very quickly. It isn't as flexible as SWEEP, but it's faster.

LS—Written in C (includes source), LS is a UNIX-style directory program written by R. Edward Nather.

BACKSCRL—A bi-directional scrolling utility, BACKSCRL buffers screen scrolling so you can recall with a few simple keystrokes data that's been saved from the screen. Read BACKSCRL.DOC for a thorough explanation of setup.

#MS2
Cheap Assembler Disassembler, RAMdisk

CHASM—Written by David Whitman, CHASM is a subset of MASM and fits into 64K. It's good for writing short subroutines to call from BASIC, or for just learning 8088 assembly-language.

It allows you to define labels, but doesn't support macros.

ASMGEN—A disassembler written by J. Gerbach and J. Damke, ASMGEN will generate 8086, 87, or 88 code. It's MASM-compatible, and output can be directed to the console or to a disk file. Handles up to 64K files. Includes a long doc file.

MEMBRAIN—Creates a file named 'MEMBRAIN.SYS', a DOS device driver for a RAM disk drive.

FSPOOL—This neat little program redirects output to a diskfile. Very handy for creating a file from DEBUG.

UNWS-A menu-driven BIT7 of the DOS world. Resets bit 7 (which has been set high in some characters in WORDSTAR), turning your WORDSTAR doc files into standard ASCII files.

DEBUG.DOC—A file of tips on using DEBUG. Good for the beginner.

***.ASM**—These source files for SDIR, RAMDISK, and UNWS will really help you get your feet wet in assembly language programming. Or if you already know the ropes, you can improve these programs.

#MS5
Util, ST, PC-WINDOW, Z

Z.EXE—Move about hard disk directories.

PROTECT—Make sure that your .exe and .com files cannot be erased by the erase command.

UTIL—moves files between subdirectories, sorts directories, redefines the keyboard, lets you type directly to your printer, pipes output, and lots more.

PCWINDOW—A semi-sidekick, PCWINDOW combines notepads, multiple timers, ASCII reference code and other features.

DOSEDIT—A simple editor for DOS commands.

#MS15
Utilities

Here are utilities to make your life more efficient.

DESKMATE is a 'Sidekick' lookalike with notepad, calculator, calendar, and access to DOS commands.

EASY-ZAP, a disk inspector, will allow you to examine and modify sectors. It works on hard disks as well as floppies.

UNERASE is the essential utility to save you from your own recklessness. If you've unintentionally erased a file, UNERASE will undo the damage. Handy.

#MS22
Dynamite Utilities

We've included some genuine gems on this disk.

V20-80—CP/M emulator software which enables IBM PC compatibles (i.e. personal clones) equipped with the NEC V20 CPU (See Micro C Issue No. 29 for details) to run 8080-coded CP/M programs.

LIST—A dynamite TYPE lookalike (the best we've used), with line up, line down, page up, and page down in 16 variable colors.

SPEEDUP speeds up and quiets your drive by changing the step rate from 8 milliseconds to 4 milliseconds.

TURBO HELP—A memory resident help facility to help you learn (and use) TURBO Pascal. It's ready at a keystroke in an attractive window.

INLINER—Translates your assembler mnemonics into TURBO Pascal inline code. Written in Turbo Pascal; includes source.

LASTCOM—TURBO Resident program to save your last 10 MS-DOS commands. Includes source.

The **SECRET** Group (MD, CD, and RD)—lets you make, remove and find secret files.

#MS25
Ultra Utilities

The three Ultra Utilities programs will allow you to map disks, unerase files, format non-standard disk, interrogate sectors, and much, much more.

This is a very useful set of utilities (a poor man's NORTON). Many hours of work and frustration can be avoided by learning them, so have at it.

#MS27
System Primer

We think this disk will satisfy at least some of your curiosity about MS-DOS systems programming. We've included lots of assembler source code, so dig in.

SCAV finds and marks bad blocks on both floppies and hard disks.

WHEREIS finds files anywhere within a directory structure.

DIAGS—Special serial, parallel, and video diagnostics for the PC. Use this excellent program to explore your system.

ASYNC—Loadable asynchronous device driver for MS-DOS.

LPTX intercepts BIOS interrupt 17, the line printer interrupt. It redirects the output of LPT1, LPT2, or LPT3 to a disk file. All three may be active at the same time.

DOS1, ROLLDOS1 & 2, DRIVER & DRIVER1—Stop wondering how device drivers really work and explore these tutorials. Good examples of character device drivers and de-bugging techniques.

STUFIT stuffs your least used files into the inner tracks of the disk. This frees the outer tracks for work space and speeds access times considerably.

#MS36
General Utilities

BATMAKER helps create .BAT files. Perfect when using FIND on all .TXT files, for instance. Very handy.

BWVID lets you see what is happening on the screen when you have a color graphics card (CGA) and a monochrome monitor.

CED is called a Command line EDitor but it's far, far more than that. Includes macro definitions, control of DEBUG, repeating and editing of previous commands, etc.

DEBUG.DOC is a simple but very handy quick reference guide to DEBUG.

EXPAND and SHRINK are detab and entab utilities.

PC-STAT—Reports system information—memory available, drive status, etc.

PC-TEST is similar to Norton's speed test, but its test takes longer and it doesn't report such wildly optimistic speed figures.

POPALARM is really neat. It's a memory resident alarm clock that reminds you to do what you'd otherwise prefer to forget.

RECALL remembers the last 50 DOS commands. Commands may be edited and/or reexecuted.

REMIND—This is a daily black-book that stores its data on disk.

SCR*.*—Utilities for creating batch files which incorporate screen images. This is a great extension to MS-DOS batch capabilities.

FILTERS—The remaining files are classics from the Software Tools book. One of the real attractions of these filters is that they come with assembly language source.

#MS37
Disk Utilities

COVER prints out directories in compressed format to be pasted on floppies.

CRC67—Finally, an MS-DOS cyclic redundancy checker (CRC) that works (Fast!). Checks CRC values for files against a previously recorded list of CRC's.

DISKORAY checks floppy rotation speed and allows stepping of the head.

DISKPARK parks the heads of all hard disks in your system using the innermost track.

DISKWIPE—Be careful. This completely erases a disk, including the formatting.

FDATE allows editing of the time and date stamp on DOS files.

FILES—A very complete directory program.

MOVE2—Intelligent COPY routine.

REFRESH rerecords data on a disk. It does 12 retries on reads and 2 on writes so it may be able to recover those "bad" sectors.

SDIR—Version 5.0 of the super directory program.

SST—Just what every busy hard disk needs. SST reorganizes files into contiguous sectors on the disk. This really speeds up disk accesses.

TIMEPARK parks the heads on a running hard disk after a user specified amount of time without accesses.

WD*.*—Everything you always wanted to know about Western Digital's WD1002S-WX2 hard disk controller. Also information on optimizing its performance with the Seagate ST225 drive.

WHEREII searches for one or more files through all the directories on a specified drive. Supports wild-cards.

For Technical Help:
(503) 382-8048 (9 a.m.—noon, PST, M-F).

Order today from:

MICRO CORNUCOPIA

VISA P.O. Box 223 • Bend, Oregon 97709 MasterCard

Phone orders: (503) 382-5060

TIDBITS

AI, Statistics, & Benchmarking The Turbos

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Gary reports on his favorite subject, AI. Then when you add statistics, the results can be highly interesting, if not highly calculating.

In April, 1985, AT&T Bell Laboratories sponsored a workshop on artificial intelligence and statistics which attracted the leaders of this new interdisciplinary field.

Artificial Intelligence & Statistics is the 400-plus page collection of 17 papers presented at the workshop on subjects editor William Gale (of AT&T Bell Laboratories) neatly divides into six categories -

- uncertainty propagation
- clustering and learning
- expert systems
- environments for supporting statistical strategy
- knowledge acquisition
- strategy

In his introduction to the book, Gale focuses on one of the central problems in statistics:

"We need to make statistical strategy available to more people to prevent misuse of statistical packages. Current packages provide excellent numerical processing, but the user is responsible for determining whether the processing is appropriate, and what the results mean. Statistical strategy is simply missing from [current statistical] packages."

Although few programmers have attempted to develop this kind of statistical strategy on a commercial level, I believe this is an ideal opportunity for AI programmers to show their stuff.

We can describe statistical strategies in flow charts, in trees, and with rules and conditions. If you're hazy about these concepts, glance back to "Expert System" in *Micro C* issue #35 and the PC Diagnosis problem.

Using a simple inference engine like the one in issue #35, we can program an expert statistical system by simply changing the knowledge domain (or rules) to cover statistical strategy instead of PC problems.

So, a statistical rule might take the form:

```
Statistical test(Kruskal-Wallis) :-
    We're looking for differences,
    The differences are between means,
    We have more than 2 samples,
    Our distribution isn't normal (i.e.,
    non-parametric).
```

If you're interested in knowing more about the state of statistical research, check out this collection of papers published by Addison-Wesley.

If you want to know more about applying expert system technology to this and other problems, write or call me in Davis.

Benchmarking The Turbos

For years now (six to be exact), *Micro C*'ers have enthusiastically expressed their interest in benchmarking by consistently sending us the most mail after we've tested the speed of a group of computers or compilers.

And over the years, while it's been customary for the reviewers of most mainstream magazines and journals to use a standard test like the sieve (which has in turn led to the now old joke that some compilers are optimized for the sieve), it's been just as customary for *Micro C* reviewers to use nonstandard tests.

Actually, we don't select nonstandard tests to be ornery, but to be practical and to have fun. So don't get me wrong - I definitely support and feel that standard tests are important (perhaps *the* most important tests). But, as usual, I've been having some fun solving a practical problem and have come up with some, well, interesting results from some of my recent benchmarking.

Here's the story.

I've been writing a statistical expert system (along with Mollie Messimer at the University of Virginia) with "micro einstein," an expert development system from Acquired Intelligence (my company), and I needed to write some *really fast* statistical functions.

Although the mean is undoubtedly the most commonly used statistic, it doesn't even begin to tell the complete story about a sample. Radically different distributions of data points might have identical means.

I wrote "micro einstein" in PROLOG, so I felt it logical to try to write my statistical functions in PROLOG as well. I realized the suspicious logic that accompanied my choice (PROLOG has been described by one of its designers as a "non-numerical programming language"), but I was feeling smart (little did I know) and wanted to see how long it would take for logic and human fallacy to become totally entangled.

I wrote the little piece of code in Figure 1 to find the mean, variance, and standard deviation of a file of data

(continued next page)

Figure 1 - Statistics in PROLOG

```
DOMAINS
   real_list=real*

DATABASE
   answer(real)
   answer2(real)
   data(integer,real_list)
PREDICATES
   main
   process(real_list)
   process2(real_list)
   mean(real_list,real,integer)
   sd(real_list,real,real,integer)
CLAUSES

main:-
   write("Enter file that contains data."),
   readln(Filename),
   consult(Filename),
   data(1,List),
   process(List),       /* Pass the list to a function */
   process2(List).      /* for processing.             */

process(List):-         /* This function sets up for the mean.*/
   write("Start"),      /* Start timing. */
   N=0,
   S=0,
   mean(List,S,N),      /* Pass the list & variables to mean.*/
   answer(Mean),        /* Get the answer from storage. */
   write(Mean),nl.

process2(List):-        /* Set up to calculate variance & */
   answer(Mean),        /* sd.                            */
   sd(List,Mean,0,0),   /* Pass list and mean (which we   */
   answer2(Var),        /* use to calculate the vari & sd.*/
   write(Var),nl,       /* Get variance from storage.     */
   Sdev = sqrt(Var),    /* Find sd (sqrt of variance).    */
   write(Sdev).

mean([H|T],S,N):-       /* Find the mean recursively.     */
   Y=H+S,
   N2=N+1,
   mean(T,Y,N2).
mean([],S,N):-
   Z=S/N,
   assert(answer(Z)).   /* Store the result in a db.      */

sd([H|T],Mean,Squares,N):    /* Find vari/sd recursively.  */
   Dev = Mean - H,
   Square = Dev * Dev,
   NewSquares = Squares + Square,
   N2 = N + 1,
   sd(T,Mean,NewSquares,N2).
sd([],Mean,Squares,N2):-
   Var = Squares/(N2-1),
   assert(answer2(Var)).

GOAL
   main.
```

IS NOTHING SACRED?

Now the FULL **source code** for **TURBO Pascal** is available for the **IBM-PC!** WHAT, you are still trying to debug without source code? But why? Source Code Generators (SCG's) provide completely commented and labeled ASCII source files which can be edited and assembled and UNDERSTOOD!

SCG's are available for the following products:

__ TURBO Pascal (IBM-PC)*	$ 67.50
__ TURBO Pascal (Z-80)*	$ 45.00
__ CP/M 2.2	$ 45.00
__ CP/M 3	$ 75.00

* A fast assembler is included free!

"The darndest thing I ever did see..." Pournelle, BYTE

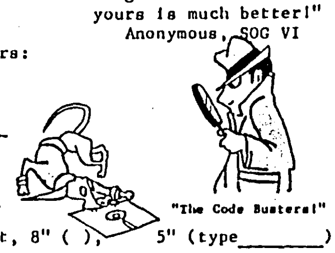
"I have seen the original source and yours is much better!" Anonymous, SOG VI

The following are general purpose disassemblers:

__ Masterful Disassembler (Z-80) ..	$ 45.00
__ UNREL (relocatable files) (8080)	$ 45.00

VISA/MC/check card# ____ ____ ____ ____ expires __/__

Shipping/Handling $ 1.50
Tax $______
Total $______



All products are fully guaranteed. Disk format, 8" ( ), 5" (type________).

C.C. SOFTWARE, 1907 ALVARADO AVE., WALNUT CREEK, CA. 94596, (415) 939-8153

CP/M and TURBO Pascal are trademarks of Digital Research & Borland Int.

(continued from page 85)

points.

For those of you a little hazy about statistics - the mean or central tendency is an impressively useful statistic used in a variety of fields: the sciences (for statistical inferences and as the basis for more sophisticated statistical inferences); in sports (batting averages, shooting percentages, betting odds, etc.); in government, in business, etc. It's also sometimes considered a descriptive statistic of location (i.e., what's the position of a sample along a given dimension representing a variable).

Although the mean is undoubtedly the most commonly used statistic, it doesn't even begin to tell the complete story about a sample. Radically different distributions of data points might have identical means.

The most common next way to add more to the story is to weight each item by its distance from the center (or mean) of the distribution.

The variance and especially the standard deviation are statistics to help us add to our description of the shape of a frequency distribution.

For more information about these and other more complex statistical tests, check out my reference to Bradley.

So, back to the code in Figure 1, written in Turbo PROLOG, which uses recursion and PROLOG's built-in linked list to calculate these simple statistics. It's very fast, calculating the mean of 7000 real numbers (data points) in 2.5 seconds on a PC-Tech X16B (8 MHz, 80186 CPU, no math co-processor).

I thought this was pretty fast, but I wasn't sure how fast, so I decided to write a real-processor in C (see Figure 2). After trying different approaches to handling the data, I decided that arrays were fastest. And to keep things simple, I had the program generate a list of the first 7000 integers and store them in the array for processing. I started timing the calculation after the array had been passed to the stat function (notice "start").

The results were surprising. It took Turbo C 4.4 seconds (on the same computer, same time of day) to find the mean (of 7000 reals), and 18.2 seconds to find the mean, variance, and standard deviation. Turbo PROLOG calculates the mean, variance, and standard deviation of 7000 reals in 9.1 seconds.

Very interesting, I thought. Very surprising, said Mike Floyd (PROLOG whiz) at Borland International. Highly irrational, said a number of C program-

Figure 2 - Statistics in C

```
#include <stdio.h>
void stat(double *list);

main()
{
   double list[7000];
   double i;

   for(i = 0; i < 7000; i++) /* Create a list to process. */
   list[i]=i + 1;
   stat(list);          /* Pass the list to stat function.*/
}
void stat(double *list)
{       int i,c;
        double x,z,dev,square,squares,var,sd;
   double sqrt(double sd);
   x = squares = 0;

   puts("Start");              /* Start timing.              */
                               /* Read array elements.       */
   for(i = 0; i < 7000; i++){
     x = list[i] + x;
   }
   z = x/i;                    /* Mean = sum of list/count. */
   printf("%f\n",z);
                               /* Calculate variance.        */
   for(i = 0; i < 7000; i++){
     dev = z - list[i];
     square = dev * dev;
     squares = squares + square;
   }
   var = squares/(i-1);
   printf("%f\n",var);
   sd = sqrt(var);             /* Find sd.                   */
   printf("%f\n",sd);
}
```

mers at SOG VI.

So, I went back to the drawing board, tried different code and different memory models and optimization techniques, but I could write nothing that was faster. So, before I complete my story (there's one more surprise), let me urge you to send in your fastest C code. If you're computing on a fast 10 MHz (80286-based) AT, your time to beat is about 2.6 seconds for the mean (the time my C code took on Mike's computer). The PROLOG code gets well under 2 seconds on that machine.

But back to the story. I'm easily convinced that I'm not an expert C programmer so I could be overlooking something. But I am at least a fair Pascal programmer so I thought I'd look for my flaws with another Borland compiler, Turbo Pascal. My fastest Pascal code is in Figure 3.

The results - 1.3 seconds for the mean, 4.8 seconds for the mean, variance, and standard deviation.

What does this mean? We're looking into it (via a little disassembly); I'll get back to you.

Meanwhile, send us your fast code and ideas (fast or slow) about compilers.

And that, friends, is Tidbits.

Half-Step Toodle-Loo.

References

Bradley, J. & J. McClelland. *Basic Statistical Concepts*, 1978. Scott, Foresman, and Co.

Gale, W.(ed). *Artificial Intelligence & Statistics*, 1986. Addison Wesley.

Figure 3 - Statistics in Pascal

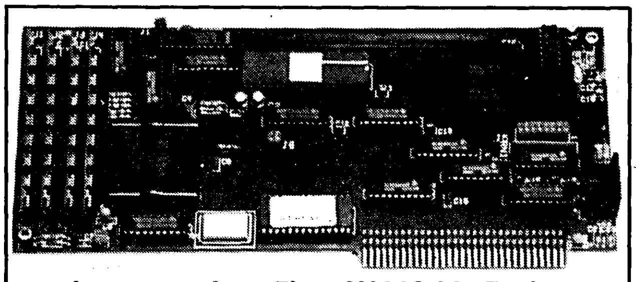
```
program stat;

type

L = array[1..7000] of real;

var

I : integer;
List : L;
X,Z,Dev,Square,Squares,Vari,Sd : real;

begin

For I := 1 to 7000 do      { Create a list to process. }
  List[I] := I;

X := 0;
Squares := 0;
Dev := 0;

Writeln('start');          { Start timing.             }

For I := 1 to 7000 do      { Read array elements.      }
  X := X + List[I];

Z := X/I;                  { Find mean.                }
writeln(Z);

For I := 1 to 7000 do      { Calculate variance.       }
begin
  Dev := Z - List[I];
  Square := Dev * Dev;
  Squares := Squares + Square;
end;

Vari := Squares/(I - 1);
writeln(Vari);
Sd := sqrt(Vari);          { Find sd.                  }
writeln(Sd);

end.
```

Figure 4 - Benchmarking The Turbos

	Compile/Link	Size	Execute
Turbo C	4.7/15.0	20K	18.2 sec
Turbo Pascal	*2.2	12K	4.8 sec
Turbo PROLOG	3.8/10.5	41K	9.1 sec

```
*  Turbo Pascal compiles to .COM.
   Turbo PROLOG & C compile/Link to .EXE.

All three compilers were tested on an 8MHz (80186 CPU) PC
Tech X16B, using an Seagate ST225 hard disk.
```

We Dare You To Abuse This Board



Announcing The SLY40-XT Turbo

A Reliable PC/XT Compatible For The Corner Stone of Your Products

- Small Board Size (4-1/4" By 9-1/4"), Four Layer
- High Integration — Composed of just 15 Low Power CMOS ICS
- NEC's 8 MHZ V40
- One Megabyte of Zero Wait State RAM
- Up to 64K of ROM
- 8087 Co-Processor Socket
- Customized Serial Port
- Standard Keyboard Connector
- Board Simply Plugs Into PC Bus or Slicer's Eight Slot Bus Board
- Slicer's Own Bios, Source Code Included
- Ideal For Tough Industrial, OEM and Portable Applications
- American Made and Fully Supported by Slicer

SLY40-XT:

0K	$300.00
256K	$350.00
1 MEG	$480.00

Eight Slot Bus Board: Call

Ask About Our Complete Line of Computer Products and Accessories!

MasterCard, Visa, Check, Money Order, or C.O.D.
Allow four weeks for delivery.
Prices subject to change without notice.
NOTE NEW ADDRESS & PHONE NO.

SLICER™

Slicer Computers Inc.
3450 Snelling Ave. So.
Minneapolis, MN 55406
612/724-2710
Telex 501357
SLICER UD

PC and XT Are Trademarks of International Business Machines

AROUND THE BEND
(continued from page 2)

I called Microscience and explained I had a very new and very dead 725. The voice on the other end of the line asked:

"Where'd you get it?"

"From you."

"No, I mean where'd you purchase it?"

"I didn't purchase it, you sent it to me."

"Why would we send you a drive?"

"Because I'm an editor and you wanted me to see how good it was."

There was a pause, then a muffled word that indicated quite plainly he'd had better days. Finally:

"Would you mind sending it back?"

I returned the defective 725 and they sent me an 825 (20 meg 65ms). (The model 825 has replaced the 725.) It fired right up, in fact it even had "No Errors" written on its error chart.

Larry and I figured the most dastardly way to abuse any drive was to stick it into the Micro C RBBS (running 24 hours, filled with all those heavy messages). So we pulled out the Seagate 225, which has lately been a little slow of seek and loud of spindle, and stuck in the 825. It came up very quietly and very quickly. (Now when you log onto the board you won't hear that awful whine.)

Meanwhile, I've gotten a letter about a Seagate 238 fix. Adaptec has upgraded its ACB-2070A RLL controller to work with the Seagate 238. The controller's new microcode makes it possible to recover from seek errors. Adaptec also changed the characters written in the gaps between the ID and data fields to enhance data recovery. Board revision number was rolled from 401400G to 401400H. (I've had no independent word on whether the fix works.)

Also, Seagate has brought out a new model. It's a 40-meg, half-height, stepper motor style drive that's very quick. I understand it supports 28 ms access time if the drive is partitioned into two 20s, 38 ms access if it's a 40. It also has automatic head park at power-down. If it's solid, it's hot. They call it the 251.

I took an informal survey at SOG VI. Six folks already had 251s. Five had had no problems. One had his die right after delivery. The 251 is selling at last year's 225 price (under $600 with controller).

Also at SOG, I asked Charles, owner of Mc-Tek, about the Seagate 225s.

"Oh yes, they had trouble for six months. They're much better now."

So, now you know as much as I know about hard drives (maybe more). NECs are available if you can find a dealer who's stashed a few away. Miniscribes are still solid performers but noisier than the NECs. And you can check the latest status on our Microscience 825 by logging onto our RBBS. If you get in, it's running.

What You Want

For me, one of the highlights of SOG was something I didn't even attend. I'd asked Peter Schenck, the Saturday dinner speaker and the best marketing mind this side of Tumalo, to conduct a focus group. I selected (randomly, of course) eight attendees and turned him loose on them. I was excluded from the fray.

In under an hour those fine folks had unequivocally laid out what they liked about *Micro C,* where they wanted the magazine to go, and where they didn't want it to go.

Declaration Of Independence

Micro C readers are independent.

If you're anything like this group, then you'll go clone if there's a reason to go clone. But you won't do it just to follow the crowd. You're also quite willing to warm up a soldering iron and try something yourself.

Micro C readers really appreciate independence and honesty in their publications.

I've been hearing this from you for a long time, but I didn't understand what you were saying. Nearly all of you, at one time or another, have told me not to go slick. To me that meant you didn't want me to change the appearance. No coated paper, no color, no graphics on the cover. (So, for a long time I resisted changing *Micro C's* appearance. It was the Infinite Improbability Drive that finally drove me to do it. Remember?)

Anyway, you were telling me to avoid slick and that's what this group was saying, too. But the group went on to explain why.

The group was afraid that if *Micro C* started looking like other magazines it would start acting like other magazines. As a fancy magazine we would attract fancy ads. Fancy ads would make us more "careful" what we wrote about our fancy advertisers. We'd lose our candor and our credibility. (And even if we didn't pull our punches, the fancy feel would put readers on their guard. Their expectations would change.)

As a fancy magazine we'd become more distant, less accessible, less informal, and less "off the wall." *Micro C's* style made the members comfortable that *Micro C* was theirs, a place where they could contribute ideas and articles or request information. For them we're an international user group. A *Mother Earth News*.

Speaking Of Mother

Sandy and I began subscribing to *Mother Earth News* during their first year of publication, and we spent long evenings reading each new copy cover to cover. For many years, their dream was our dream, and though we didn't move to a farm (a longtime dream), we rooted from a distance as they bundled up their journal and moved it onto an incredible spread.

Then yesterday, as we discussed the focus group, Peter mentioned how members of the group associated us with *Mother*.

He also mentioned that the latest issue of *Mother Earth News* is glossy, absolutely slick. Totally different from their classic newsprint paper with folksy drawings.

I was bummed. Of course, their dream may have survived the change, but I'm suspicious. I know how you feel about slick.

Where To From Here

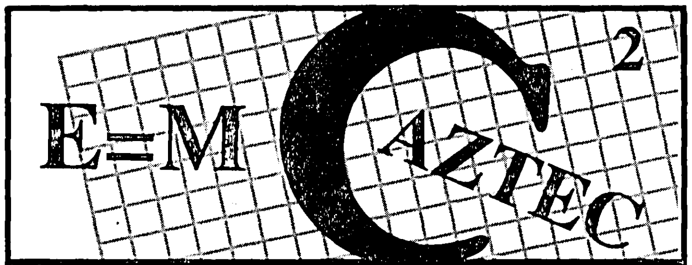
We received high praise (from the focus group) for the hard drive article in issue #36. It wasn't an exhaustive look at every conceivable (and inconceivable) hard drive, but I didn't pull any punches when it came to flaky units.

We also received high praise for the projects, personality, and humor. And, I'm here to say, unequivocally, and without malice of forethought, that we're keeping everything but the humor.

We'd like to do more such critical pieces, but we need your help. If you know something's good, or bad, tell us. Or, if you want to know if something's good or bad, ask us. (We may not know, but it'll tell us what you're interested in.)

It's easiest for us if you'll write a letter to the editor or leave a message on the Micro C RBBS. But calls are fine, too. Be sure

(continued next page)



Genius Begins With A Great Idea...

FREE 2 DAY DELIVERY

Aztec C86 4.1
New PC/MS-DOS
CP/M-86 • ROM

Superior performance, a powerful new array of features and utilities, and pricing that is unmatched make the new Aztec C86 the first choice of serious software developers.

Aztec C86-p....................$199
• optimized C with near, far, huge, small, and large memory - Inline assembler - Inline 8087/80287 - ANSI support - Fast Float (32 bit) - optimization options • Manx Aztec 8086/80x86 macro assembler •Aztec overlay linker (large/small model) • source level debugger • object librarian • 3.x file sharing & locking • comprehensive libraries of UNIX, DOS, Screen, Graphics, and special run time routines.

Aztec C86-d...................$299
• includes all of Aztec C86-p • Unix utilities make, diff,grep • vi editor • 6+ memory models • Profiler.

Aztec C86-c...................$499
• includes all of Aztec C86-d • Source for library routines • ROM Support • CP/M-86 support • One year of updates.

Third Party Software

A large array of support software is available for Aztec C86. Essential Graphics • C Essentials • C Utility Library • Greenleaf Com. • Greenleaf General • Halo • Panel • PC-lint • PforCe • Pre-C • Windows for C • Windows for Data * C terp • db_Vista • Phact • Plink86Plus • C-tree.

C' Prime
PC/MS-DOS • Macintosh
Apple II • TRS-80 • CP/M

These C development systems are unbeatable for the price. They are earlier versions of Aztec C that originally sold for as much as $500. Each system includes C compiler, assembler, linker, librarian, UNIX routines, and more.. Special discounts are available for use as course material.

C' Prime $75

Aztec ROM Systems
6502/65C02 • 8080/Z80
8086/80x86 • 680x0

An IBM or Macintosh is not only a less expensive way to develop ROM code, it's better. Targets include the 6502/65C02, 8080/Z80, 8086/80x86, and 680x0.

Aztec C has an excellent reputation for producing compact high performance code. Our systems for under $1,000 outperform systems priced at over $10,000.

Initial Host Plus Target...$ 750
Additional Targets...........$ 500
ROM Support Package....$ 500

Vax, Sun, PDP-11 ROM HOSTS

Call for information on Vax, PDP-11, Sun and other host environments.

Cross Development

Most Aztec C systems are available as cross development systems. Hosts include: PC/MS-DOS, Macintosh, CP/M, Vax, PDP-11, Sun, and others. Call for information and pricing.

CP/M • 8080/Z80 ROM

C compiler, 8080/Z80 assembler, linker, librarian, UNIX libraries, and specialized utilities.

Aztec C II-c CP/M & ROM....$349
Aztec C II-d CP/M................$199

How To Become A User

To become an Aztec C user call 800-221-0440. From NJ or international locations call 201-542-2121. Telex: 4995812 or FAX: 201-542-8386. C.O.D., VISA, Master Card, American Express, wire (domestic and international), and terms are available. One and two day delivery available for all domestic and most international destinations.

Aztec Systems bought directly from Manx have a 30 day satisfaction guarantee. Most systems are upgradable by paying the difference in price plus $10. Site licenses, OEM, educational, and multiple copy discounts are available.

MANX

To order or for information call today.
1-800-221-0440
In NJ or international call (201) 542-2121
TELEX 4995812

Manx Software Systems
One Industrial Way
Eatontown, NJ 07724

(continued from page 89)

your messages include address and phone number(s).

800 Number

For six years I've refused to get an 800 number. Sure, everyone else has one. After all:

- Phone orders are processed faster.
- Phone orders are a lot easier to place. No hassle finding an envelope, stamp, pencil, checkbook, mailbox...
- 800 numbers make phone orders easier and cheaper for you.

However, 800 calls are really collect calls (they cost us about 25 cents a minute), and because they are so expensive, we have to limit them to orders. Otherwise we would talk ourselves into poverty - in a matter of weeks.

So, we've limited the 800 number to the order department. If you need technical information about something you've purchased from us or a mod you're doing, or whatever, call (503) 382-8048 between 9 a.m. and noon, Pacific Time. Just like you've been doing.

The same goes if you have something to share. Please write it down, use the bulletin board, or call (503) 382-8048.

We're getting the 800 number to encourage more people to place orders. I've spoken with other magazines, and they report that 70% of their new subscriptions come in on their 800 lines. I'd like to have 70% of their subscriptions. (Now if I could just finagle the same 800 number as Byte...)

SOG

It was incredible. Never has so large a group spent so much time just enjoying itself. Thursday's rafting and cookout ended with cutthroat volleyball (semi-official sandlot rules) which was broken up at dusk with a quick search up the wooded canyon for a couple of misplaced youths. (I was a misplaced youth, too, but in my case there were no search parties. No parties at all.)

Of course, not everything was fun and games. That same day, three speakers called to cancel - including Andy Bakkers who had just checked into the local hospital suffering from exhaustion, an American virus, and who knows what else. (His Dutch friends told him not to drink the water.)

Two of the dropouts kept their words, but Andy showed up on Friday to speak about the transputer (though he bowed out after a shaky 40 minutes). On Saturday he was back again, his fever broken, and some of his color back. His wife and son had come with him from Holland, and I understand he's thinking about moving the whole family to Bend. Hooray. (He hasn't missed a single SOG.)

All Nighters

Anyway, it was great fun. We kept the dorm's rec room alive and well until the wee (actually all) hours. Someone would shout out a topic and a cluster would form. It was dynamically allocated parallel processing at its best. (And when the chips were down, spares were dispatched to the local market for replacements with dip.) Topics ranged from transputers to food SIGs.

Speaking of food: On Thursday and Saturday evenings, our feeding frenzies are prearranged. Friday night has traditionally been "fend for yourself." This year, on a lark, over 50 of us descended upon a new Chinese restaurant (run by three generations of Chans). (Soon to become three generations of exhausted Chans.)

In 20 minutes we'd taken over half the place. They had never had such a group, much less such an unannounced group. We were willing to eat anything.

After two hours of the finest Chinese potluck, 56 stuffed hackers gave the entire Chan family a standing ovation. (Many of the un-SOGy patrons joined our tribute.)

It was that kind of SOG, spontaneous, relaxed, more than a little crazy. We had great food, friendly help from dorm managers and bus drivers, good weather, lucid speakers (generally), fantastic information, and super attendees.

Display Area Highlights

PC Tech brought its new color graphics card. The TI graphics chip is incredible. It's a 32-bit processor that not only does great color animation but also has lots of plain old processing horsepower. They had the graphics chip generating fractals while their 80186 board was adjusting the color table on the graphics card. The TI chip was generating a fractal in an hour that would take a weekend on an AT. Larry knows, he'd had every system in the office tied up nights, weekends, days...

"Larry, can I use my system this week?"

"Is the fractal finished?"

"I don't know, I just bumped the reset. Accidentally."

"Accidentally?"

Dean and Earl stopped by the Micro C office when they pulled into Bend - they hadn't slept for 33 hours. It turned out that they had received the first batch of graphics cards at 4 p.m. the day before and both had stayed up all night populating and testing the first 30. (Sound familiar?)

If there was ever a reason to print full color in *Micro C* (nay, bind video tapes into *Micro C*), that graphics card would be it. The colors, the speed of the animation, the resolution, everything. You'll have to see it. And even then you wouldn't believe

DISK FORMAT CONVERSION

XenoCopy-PC

$79.95 + $5.00 S/H
Sales Tax if CA.

VISA MasterCard UPS COD

PC-DOS program lets your PC

READ / WRITE /
FORMAT / DUPLICATE
Disks from over 300 other micros

Upgrades available from previous versions
for only $25.00 Call for Authorization

To Order Contact:

XENOSOFT™

1454 Sixth Street, Berkeley, CA 94710
(415) 525-3113

this was on porpoise

good software is it's own reward

it. They can come back next year if they'll bring their graphics.

More Curiosities

Mc-Tek and MicroSphere were peddling hardware hot and heavy. Both had dropped their prices from a year ago, and both said sales were up significantly (three to six times).

Integrand was showing its very latest cabinet. It's a solid PC/XT/AT cabinet with an industrial duty linear power supply and room for (at least) six fans. The cabinet and supply combination remind me of an early S-100 system. I'm sure you could drop this cabinet 10 feet onto concrete without damage (assuming you could lift it 10 feet without damage).

Joe and Marla Bartel (Hawthorne Technology) showed their KAOS operating system running on their 68000-based Little Giant. I knew the little giant was small, but it was still a shock to see it. Their table was the center of a constant crowd.

The Hawthorne Technology card reminded me of Ampro's Z80-based Little Board. The 68000 is very easy to use for controller applications because it feels like a grown-up Z80. No funny addressing schemes, just lots of performance and lots of address space.

(Speaking of things to slap up along side a 5 1/4" drive, Ampro also has a new CMOS-based PC compatible complete with PC bus, multi-emulation video controller, and SCSI. So now you have 68000, Z80, and 8088 options for your next little project.)

Tapes Of The Talks

See Gary's Last Page for a list of the speakers and their topics. You'll notice that some have asterisks by them. These are the talks for which we have good quality, intelligible, audio tapes. (We also listened to the tapes to be sure we could follow the presentation without seeing the blackboard, etc.) The tapes are $6 each, postpaid. You can order one (or more) by specifying the speaker's name on a letter, the phone, or our regular order form. (Joe Bartel spoke twice, so indicate which talk.)

Two Copies Of Micro C

Last issue we sent 20,000 copies of *Micro C* to *Dr. Dobbs* readers. Of course, a number of you subscribe to Micro C and the good doctor so you received two copies. If so, please pass along one of your copies to some terminal-ridden person. Or, better yet, share it with a whole group of the terminally ill. People like your local SPCH (Society for the Prevention of Cruelty to Hackers). Or, best of all, take it to work and leave it in the reception area. (Especially if you work for *Dr. Dobbs, Byte, Computer Language, Programmer's Journal...*)

Short Listing

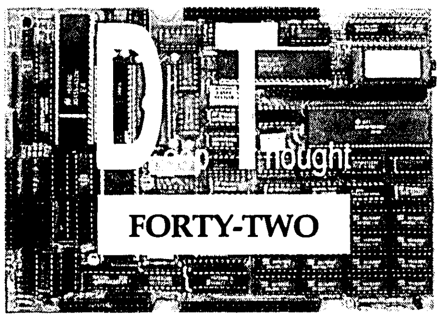
For those of you who would like the rest of the Figure 2 listing by Russ Eberhart (Teaching Your PC To Listen, pg. 51, issue #37) can call the Micro C RBBS (503) 382-7643 or send us a self-addressed, stamped envelope.

Puzzles Wanted

I'd like to include some brain twisters (human and otherwise) in the nooks and crannies of each issue. They don't have to be computer related puzzles, and they don't have to be completely original. As long they're fun, and we can get permission to print them, heck, we'll even pay.

And that's all from greater Bend.

David Thompson
Editor & Uh...



The "thoughtful" alternative from *SemiDisk.*

Designed around the 64180 microprocessor, the DT-42 is loaded with *more* of all the features you need: More speed, more memory, more ports and more TPA!

How did we fit all these features on one 5.75" by 8" single-board computer?

- 9.216MHz 64180 Microprocessor (runs Z80 programs)
- 512K DRAM, Zero wait states, fully populated.
- Three RS232C serial ports (Standard baud rates to 38,400)
- One Centronics parallel printer port
- WD2793 disk controller (up to 8 drives, SD, DD or High Density, 3½", 5¼", and 8" drives)
- SASI channel for hard disk controller (software provided)
- Many popular disk formats supported
- Requires only +5V @ 1 amp.
- ZRDOS/ZCPR3 with exclusive *"Hyperspace"* operating system, offering 57.5K TPA (NOT 48K like some others. *No 8 bit is bigger!!)* Richard Conn's ZCPR3, The Manual included free!
- Provisions for real-time clock and on-board terminal options.
- Socket for 28-pin EPROM.

Compare! You won't settle for less. Or slower. Or smaller.

DT-42 Computer	$ 499
TMP (on-board terminal)	$ 100
SmartWatch	$ 50
Z-system software	$ 50
ZAS & Debuggers	$ 25
8MB disk emulator w/ SCSI	$ 2049
Battery backup for above	$ 150

Call or write for more information or to place an order.

SemiDisk

11080 SW Allen Blvd., Beaverton, OR 97005
(503) 626-3104

68000 NOW!

TinyGiant 68000 Single Board Computer

The HT68K TinyGiant is a great little 68000 single board computer. It uses only +5 and +12 volts and has all of these features: Two Serial Ports, 1 Parallel Port, Expansion Bus, 128K RAM - Expandable on Board to 512K. Floppy Disk Controller, uses MS-DOS Disk Format. 5.75" x 8.0", Fits on a 5 1/4" Drive. K-OS ONE Operating System and Software.

K-OS ONE 68000 Operating System Package

Get the K-OS ONE operating system for your 68000 hardware. With it you can read and write MS-DOS format diskettes on your 68000 system. Included in the package are: K-OS ONE Operating System Source Code, Editor, Assembler, HTPL Compiler, Sample BIOS Code.

HT68K TinyGiant with K-OS ONE . .	$395.00
K-OS ONE Operating System Package	$50.00
HT - Forth Language	$100.00
Edit Toolkit - HTPL Source & Manual Line Editor, Screen Editor and Text Formatter	$50.00
Lizard Land - An HTPL Adventure Program with Source Code	$15.00

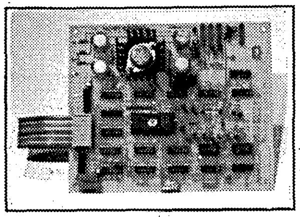
Write or call today for more information and a free 68-KNEWS letter.

Order Now:
VISA/ MC /COD
(503) 254-2005

Hawthorne Technology
8836 Southeast Stark
Portland, OR 97216

MS-DOS, CP/M E/EEPROM PROGRAMMING SYSTEM

2708
2758
2716
2516
2532*
2564*
68764*
2816A
2732
2732A



2764
2764A
27128
27128A
27256
27512
27CXXX
2864A
8751*
*SOCKET ADAPTER REQUIRED - DIAGAMS INCLUDED

A FULL FEATURED HARDWARE/SOFTWARE PACKAGE

- FAST PROGRAMMING ALGORITHM
- NO PERSONALITY MODULES REQUIRED
- INSTALL PROGRAM FOR SOFTWARE
- ALL SUPPLIES ON BOARD
- PROGRAMS 26, 25, 21 & 12.5V E/EEPROMS
- LARGE COMPREHENSIVE MANUAL
- USES NO SYSTEM POWER OR CHASSIS SLOT
- STAND-ALONE BOARD
- HIGH SPEED PARALLEL OPERATION
- FIVE LED STATUS/ACTIVITY INDICATORS
- HIGH QUALITY "TEXTOOL" ZIF SOCKET
- REQUIRES 24 OR 25 VOLT XFMR FOR POWER

PARALLEL PRINTER INTERFACE

CONNECTS TO ANY PARALLEL PRINTER INTERFACE
USES 8 OUTPUT DATA BITS AND THE PRINTER BUSY LINE FOR DATA INPUT

CONTROL PROGRAM COMMANDS

- PROGRAM EPROM(S) FROM DISK FILE
- READ DISK FILE INTO BUFFER
- READ EPROM(S) INTO BUFFER
- VERIFY EPROM IS ERASED
- CHANGE EPROM TYPE
- SAVE EPROM(S)/BUFFER TO DISK
- PROGRAM EPROM(S) FROM BUFFER
- COMPARE EPROM(S) WITH BUFFER
- COPY EPROM(S)
- BUFFER MONITOR MODE (SEE BELOW)

THE BUFFER MONITOR MODE HAS 17 SUB-COMMANDS FOR DETAILED OPERATIONS. THESE INCLUDE: FILL, DUMP, TRANSFER, PROGRAM, READ, VERIFY, EXAMINE, MODIFY, CHECKSUM, BIAS, INSPECT, SINGLE BYTE BURN, LOGICAL OPERATIONS(AND/OR/XOR), SET BUFFER BIAS, HEX ARITHMETIC, ETC.

ASSEMBLED AND TESTED UNIT WITH COMPLETE DOCUMENTATON AND SOFTWARE ON DISKETTE **$199**

PARTS KIT WITH SOFTWARE AND DOC.-$179 BARE BOARD, SOFTWARE & DOC.-$69
SOFTWARE AVAILABLE ON 5 1/4" OR 8" DISK FOR IBM, KAYPRO, & OTHER FORMATS

TO ORDER SEND CHECK, MONEY ORDER, WRITE OR CALL:

VISA

ANDRATECH
P.O. BOX 222
MILFORD, OHIO 45150
(513) 752-7218

MASTER CARD

CALL OR WRITE FOR MORE INFORMATION -- ADD $4.00 FOR SHIPPING - $3.00 COD

LETTERS
(continued from page 4)

The SOG Was Great!

This year was even more enjoyable than the last for a number of reasons: 1) My wife came and enjoyed your hospitality; 2) We went on the long raft trip - Becky got to inspect the bottom of our raft for a few minutes; 3) Some of the lectures were exceptional; 4) The B-B-Q was perfect, although I didn't play as much volleyball as I would have liked - the food was without fault - my compliments to the college's staff; and lastly a reason I'll touch on later.

The session on the Mandelbrot graphics was excellent, a use for the computer that in and of itself serves no profit. I enjoy pure math for its own sake and found the premise for the lecture rather exciting.

Akbar Tahayeri ("Computing for the Handicapped") was a diamond in the rough; your choice for a fill-in lecture was wonderful. Those of us at the conference all play with machines - we all process numbers and words and make pretty pictures, pretty much without a second thought. Here was a representative of a firm that takes the stuff we develop and uses it to open up the whole world to a group of folks that otherwise would have difficulty. I would like to see Zygo at the next SOG, showing off some of the real tools that they provide. Top Cabin! Thanks Akbar!

Speaking of wishes, I'd like to see a lecture on simple robotics, perhaps a demo or a contest where the runoff took place at the SOG.

This year's T-shirts were pretty, but if it weren't for Gary Entsminger around, I'd have never known what the plant on the shirt was or its significance.

Lastly (maybe), with the rumors running about the vaporous SOG VII, let me say this. I will gladly book a room at any hotel/motel in Bend just to go to the SOG. If a large enough group of us feel this way, most motels have a group discount rate. Sure it wouldn't be the dorm, and it would be more costly, but it would be well worth the cost if the only hang up to the next SOG is the unavailability of the dorms. If, on the other hand, the auditoriums and such are also going to be unavailable, book the armory or a motel convention center or whatever. Long live the SOGs! SOG VII lives! Support your local SOG!

Thank you for the great time. I really appreciate all of the hard work and time you all put in to make it run so smoothly. My wife could not believe how well organized the whole thing was (she must know how organized us computer types really are, eh?). I really do hope that we don't miss a year of SOG, and BRING ON THE VOLLEYBALL!!! YEAH!!!

Al J. Szymanski
8991 Edcliff Ct. SE
Aumsville, OR 97325-9549

Editor's note: Thanks a lot for the letter, Al. Actually, I think Gary was mistaken about the plant on the T-shirt. It wasn't that kind of plant at all. Honest.

About SOG VII, the college is planning to have the dorm closed for the summer while they refurbish it. Since the dorm is such a great center for all the after-hours activity, we're already trying to get them to commit to a time when we can still use the facility. Whether we get it or not, we'll have a great SOG VII.

■■■

ORDERFORM

POSTAGE-PAID SELF-MAILER
Tear out, fold, and staple both ends if check is enclosed.

TEAR HERE

TEAR HERE

☐ YES, I WANT TO SUBSCRIBE!

☐ NEW ☐ RENEWAL

	U.S.	CAN./MEX.	FOREIGN
1 yr. 6 issues	☐ $18	☐ $26	☐ $36
2 yrs. 12 issues	☐ $34	☐ $50	☐ $68
3 yrs. 18 issues	☐ $48	☐ $72	☐ $99

THE MICRO TECHNICAL JOURNAL
MICRO CORNUCOPIA
The Way We Were...
PC Mouse Drivers
Build A Midi Interface For Your PC
Designing A Database, Part 2
Interrupts On The PC
Hacker's View Of MS-DOS Vrs 3.X
And Much Much More
Mouse Control

☐ DISKS ☐ MS DOS 5¼" ☐ MS DOS 3½" ☐ Other
Specify Disk # and size

DISK TOTAL

Save **24%** Off the newstand price

☐ OTHER PRODUCTS
Back Issues, T-shirts... specify size

OTHER TOTALS

GRAND TOTAL

☐ CHECK ENCLOSED
U.S. funds drawn on a U.S. bank, please

To Place Your Order Immediately
CALL: 1-800-888-8087
9-5, M-F, Pacific Time

☐ VISA ☐ MASTERCARD

☐☐☐☐ – ☐☐☐☐ – ☐☐☐☐ – ☐☐☐☐   /
Expires

Are you a current Micro C subscriber? ☐ Yes ☐ No

NAME ....

COMPANY ....

ADDRESS ....

CITY .... STATE .... ZIP ....

NEXT ISSUE

HICS SPECIAL

- Designing A Graphics Workstation Around The TMS 34010
- Deep In The Heart Of Video: The Turbo Prolog/Turbo C Connection
- The Mandelbrot Set
- Graphics Packages For Desktop Publishing
- LISP Programming From The Bottom Up
- More Parallel Processing

TEAR HERE

FOLD HERE

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 19 BEND, OR

POSTAGE WILL BE PAID BY ADDRESSEE

THE MICRO TECHNICAL JOURNAL

MICRO CORNUCOPIA
P.O. Box 223
Bend, OR 97709-0223

FOLD HERE

SOG TAPES

Cassette tapes are available of most of the 50 minute SOG sessions.

$6.00 each ppd. — U.S./Canada/Mexico
$8.00 each ppd. — Foreign air mail

See The Last Page (Page 96) for the list of talks. The ones marked with asterisks (*) are available on tape. When ordering specify speaker's name.

TEAR HERE

STAPLE TO CLOSE

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 3 times. $474 for 6 times (a best buy at only $79 per insertion). Full payment must accompany ad. Each ad space is 2¼" by 1¾".

16 Megabytes EMS and/or Extended Memory

- Works on 8 or 16 bit bus
- 16 bit transfer on AT bus
- Single board design
- Includes RAM disk and extensive diagnostics
- Quantity/OEM discounts

XT and AT Compatible

Designed, Manufactured, Sold and Serviced by PC Tech

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

LOGITECH MOUSE

PUBLISHING Solution .. ~~$\$179^{00}$~~ $\$149^{00}$
LOGITECH Mouse with
PFS: First Publisher

CADD Solution .... ~~$\$189^{00}$~~ $\$159^{00}$
LOGITECH MOUSE with
Generic CADD 3.0 plus DotPlot.

PAINT Solution .... ~~$\$149^{00}$~~ $\$129^{00}$
LOGITECH Mouse with
LOGIPAINT (PC Paintbrush)

★*Choice of Serial or Bus Mouse*

Microsphere, Inc.•P.O. Box 1221•Bend, OR 97709
(503) 388-1194 • Hours: Mon.-Fri. 9:00-5:30 Pacific Time

INSTRUMENT FLIGHT SIMULATOR

$19

NO GRAPHICS NEEDED!

4 aircraft: trainer through fighter, air traffic control, realistic navigation, flight lessons, 32 page manual with charts. Pilots or beginners, MSDOS, CP/M Kaypro or CP/M 8".

Bailey Tech

304 WS College Yellow Springs, OH 45387

$79 *per ad*

6X rate

Z80 Development System - $\$49^{95}$

Includes Macro Assembler, Linker, Library Manager with routine library and DDT like debugger.

Also available:

Screen editor $19.95, Overlay Linker $19.95, Xref $9.95, 8080 to Z80 translator $9.95, Z-80 Disassembler $19.95.
Over 400 CP/M Public Domain disks - 100+ page catalog $8.50 pp
P & H $2.50 per order SASE

eA ELLIAM ASSOCIATES Visa/MC 818-348-4278
6101 Kentland Ave. Suite 130
Woodland Hills, California 91367

VIDEO TAPES

Understanding C $49.95

Power Programming in C $69.95

— 4 hours of instruction
— diskette with source
— companion manual

Add your sales tax, plus $3 shipping

Applied Logic

2309 Royce Dr., Arlington, Texas 76016

RAM DISK

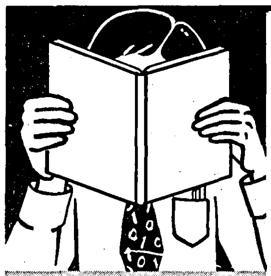
S-100
2 Meg, Port I/O
New, Warranteed
$725

S. Lugert
439 Peck Slip or call:
NY, NY 10272 718-622-0654

EFFECTIVE LOW-COST, DISPLAY ADVERTISING

ADVERTISERS INDEX

Issue 38



THE LAST PAGE

SOG VI - The Micro C Computing Conference

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Before Dave or Larry beats me to it (they're always trying to get someone in the office to corroborate their versions), let me give you my slightly unbiased report.

A SOG VI Big Eight

1. Thursday (July 30th) night's semi-official get-together at the Micro C office, after the raft trips and the barbecue along the river, (with DJT leading a behind-the-scenes tour and discussion of computing and publishing), has to rank at the top of any serious SOGy's top ten. And behind this scene one could have found Larry Fogg, cool as a cucumber, polishing his fractals in the very wee hours before Friday sunrise.

2. PC Tech graphics (Mandelbrots & Julias from Prairie Home Companion-land) were figures to feast your eyes on, leading more than a few SOGies to wag this opinion, "Wish Micro C had their lines."

3. Parallel processing (via The Netherlands - California connection) is heating up - see parallel features this issue.

4. Bruce Eckel's *simple!!* alternative to flow charting (or a Seattle writer's argument from design) led someone to ask, "Who was that marked man?"

5. Dave Rand's eleventh hour surprise appearance and the "Everything you wanted to ask Dave about UNIX Show" was a blast.

6. So was the Jefferson Street semi-official Bachelor & Couples party, which was clearly or even not so clearly an exercise in genuine slumbering, and surely the poor computerist's reply to a Scotts Valley Toga Dance.

7. Laine Stump's description of the computing biz and rafting in Turkey, and Akbar Tahayeri's "Computing for the Handicapped," were (to say the least) very interesting.

8. And the clear, cool high desert days and Cascade volcano views set the stage for a stimulating group of speakers. Thanks again to -

- Roger Armstrong, "Programming With Microsoft Windows"*
- Andy Bakkers, "Computing In Europe"*
- Joe Bartel, "Inside K-OS One"*
- Marla Bartel, "Single Board Computer Setup"*
- Earl Brabandt, "Intro To ASIC Chip Design"*
- Chris Cale, "The State Of Modula 2"*
- Jack Dennon, "New Developments In RP/M"
- Bruce Eckel, "Structured Development For Real-Time Systems"*
- Allyn Franklin, "Drive Workshop"
- Mike Freiling, "Knowledge Engineering: A Software Technology"*
- Roe Fulleton, "Data Base Design"*
- Peter Henry, "Real-Time Motion Control: The Cornea Lathe"*
- Earl Hinrichs, "Expanded Memory On The PC"*
- Chris Jones, "Parallel Processing & Computer Architectures"
- Dean Klein, "Narrowing The Gap Between PCs And Work Stations"*
- Trevor Marshall, "The Promise Of RISC"
- Tom Ochs, "Open Architecture Software"*
- John Popplett, "Parallel Processing & The Transputer"*
- Dave Rand, "Inside UNIX"*
- Mike Sequeira, "Computers 101"*
- Reese Shepard, "Money, Marketing, & Management For Small Businesses"*
- L. Nelson Spohnheimer, "LAN Implementation & Application"*
- Laine Stump, "Trials & Tribulations In Turkey"
- Akbar Tahayeri, "Computing For The Handicapped"
- Mike Vore, "Packet Radio Today"*
- Greg Wolfson, "StarLan, Ethernet, & Cheapernet"

(Editor's note: Those talks marked with asterisks (*) are available on tape for $6 each, post-paid, from Micro Cornucopia. Specify speaker's name on a Micro C order form, or call 1-800-888-8087.)

And thanks to the 300+ of you who attended. See you next year, and in the back pages of *Micro Cornucopia.*

ASSEMBLE YOUR OWN!

VIDEO CARDS

Color/Graphics	55.00
Color/Graphics/Parallel	65.00
EGA Graphics	195.00
Mono/Graphics	59.00
Mono/Graphics/Parallel	69.00
ATI Graphics Solution— Mono, Herc. Color Emulation on Mono CGA .....(List 299)	150.00
ATI Wonder— Mono, Herc, CGA, EGA, VGA .....(List 499)	299.00

Any monitor, Any software, Auto conversion

EXPANSION CARDS

Clock Card	25.00
Floppy Disk Controller —2 drives	25.00
Game Port	19.00
Multi-Function, 1 ser/par/clk/game/ 2 floppy	79.00
Parallel (printer)	19.00
Serial Port (RS232) 1 port + 1 optional	29.00
640K RAM (ØK installed)	35.00
XT/AT RS232 (4 port/2 installed)	59.00

CASES & POWER SUPPLIES

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

MONITORS

EGA/CGA (Auto Switch)	495.00
CGA Color	339.00
Samsung Amber 12″ TTL	89.00
Samsung Green 12″ TTL	89.00

MOTHERBOARDS

XT/Turbo 4.77/10mhz	119.00
AT 6/10 mhz (5 layer)	425.00
XT/Turbo 4.77/8 mhz (2 layer)	109.00
XT/Turbo 4.77/8 mhz (4 layer)	119.00
For XT/AT: (120ns chips) 640K memory	75.00
For AT: 1MB memory	125.00

KEYBOARDS

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

FLOPPY DISK DRIVES

Fujitsu 360K	97.00
Toshiba 360K	99.00
Toshiba 1.2 MB	145.00
3½″ Drive Kit	145.00

GAMES, GAMES, GAMES

Where in the USA is Carmen Sandiego by Broderbund	44.95
Flight Simulator by MicroSoft	49.95
* Hitchhikers Guide to the Galaxy	34.95
* Wishbringer	39.95
* Moonmist	34.95
Night Mission Pinball by SubLogic	39.95
*Some Infocom games still available in CP/M Kaypro format	Call

HARD DRIVES & CONTROLLERS

AT 40 MB Seagate #251	575.00
AT Hard Drive & floppy controller (WD)	199.00
20 MB Miniscribe HD with controller	349.00
30 MB Miniscribe HD with controller	399.00
AT 30 MB Seagate HD #4038	599.00

New Prices **SPECIAL KITS**

All kits include: 640K RAM, serial, parallel and game ports, clock/calendar, AT-style keyboard, cabinet, power supply, mono graphics card and amber or green monitor. Keyboard switchable turbo.

XT Systems with 2 360K Floppy Drives:

8 mhz with standard slide cabinet	649.00
10mhz with lock, LED, Reset & Turboswitch	699.00

XT Systems w/20 MB Miniscribe, Hard Drive & 1 360K Floppy Drive

8 mhz with standard slide cabinet	949.00*
10mhz with lock, LED, Reset & Turboswitch	995.00*

*(For 30MB Miniscribe add $50.00)

AT System with 1 1.2 MB Floppy Drive, 0 wait state
1 360K Floppy drive and Seagate St 4038 Hard Drive, 6/10mhz 1895.00

**Suggested additions —

MS DOS 3.21 with GW Basic	95.00
5339 Keyboard Sub	24.00

Assembly and testing is available.

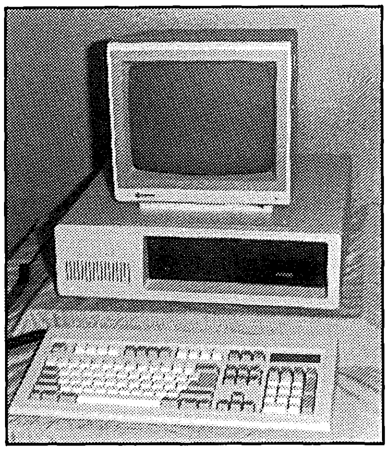
XT Systems	60.00
AT Systems	80.00

From The Inside Out: Tech Tip —
Installing the correct video card is critical to allowing most software packages to operate correctly. Examples of available types of graphics are monochrome (text only), mono graphics (herc 720x348), CGA Color (std IBM color 640x200), EGA color (enchanced color 640x350), VGA color (640x480) ATI's Wonder cards and 7 MODE MULTIVIDEO cards offer multiple video solutions from one card.

ACCESSORIES

1200 Baud Modem — Internal (Leading Edge Model L) Hayes compatible	99.00
1200 Baud Modem — External Hayes compatible	119.00
MS-DOS 3.21with GW Basic	95.00
Joystick (IBM Style)	25.00
V20-8mhz	14.00
Memory Chips	(call for prices)
Borland Turbo C	89.00

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



— Pictured keyboard is 5339 —

BUILDING YOUR OWN CLONE

****FREE BOOKLET****

****90-day warranty/30-day money-back (Subject to restock fee).***

Free instructions with each system.

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

MicroSphere

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

MasterCard VISA

Borland's Turbo Prolog, the natural introduction to Artificial Intelligence

Nothing says Artificial Intelligence has to be complicated, academic or obscure. Turbo Prolog® proves that. It's intelligent about Intelligence and teaches you carefully and concisely so that you soon feel right at home.

Which is not to say that Artificial Intelligence is an easy concept to grasp, but there's no easier way to grasp it than with Turbo Prolog's point-by-point, easy-to-follow Tutorial.

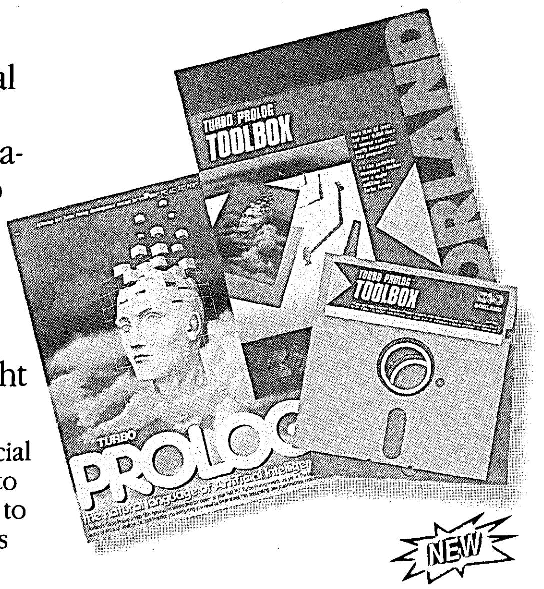
Turbo Prolog is for both beginners and professional programmers

Because of Turbo Prolog's natural logic, both beginners and accomplished programmers can quickly build powerful applications—like expert systems, natural language interfaces, customized knowledge bases and smart information-management systems. Turbo Prolog is a 5th-generation language that almost instantly puts you and your programs into a fascinating new dimension. Whatever level you work at, you'll find Turbo Prolog both challenging and exhilarating.

Turbo Prolog is to Prolog what Turbo Pascal is to Pascal

Borland's Turbo Pascal® and Turbo C® are already famous, and our Turbo Prolog is now just as famous.

Turbo Pascal is so fast and powerful that it's become a worldwide standard in universities, research centers, schools, and with programmers and hobbyists. Turbo Prolog, the natural language of Artificial Intelligence, is having the same dramatic impact.



Borland's new Turbo Prolog Toolbox adds 80 powerful tools

Turbo Prolog Toolbox™ includes 80 new tools and 8000 lines of source code that can easily be incorporated into your own programs. We've included 40 sample programs that show you how to put these Artificial Intelligence tools to work.

Already one of the most powerful computer programming languages ever conceived, Turbo Prolog is now even more powerful with the new Toolbox addition.

The Critics' Choice

"I really wouldn't want to choose *the* most important MS-DOS product developed last year, but if I had to, I think it would be Borland's Prolog, which gives users a whole new way to think about how to use their computers.

Jerry Pournelle, 'A User's View,' InfoWorld

Turbo Prolog offers the fastest and most approachable implementation of Prolog.

Darryl Rubin, AI Expert"

Turbo Prolog Features:

- ☑ A complete development environment
- ☑ A fast incremental compiler
- ☑ A full-screen interactive editor
- ☑ Graphic and text window support
- ☑ Tools to build your own expert systems
- ☑ Full DOS access and support
- ☑ A free Tutorial
- ☑ The free GeoBase™ natural query language database
- ☑ An easy-to-understand 200-page manual

All this and more for only $99.95!

The new Turbo Prolog Toolbox includes:

- ☑ 80 tools
- ☑ 8000 lines of source code that can easily be incorporated into your own programs
- ☑ 40 sample programs
- ☑ Business graphics
- ☑ File transfers from Reflex,® dBASE III,® 1-2-3® and Symphony®
- ☑ Sophisticated user-interface design
- ☑ Screen layout and handling—including virtual screens
- ☑ Complete communications package including XMODEM protocol
- ☑ Parser generation
- ☑ Opportunity to design AI applications quickly
- ☑ 5th-generation language and supercomputer power to your IBM®PC and compatibles

Only $99.95!

For the dealer nearest you or to order by phone call
(800) 255-8008
in California (800) 742-1133 in Canada (800) 237-1136

BORLAND
INTERNATIONAL

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

BI-1120