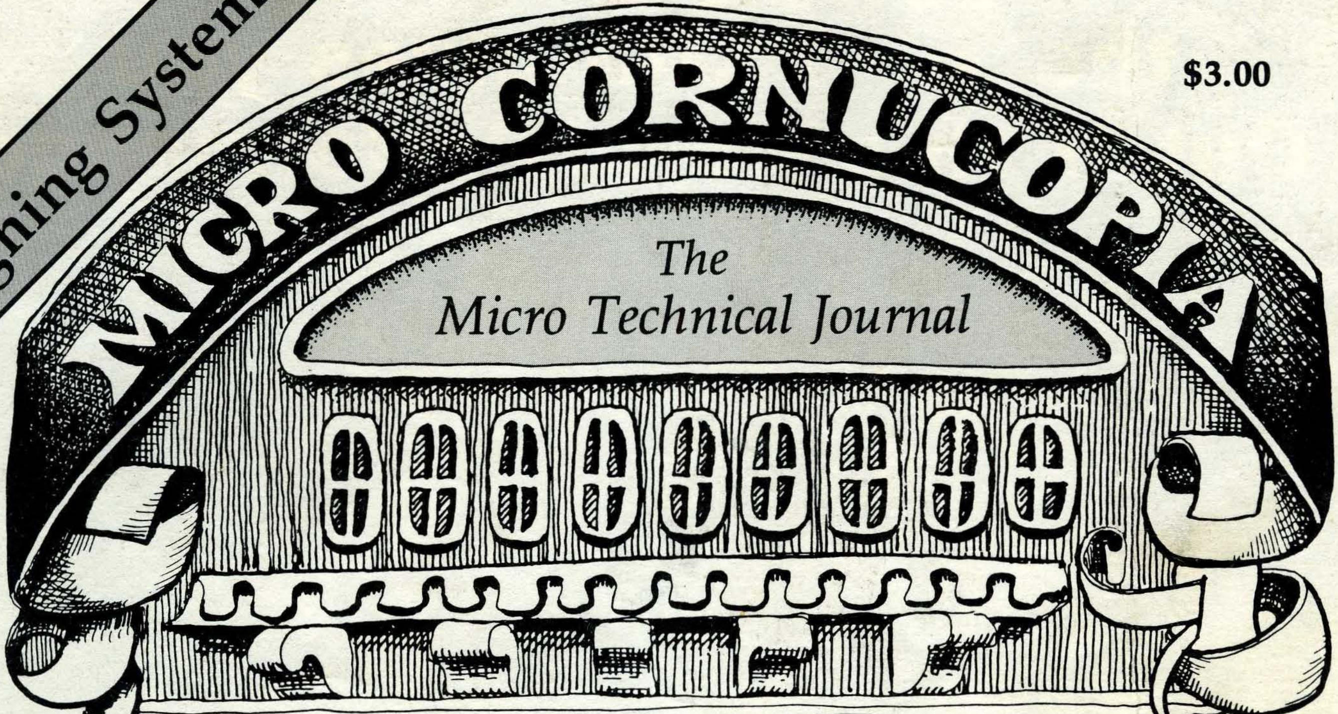
Designing Systems

$3.00



February-March 1987 No. 34

Hardware

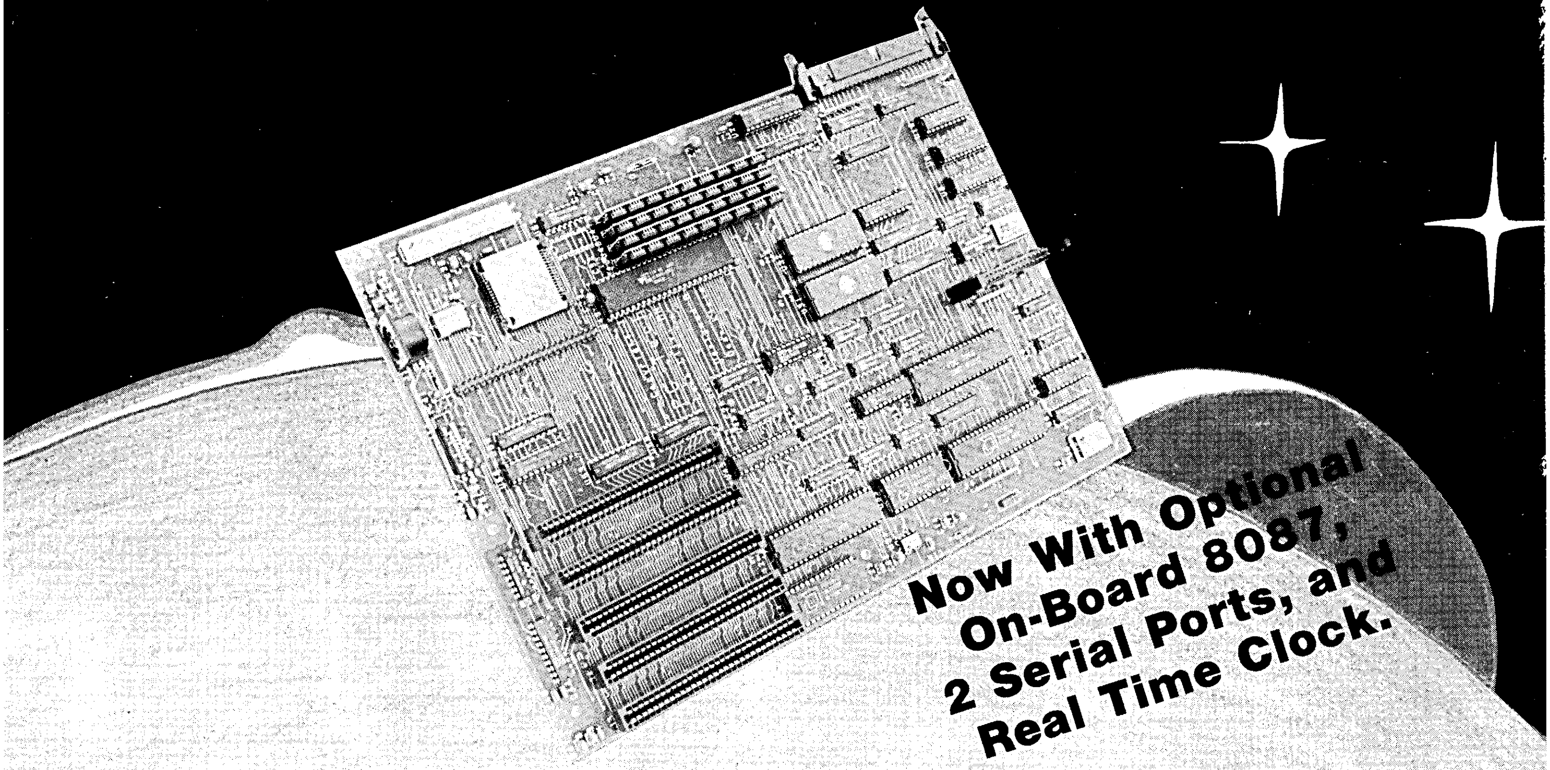
Software

02

0 74470 19388 3

NEW LOWER $

the X-16B gives you Out of This World Performance



PLUS PC COMPATABILITY!

Features:

- 8 MHz 80186 microprocessor with true 16-bit data bus.
- True PC compatability with our own custom ROM BIOS and PC cardslots for the video of your choice.
- Standard 512K zero wait DRAM, or 1M options.
- NEW 8087 numeric coprocessor support on board.
- On-board floppy disk controller for up to four 5¼" drives.
- On-board SCSI hard disk controller port. Supports WD1002 SHD, WD1003 ACS, OMTI 5100, 5300 ADAPTEC ACB 4000.
- NEW 2 On-Board Serial Ports.
- Battery Back-Up Real Time Clock.

Designed, Manufactured and Serviced by:

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

COD, VISA, MASTER CARD, CERTIFIED FUNDS

NEW LOWER PRICES

- Assembled and Tested
  - W/O SCSI, serial ports, clock .. 1-4 $ 440.00
  - 5-9 $ 525.00
  - SCSI Port .... add $ 20.00
  - Serial Ports .... each - add $ 12.50
- ½ MEG Add-On .... $ 105.00
- XT Style, Enclosure .... $ 75.00
- 150 Watt Power Supply, Tested ..... $ 105.00
- NEW 8087-2 Coprocessor .... $ Call

Total Solutions

Assembled and Tested Systems

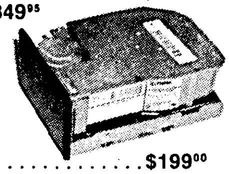
- Mono-Chrome System $1099.00
  Includes: 512K, Power Supply, Enclosure, Keyboard, Monitor, Two Drives, and Printer Port, 1 Serial Port, Real Time Clock.
- Quantity Discounts Upon Request
- Call For Info on Add-Ons, Hard Disk, Etc.

—HARD DISKS—

SEAGATE ST225 . . . . $399.00
—WITH CONTROLLER
—20 MB ½ HEIGHT
—1 YEAR GUARANTEE

RODIME 202E $279.95
—20 MB FULL HEIGHT
—55 MS ACCESS
—30 DAY WARRANTY

WITH CONTROLLER (XEBEC)
$349.95



RODIME 352 . . . . $199.00
—10 MB, LOW POWER
—3½" SHOCK MOUNTED IN 5¼" FRAME
—PERFECT FOR YOUR PORTABLE!
—FULL HEIGHT BEZEL

WITH CONTROLLER (XEBEC)
$249.95

WITH WD CONTROLLER
$279.95

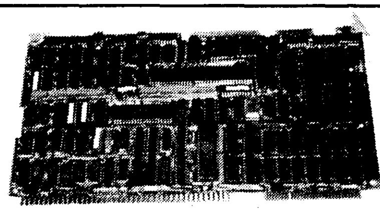
—HARD DISK CONTROLLERS—

XEBEC 1210A $79.95
—FULL LENGTH
—CONTROLS 10 & 20 MB
—CUSTOM CONFIGURATIONS AVAILABLE

WESTERN DIGITAL WX2 . . . . . . $119.95
—WITH CABLES

OMTI 5520 . . . . $124.95
—WITH CABLES
—SHORT SLOT
—CONFIGURABLE BIOS
—RUNS ANY 3" OR 5" HARD DISK
—SOFTWARE

XEBEC 1410A $109.95
—ORIGINAL XEBEC SASI CONTROLLER
—CONTROLS 2 HARD DISKS



HAZELTINE 1500 TERMINAL

USED, TESTED, GOOD
$89.95

VARACTOR TUNER

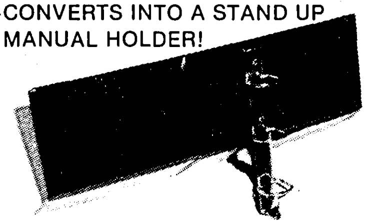
$7.95
—UHF/VHF
—133 CHANNEL CABLE READY
—VOLTAGE CONTROLLED CHANNEL SELECT
—DOCUMENTATION INCLUDED!

SYNC-SEPARATOR KIT

$12.95
USE WITH ANY 3 LINE TTL MONITOR ALLOWS YOU TO RUN A 3 LINE MONITOR FROM A COMPOSITE VIDEO SOURCE

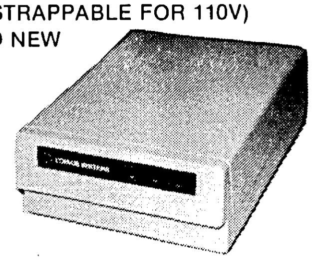
6" BINDERS

CASE OF 28 — $24.95
—STANDARD 6" 3 HOLE BINDERS
—COMES WITH DISK ENVELOPE PAGE
—PERFECT FOR SOFTWARE MANUALS!
—CONVERTS INTO A STAND UP MANUAL HOLDER!



CORVUS EXTERNAL HARD DISK CHASSIS

$99.95
—ATTRACTIVE, RUGGED CHASSIS
—INCLUDES FAN & POWER SUPPLY (220v, STRAPPABLE FOR 110V)
—BRAND NEW



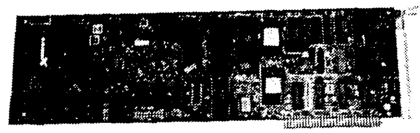
SAME AS ABOVE WITH 5MB HARD DISK & VCR BACKUP CONTROLLER
$149.95

INTEL 86/30 . . . . $495.00
—8086 MONOBOARD COMPUTER
—MULTIBUS
—NEW, SCHEMATICS INCLUDED

SPECIAL OF THE MONTH

XEBEC 1210A
HARD DISK CONTROLLER
FOR IBM PC/XT
$79.95

• 10 or 20 MB
• CONTROLS 2 HARD DRIVES
• CUSTOM CONFIGURATIONS AVAILABLE



— DISK DRIVES —

8" SHUGART SA801R $179.95
—SINGLE SIDED DOUBLE DENSITY
—600 KB FORMATTED CAPACITY
—BRAND NEW

—DISKETTES—

SHUGART DSDD 8" BOX OF 10 . . $12.50
VERBATIM SSSD 8" BOX OF 10 . . $ 4.95

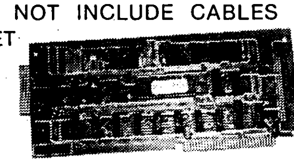
5¼" MPI FULL HEIGHT $49.95
—DOUBLE SIDED
—IBM COMPATIBLE

TANDON TM 50-2 . . . . $79.95
—DOUBLE SIDED, HALF HEIGHT
—IBM COMPATIBLE

TEAC 55BV $109.95
—DOUBLE SIDED, HALF HEIGHT
—360K IBM COMPATIBLE

TEC FB503 . . . . $99.95
—IBM COMPATIBLE, HALF HEIGHT
—SIMILAR TO TEAC

DUAL FLOPPY CONTROLLER . . . $19.95
—IBM COMPATIBLE
—48 & 96 TPI DRIVES
—DOES NOT INCLUDE CABLES & BRACKET



HALTED SPECIALTIES CO., INC.

827 E. EVELYN AVE., SUNNYVALE, CA 94086

MAIL ORDERS CALL: (408) 732-1573

3 CONVENIENT LOCATIONS

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

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

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

WE SHIP C.O.D.

MasterCard VISA AMERICAN EXPRESS

TERMS: Minimum order $10.00. California residents add 7% sales tax. Prepaid orders sent freight C.O.D. or call for charges, shipping will be added to credit card and C.O.D. orders. Prepaid orders over $100.00 use money order or certified check. Please do not send cash. Some items limited to stock on hand. Prices subject to change.

MICRO CORNUCOPIA

The Micro Technical Journal

Editor & Publisher
David J. Thompson

Associate Editors
Gary Entsminger
Margret Rosenberg

Technical Department
Larry Fogg

Director of Advertising
Laura Logan

Director of Marketing & Graphics
Julie Pope

Accounting
Sandra Thompson

Circulation Manager
Tracey Anthony

Order Department
Cary Gatton Tammy Westfall

MICRO CORNUCOPIA (ISSN 0747-587X) is published bi-monthly for $16 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)	$16.00
2 yr.(12 issues)	$30.00
3 yr.(18 issues)	$42.00
1 yr.(Canada & Mexico)	$22.00
1 yr.(Other foreign)	$30.00

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.

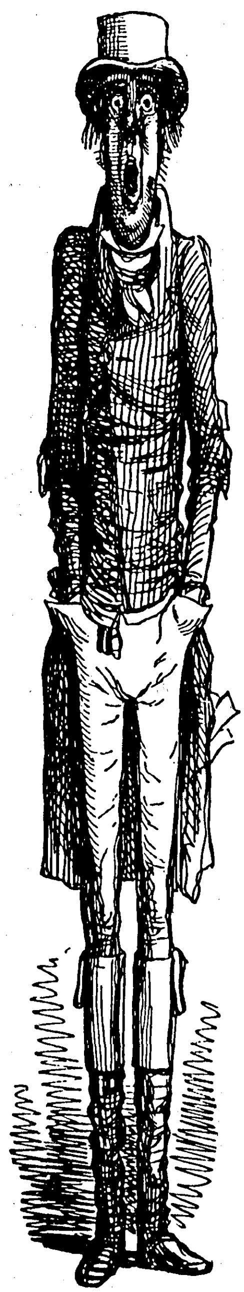
RBBS - 24 hrs. 300-1200-2400 baud
8 Bits, No Parity, 1 Stop Bit
503-382-7643

ISSN 0747-587X

AROUND THE BEND

By David Thompson

Duplicating My Efforts



Copy Protection

Copy protection used to be the exclusive bane of the Apple world. CP/M just didn't lend itself to such curses — too diverse a hardware base and too sophisticated a user base.

The PC world, on the other hand, was perfect for copy protection, and copy protection it got. Soon there were copy protectors copying copy protectors (a bit incestuous I'd say).

PC users suffered through it, stoically at first. (What was the alternative? Nearly everything worthwhile was protected.) Later, they purchased unprotectors and suffered with the compatibility/incompatibility problems they caused.

Through all of this, purchasers were voting with their wallets. Given an alternative, they weren't voting for copy protection.

Thus, we've seen a rash of recent announcements as major software manufacturers have dropped copy protection. Shucks.

Shucks?

I kind of like copy protection.

Sure it's a pain in the you-know-where if you can't make backups. Sure you can lose a hard disk while uninstalling a copy protected program. Sure it may cost you (and the software company) $XXX.XX per system for the hardware attachment that lets the software run (and there's no guarantee the hardware will always work).

But think of the advantage of protectionism.

It leaves the market wide open for newcomers.

Businesses are willing to pay extra for the very best. They buy name, name, name. But they are very leery of copy protection. If someone offers a non-copy protected version, that means a lot.

Also, businesses are made up of people (you heard it here first). And one or more of those people (often the house computer freak) makes the decisions on which packages to purchase. The house freak may well have a purloined copy of someone's spreadsheet, and someone else's database — as well as a couple of word processors.

(continued on page 84)

February-March 1987
Issue No. 34

CONTENTS

Features

Columns

Future Tense

By Gary Entsminger

MS-DOS Goodies

I recently switched from my BBI to a PC clone, mainly because there was no more software being written for CP/M. While I kind of miss the old machine, I really like the advanced software that I can run on this new machine. One product that I haven't seen much about is Homebase v. 2.0. I've used Sidekick, and Homebase is a far superior product.

During a recent business trip to San Jose, I went to a computer swap meet and ran across a couple of products you might be interested in. Microware Exceltek had a 2 meg Above Board without RAM for $109 plus tax. It came with the software for the device drivers and sample CONFIG.SYS files, but no RAMdisk or print spooler. (However, VDISK.COM (IBM) works fine. I still haven't found a print spooler that works in the Above Board memory — if you know of one I'd like to hear about it.) Microware Exceltek's address is:

Microware Exceltek
100 Produce Ave., Unit E So.
San Francisco, CA 94080
Phone: 415-952-5375/5377

I wanted to be sure that the thing worked before I returned to Texas, so I asked for the best place to buy RAMs and was referred to Fry's Electronics.

It was an experience. When I first walked in I looked to my left and saw toothpaste, shaving cream, and other assorted items. Looking straight ahead, I saw TVs, VCRs, and stereo equipment. I thought I was in some sort of consumer store, not an electronics shop. But, looking a little further, I saw components for RS-232 cables, printer cables, extension cables, plus PC cards, computers, printers, and lots of software.

Anyway, after I recovered from my shock (I've never heard of, much less been in, a store like this), I was pleasantly surprised when the bill for nine 256K RAM chips came to $21.08. Now in all fairness, I saw their ad in a local magazine advertising the same chips for something like $27.00. Still, if you're trying to fill out one or two megs of 256K RAM, you might want to give them a call. Their address is:

Fry's Electronics
541 Lakeside Dr.
Sunnyvale, CA 94086
Phone: 408-733-1770

I also ran across an inexpensive EPROM programmer. It plugs into a PC parallel printer port, uses an external wall type power supply (included), and programs 2716-27256 EPROMS. It uses both standard and fast programming algorithms, has software on disk, and the great part is the cost — $59.95, plus tax and shipping. It's available through Micro Mart (201-654-6008), and is made by:

Sterling Technology
1010 12th St.
Sparks, NV 89431

Now I want to put in a request for a

future User Disk. I'd like a hard disk utilities disk. It would have programs to do such things as return the heads to the home position before you turn the power off, clean up scattered out sectors (like Disk Optimizer), and check memory (both main memory and expanded).

Thanks for a nice publication.

John Scott
2206 Cambridge Court N
League City, TX 77573

Chrono Pascal Strikes Again

In issue #33, in your call for material for the April 87 issue, you refer to times when you've "run science fiction as science fact." This probably explains how you work your excellent retrospective mailing system, which brought me your December-January issue today, December 2, while the postmark on the envelope, proves conclusively that it will not be mailed from Bend until December 24.

You are obviously leaving no stone unturned to ensure prompt delivery to Australia. This is far more than can be achieved simply by shipping across the International Date Line. If our postal system had a service like that, I might even come close to meeting writing deadlines.

John S. Innes
120 Macpherson St.
Cremorne NSW 2090
Australia

Editor's note:

Thanks for the note, John. It arrived here only a week before we sent out your December/January issue. I must confess however, that we didn't use Chrono Pascal, just an old trick. We had the shipping company fly it across the date line backwards, 22 times.

Support Problems

As a microcomputer enthusiast who's been involved in buying, building, interfacing, and using this equipment since the introduction of the Altair, I want to warn your readership about dealing with a firm which is well-advertised in the microcomputer journals. The firm in question is:

Express Systems, Inc.
1254 Remington Rd.
Schaumburg, IL 60195

In October 1985 I purchased a hard disk drive/controller from Express Systems. The unit was supposedly backed by a one-year repair/replace warranty, and the text of the two-page advertisement was sufficiently credible that I was led to believe they would support it.

After nine months of normal operation, the drive failed. I began trying to contact Express through their "Technical Help" phone line. This proved to be a feat in itself, but I finally obtained authorization to return the unit for repair, which the representative estimated would require two weeks. I packed the unit as prescribed and returned it.

After nearly two months had elapsed, I began trying to contact Express once more for some kind of progress report on the unit. This time

(continued on page 76)

Designing With The 80386

Inside Intel's Latest Processor

By Dean A. Klein
PC Tech Inc.
904 N. 6th St.
Lake City, MN 55041

To say that the 80386 was one of the stars of Comdex is like saying that MS-DOS is a well-known operating system. From the words I was hearing from speakers and writers, the 386 was the magic for the next century. (I heard that it would run UNIX, support multiple users, run programs written for any processor, and do it all at speeds that would make a Cray proud.)

If you believe all that, then you probably work for Intel's marketing department. However, there's just enough truth there to make the chip interesting. Very interesting.

I've just come from Comdex Fall '86 where I witnessed about as much of the latest computer software and hardware accessories as a designer can stand. The writing was on the wall. Allow me to elaborate.....

First Entries

By now most of you know that Compaq has introduced the Compaq Deskpro 386. Corona (Cordata) has also introduced an 80386-based product, as have several other manufacturers. Intel is promoting its new child both as a chip for people like us to build into our products, and also for several products of its own, some for the PC market.

Companies with deep roots in other 32-bit processors are also branching toward the 80386. Sun Computer, a longtime leader in 68000 workstations, announced at Comdex a Network Extensible Windows System (NEWS) for the 80386.

Eric Schmidt, vice president of Sun's Software Products Division, has said, "If you want to establish the standards you go after the major platforms." Apollo Computers, another 68000 workstation manufacturer, is suspiciously tight-lipped regarding the 80386.

Most of the big software houses demonstrated their products on the Compaq 386. Certainly a large percentage of Compaq's initial production of 80386 computers has gone to places like Microsoft, Novell, Goldhill, Borland, and dozens of others.

The point is — the 80386 is THE major computing platform for the next few years.

There will be at least two dozen new 80386-based computers on the market within the next six months, probably none of them from IBM. Most of these computers will be similar, and quite frankly, similar for a good reason: it's going to be tough to design an 80386 that's much different from the rest. (Oh come on Dean, where's the fighting spirit?)

Yet designing with the 80386 is anything but easy, and 80386 computers will differ widely in performance. Let's look closer at designing with the 80386, some of the factors which affect performance of the design, and the techniques used by some of the leaders in 80386 machines.

The Part

The 80386's 132-pin grid array package is quite formidable. See the pinout in Figure 1.

How's that for density? So many pins in such a small area is a key problem in designing with the 80386. It's difficult, to say the least, to connect all the pins on the chip to the connections on the board. So expect most '386 designs to have four or

Figure 1 - 80386 Pinout

	P	N	M	L	K	J	H	G	F	E	D	C	B	A	
1	A3Ø	A27	A26	A23	A21	A2Ø	A17	A16	A15	A14	A11	A8	VSS	VCC	1
2	VCC	A31	A29	A24	A22	VSS	A18	VCC	VSS	A13	A1Ø	A7	A5	VSS	2
3	D3Ø	VSS	VCC	A28	A25	VSS	A19	VCC	VSS	A12	A9	A6	A4	A3	3
4	D29	VCC	VSS									A2	NC	NC	4
5	D26	D27	D31									VCC	VSS	VCC	5
6	VSS	D25	D28									NC	NC	VSS	6
7	D24	VCC	VCC									NC	INTR	VCC	7
8	VCC	D23	VSS									PEREQ	NMI	ERROR#	8
9	D22	D21	D2Ø									RESET	BUSY#	VSS	9
1Ø	D19	D17	VSS									LOCK#	W/R#	VCC	1Ø
11	D18	D16	D15									VSS	VSS	D/C#	11
12	D14	D12	D1Ø	VCC	D7	VSS	DØ	VCC	CLK2	BEØ#	VCC	VCC	NC	M/IO#	12
13	D13	D11	VCC	D8	D5	VSS	D1	READY#	NC	NC	NA#	BE1#	BE2#	BE3#	13
14	VSS	D9	HLDA	D6	D4	D3	D2	VCC	VSS	ADS#	HOLD	BS16#	VSS	VCC	14
	P	N	M	L	K	J	H	G	F	E	D	C	B	A	

more signal layers on the printed circuit board.

Because of the high frequency of the signals in a typical 80386 system, the circuit board will also have a minimum of two power planes — one for +5 V and one for ground.

A six or eight layer board is spendy. A six layer board costs about 1.8 times as much as four layers. This seems at first to be a serious drawback, however, the additional circuit layers will allow designers to pack the chips much more densely, yielding boards that will be smaller than four layer versions.

Two other benefits come to mind. The potential reduction of board size helps to keep the signal traces shorter, a very important factor at these speeds, and the reduction in size should result in a reduction in board cost, though not enough to offset the cost of the additional layers.

Address & Data

The 80386 uses a non-multiplexed address and data bus. This differs from much of the Intel family — 8088, 8086, 80188, and 80186 — which use the same pins to output an address during the first part of the bus cycle and then to write or read data to or from an I/O device or memory.

In contrast, the 80286 and 80386 can simultaneously give the address on the 34 address lines and read or write data over the data lines. There are several advantages to this type of bus configuration.

First, the bandwidth requirements (how fast they must generate or respond to data) of the pins are reduced since each pin has only one function. A second advantage is that some systems won't have to latch the address lines from the processor. This may reduce component costs.

I mentioned earlier that the 80386 has 34 address lines. But that's not entirely accurate. The 80386 actually has 30 address lines and 4 byte-select lines. Because it has to read or write one, two, three, or more of the bytes of data which may be addressed within the 32-bit word activity, each byte is controlled by an individual byte select line. See Figure 2.

Those patterns which never occur are patterns which aren't contiguous byte operations.

Having 30 address lines and the 4 byte enables allows the 80386 to directly address an outrageous amount of memory — 4 gigabytes.

Just for the fun of it, imagine you're using the new 1 megabit DRAM chips. 4 gigabytes of memory would be four thousand sets of eight chips, or 32 thousand memory devices (36 thousand if you used parity!). At the price of 20 dollars per chip (after all we'd be buying in volume, right?), this comes to a mere $640,000. One heck of a RAMDISK. Certified check only, please.

Editor's note: I wonder how long the power up RAM check would take.

Other Control Signals

Several other control signals are provided by the 80386 which indicate the type of operation being performed. W/R distinguishes between write and read cycles. D/C distinguishes between data and control or code cycles. M/IO distinguishes between memory and I/O cycles. These operations are summarized in Figure 3.

Halt and shutdown cycles can be distinguished by observing BE0, which is low for shutdown and high for halt. Halt occurs when the processor executes the HLT instruction. A non-maskable interrupt or a normal interrupt with interrupts enabled will cause the processor to exit the halt state and resume program execution. Reset will also bring the 80386 out of the halt state.

Shutdown occurs when a severe error is detected while operating in the 80386's Real Mode. For example, if a hardware interrupt or software exception occurs and the interrupt vector is larger than the Interrupt Descriptor Table, then Shutdown. This could be the case when there is no interrupt handler for the interrupt.

A second shutdown condition occurs if a CALL, INT, or PUSH instruction

(continued next page)

Figure 2 - 80386 Byte Enable Signals

/BE3	/BE2	/BE1	/BE0	Operation
1	1	1	1	none
1	1	1	0	D0-D7
1	1	0	1	D8-D15
1	1	0	0	D0-D15
1	0	1	1	D16-D23
1	0	1	0	never occurs
1	0	0	1	D8-D23
1	0	0	0	D0-D23
0	1	1	1	D24-D31
0	1	1	0	never occurs
0	1	0	1	never occurs
0	1	0	0	never occurs
0	0	1	1	D16-D31
0	0	1	0	never occurs
0	0	0	1	D8-D31
0	0	0	0	D0-D31

Figure 3 - Control Signals

M/IO	D/C	W/R	Bus Cycle Type
0	0	0	Interrupt Acknowledge
0	0	1	does not occur
0	1	0	I/O Data Read
0	1	1	I/O Data Write
1	0	0	Memory Code Read
1	0	1	Halt or Shutdown
1	1	0	Memory Data Read
1	1	1	Memory Data Write

(continued from page 7)

attempts to wrap around the stack segment when the stack pointer is odd.

Shutdown can be exited by NMI if the Interrupt Descriptor Table limit is greater than 17H and the stack pointer is greater than 5. Otherwise only a reset can restart the 80386. Most 80286 and 80386 PC designs detect shutdown and automatically reset the processor.

Dynamic Bus Sizing

An interesting aspect of the 80386 is its ability to change the size of the data bus in response to a control input. This input pin, called /BS16, is sampled at the end of every bus cycle.

If the signal is sampled low at the completion of a 32-bit bus cycle, the 80386 will ignore the data it received on the upper half of the data bus and perform a second bus cycle to read or write the other half of the data using the lower half of the data bus.

This feature is called dynamic bus sizing and is found on several of the newer 32-bit processors, including the 68020.

If /BS16 is permanently tied low, the 80386 then behaves much like an 80286 with its 16-bit data bus. Beware of 80386 "accelerator" boards which simply replace the 80286 with an 80386 on a daughter board, however. Unless these boards have a much faster clock they will be no faster than the original 80286 system. The only merit might be the ability to begin working on 80386 specific software on such a system.

The 80386 provides the address for the next operation half way through the current bus cycle, using a technique called address pipelining. This early availability of the address allows additional time for system components to get ready for the upcoming bus cycle.

These components (outside the processor) look at the address and decide how quickly the data will be available (i.e. whether they'll have to slow down the processor with wait states).

In an interleaved memory system (many banks) the next data will be available sooner if it's not from the current bank. In a system with two banks of memory, it's faster if it's from the opposite memory bank. If the system has cache memory, the circuitry must figure out whether the new address is already available in cache RAM or if it must be loaded.

This address pipelining may be disabled on a cycle-by-cycle basis through a control input called /NA. When /NA is high, the 80386 generates address timing which is very similar to the 80286.

Memory Systems

The bus bandwidth of the 80386 running at 16 MHz is an impressive 32 megabytes per second. Each bus cycle is a minimum of two processor clock cycles long. This places some interesting requirements on the memory systems used with this processor.

Adding one wait state to the bus cycle reduces the bus bandwidth to 21.33 megabytes per second. Adding two wait states reduces it to 16 megabytes per second. While these numbers are nothing to sneeze at, they show a considerable performance penalty for a poor design. Obviously there are trade-offs in a design, and it is in the design of memory systems that the 80386 machines will differ the most.

There are four basic memory system architectures which are used in such a system: Static RAM, the static/dynamic cached system, the dynamic interleaved system, and the dynamic static column system.

Static RAM offers the highest performance of any memory system for the 80386. With static RAM access times as low as 80 nsec for 8K by 8 parts, the 80386 can run in non-pipelined mode with never a wait state. The power consumption of these devices is low and battery backup may be used as well.

The downfall of such a design is the cost per bit of memory and the physical size of the parts. A 32K by 8 static RAM costs approximately 10 times as much as a 256K by 1 DRAM, even though both devices have the same amount of memory. The static RAM also takes up over four times more board space than dynamic RAM, and ten times as much as the DRAM modules we use at PC Tech. So we won't see much static RAM in PCs.

A static/dynamic cached memory system uses a small amount of static RAM, a large amount of control circuitry, and a large amount of low cost dynamic RAM to provide a very high performance system. System performance is dependent on the amount of static RAM (the size of the cache) and the methods used for cache control. (Cache as cache can.)

A cached memory system expects most memory accesses within a program to be confined to a relatively small area of memory. In fact, it relies on it. This is true of both instruction fetches and general memory references.

A cached system moves data from the slower dynamic RAM into the static RAM when that memory location is first addressed. The address is also stored in fast static RAM. Often several adjacent memory locations will be brought into the static RAM cache.

If the CPU performs a memory operation to a location which has been moved to the cache, a cache "hit" occurs and no wait state is needed. If, however, the operation is a write, both the cache and the DRAM should be updated. This results in a one or two wait state penalty for the write operation.

Cached memory systems can become quite complex, and several types of cache control methods exist. See the table in Figure 4 (from the *Intel 80386 Hardware Reference Manual*, order # 231732-001) which shows the effectiveness of several cached memory systems.

Skeptical Timeout

Personally I'm skeptical. If our machine is doing lots of task switching the cache may frequently be incoherent, resulting in performance degradation. Also consider this: the Motorola 68020 has an on-chip 512-byte cache. According to Intel, the performance of such a system would be less than that of a non-cached DRAM system. Hmmmmm, what do you 68020 guys think of that? I think it's bunk!

The advantages of dynamic memory are high density and low cost. The disadvantage is access time. At a glance one would expect that a 120-nsec access time DRAM should operate with a 16-MHz 80386 (125 nsec per bus cycle) without wait states. But this isn't correct. Unfortunately there is more to a DRAM cycle than just access time.

There's a setup time for the address and a precharge time between accesses. The precharge time is what really slows down the access, requiring as much as 90 nsec for a 120-nsec DRAM. This indicates that even 80-nsec DRAM cannot keep up back to back memory cycles with the 80386. A technique called interleaving helps somewhat.

With interleaved memory, two or more banks of DRAM are arranged so that adjacent words lie in different banks of memory. Since most programs and data are sequential in nature, there's little likelihood of consecutive memory accesses to the same bank. However, when it does happen, the system must add wait states.

Many of the 80386 machines presently on the market use interleaved memory. In a PC Magazine benchmark test, one unit was found to be only slightly faster than a 16-MHz 80286 system. Intel provides a great deal of information on the design of these interleaved memory systems, so I guess it's no surprise that many systems use it.

Within the last year we've seen a new type of 256K by 1 DRAM. This is the static column DRAM. Generally these parts are CMOS technology, so they draw very little power.

They start the memory cycle with a row address, row address strobe, and column address strobe, just like normal DRAM. But the column address can be handled differently. The column address strobe (/CAS) becomes more of a chip select signal, enabling and disabling the outputs of the device.

The access time for a static column DRAM is the same as for a normal DRAM in the normal type cycle. But as long as the row address remains the same, and only the column address is changed, the access time may be very short, typically 70 nsec.

Relatively simple circuitry is needed to detect a required change of row address and insert wait states while the memory controller precharges the DRAM and initiates another cycle. The precharge time for static column DRAMs is often shorter.

The Compaq Deskpro 386 uses static column DRAM with this type of memory controller. A disadvantage of static column DRAM is that it's about twice as expensive as standard DRAM. As these chips become more popular, their price should approach that of standard DRAM.

To optimize performance in such a design, multiple banks of memory could be used for code, data, and stack, so that the DRAM row changes less frequently. The 80386, with its demand paged virtual memory, could well take advantage of this architecture, however the designers of the Deskpro 386 didn't go that far.

At PC Tech we're evaluating a variation on the static column scheme. Most standard DRAMs may be operated in page mode, where many column addresses may be strobed into the DRAMs following the row address strobe, resulting in a much improved access time. The speed appears to be good, but the jury is still out.

Figure 4 - Effectiveness Of Cached Systems

Cache Configuration			Cache Performance	
Size	Associativity	Line Size	Hit Rate	Performance Ratio Over Non-Cached DRAM
1K	direct	4 bytes	41%	0.91
8K	direct	4 bytes	73%	1.25
16K	direct	4 bytes	81%	1.35
32K	direct	4 bytes	86%	1.38
32K	2-way	4 bytes	87%	1.39
32K	direct	8 bytes	91%	1.41
64K	direct	4 bytes	88%	1.39
64K	2-way	4 bytes	89%	1.40
64K	4-way	4 bytes	89%	1.40
64K	direct	8 bytes	92%	1.42
64K	2-way	8 bytes	93%	1.42
128K	direct	4 bytes	89%	1.39
128K	2-way	4 bytes	89%	1.40
128K	direct	8 bytes	93%	1.42
no cache-2 CLK SRAM access			(100%)	1.47
no cache-4 CLK piplined DRAM			--	1.00

Numeric Processors

The 80386 has three different numeric processor options, each with its own level of performance and price tag. The highest price and performance solution comes from a joint venture with the Weitek floating point accelerator people and Intel.

This multi-chip solution, called the Weitek WTL1167, puts a 2.5+ megaflop floating point processor chip together with the 80386 for some fantastic number crunching power. (One megaflop is one million floating point operations per second.) This number crunching is supported in software by Intel's compilers. It is not code compatible with the 8087 or 80287!

The next step down in performance is the 80387, for which data sheets will be available "real soon now." Intel claims this part will be eight times faster than a 5 MHz 80287, which should put it between 0.8 and 1.0 megaflops.

But it won't be available for some time, and in fact may be more expensive than the higher performance Weitek solution. I hope more software folks support the Weitek.

The 80287 may also be used with the 80386. This can provide from 0.1 to 0.2 megaflops. The 80386 can determine the coprocessor type at reset by sampling the /ERROR signal. If the coprocessor is an 80287, the 80386 will automatically convert the 32-bit data to 16-bit data for the 80287.

Other Considerations

Now that we have a fast processor and lots of memory, let's consider a bottleneck which has been a more

(continued next page)

DESIGNING WITH THE 80386

(continued from page 9)

limiting factor than processing horsepower — I/O. I wonder if most people who are so concerned with processing all of their data on their PCs ever consider how slowly that data moves through their system.

With the IBM AT, data can move from hard drive to memory at less than 500K bytes per second. At that rate it would take more than two and a half hours to fill the memory of our 386 4G system. Sure that's a lot of memory, but remember that the 80386 is a virtual machine which can handle up to 64 terabytes of virtual memory — memory which should be moved to and from mass storage devices as the operating system sees fit.

The 80386 can execute several multi-megabyte tasks given less than a megabyte of memory, but oh how slow that will be with a 500K per second I/O data rate. The solution is to use faster hardware for the disk interface, hardware which resides somewhere besides the PC/AT bus (i.e. an on-board SCSI port).

The SCSI port can transfer data as fast as 5 megabytes per second in synchronous mode and 2.5 megabytes per second in asynchronous mode. I think this is the minimum for an effective 80386 machine. Enough said.

Final Words

There is little doubt that the 80386 is a well designed chip and that it can be the heart of a very high performance personal computer. Certainly performance has its price in most all areas.

But don't let 386 fever take hold before you put the machine through some real world tests. Let it earn your respect. Price/performance is still important.

■ ■ ■

A Cheap 68000 Operating System

By Joe Bartel
Hawthorne Technology
8836 SE Stark
Portland, OR 97216
503-245-2005

Update On The 68000 Project

One of the hot topics at SOG V was the continued lack of an inexpensive, standard operating system for the 68000. The processor is cheap, dirt cheap, considering its power. But that lack of a generic system has really stifled its use by individuals.

Joe offered to create a single-user version of his nearly-finished multi-user operating system and to sell it for $50 per copy. Complete. He's been as good as his word and in this article on the operating system he begins a series on the 68000.

The end of July 1986, at SOG V, we demonstrated a large single-board multi-user 68000 system. At that time it would multitask, but it had no native software. There was also almost no interest in it because it was an unknown operating system with no applications software.

However, we were able to use that almost-finished operating system as a model for our single-user K-OS ONE. The two are compatible so that any programs that run with K-OS ONE will run with the multiuser system once it's available. (Next summer.)

Operating System For Experimenters

We put K-OS ONE together for experimenters. With the new ICs like the 68000, it's possible to create power hardware with little effort.

The software, though, is a big problem. When building a 6502 or 6800 system, a simple PROM monitor would be fine. For a Z80 or HD 64180, you can purchase CP/M or one of the work-alikes. There was no cheap, readily available, portable operating system for the 68000 and other large processors. So we wrote K-OS ONE.

We decided it had to be simple, small enough to run in 128K of RAM, and require one (or no) floppy drive. We didn't include low level support for graphics or sound because they weren't standardized hardware. If they're desired, they can be treated as standard devices.

The Amiga, Atari, Macintosh, and Sinclair QL all use the 68000 microprocessor. All of them have operating systems, but none of their software runs on other machines.

Also, none of the systems software is available separately. You get it with the machine. There are a few generic operating systems for the 68000, but for the most part they're expensive or difficult to use or both. The field is not dominated by a single system such as CP/M or MS-DOS.

The Base Operating System

The most common disk format today is the 360K PC variety. Because it's so common we chose to support it in K-OS. Thus we'll have access to a lot of public domain data and source code.

This choice of disk format also means that we can share text files with the PC. We can do our editing on a PC and then use the files directly on our 68000 machine. We can also use any PC based tools such as cross compilers, cross assemblers, debuggers, disk inspectors, and the like.

Though K-OS ONE is not public domain, we provide source code for the base operating system and command processor. That way you can see what's going on and experiment with modifications.

We wrote the system in HTPL, a high level language, to make it easier to understand. If it were in assembler the listing would have been several hundred pages.

Purpose Of The Operating System

The operating system manages system resources and provides a defined hardware interface for programs. Most of the services deal in some manner with storage or I/O.

We divided the functions into four areas: I/O, program control, batch control, and non-disk miscellaneous.

In system design what you leave out is as important as what you put in. A complex system is more powerful but is larger and harder to learn. We wanted a system that was no more complex than CP/M or MS-DOS. Where we could simplify, we did.

System Calls

We used a common format for all system calls. We also used parameter blocks in memory instead of processor registers or a stack, so it's easier to move our design to other processors.

The use of parameter blocks made it easier to write the OS in a high level language. It also means that it's easier to make changes without affecting current applications, and it's generally easier to access system services.

All input and output is device independent. You open a channel by supplying the path name of the file or device. Once you've opened the channel you supply only the channel number and the number of bytes to read or write. This format is familiar to anyone who uses UNIX or MS-DOS. Disk storage is transparent.

Command Processor

We included some functions to make it easy to write a command processor.

K-OS imposes no structure on program area. There are no reserved areas like the PSP in MS-DOS or the facilities below 100H in CP/M.

Programs must call the operating system to get command line arguments (the OS can return any string). A program figures out what its environment is like through system calls.

The user interface part of the command processor is as easy to modify or replace as any standard program. Our command processor is a simple one that looks a lot like MS-DOS. With a little effort you could modify the command processor to look like just about any other operating system.

If you were building a controller, you could entirely eliminate the command processor. Or, for a vertical market system, it could be replaced by a menu — only making available what's needed (and safe).

The current command processor does not support batch processing. (So we could release the package quickly.) It would not be hard to modify the current system to include batches. We have included calls to start a batch file, to get the next line of the batch file, and to get the command line that started the batch file.

Editor

The editor is a simple line editor. Line editors don't need to be installed to be usable. Until someone ports over a really good editor, it's easy to use an MS-DOS machine for words work.

Assembler

The assembler is a simple two-pass absolute assembler. It supports Motorola mnemonics and most standard pseudo ops. It generates a .HEX file so there is no provision for linking or relocation. You can include the assembler's .HEX files in HTPL programs. You can specify include files but not macros. It does have listing control.

HTPL Compiler

The HTPL compiler is a simple two-pass compiler that converts the source program into a position-independent, ready to run, .BIN file.

We supplied HTPL so you could customize the operating system. The purchase or licensing of a standard compiler would have made the cost prohibitive. (General 68000 software isn't cheap yet.) While HTPL is a new language, it's been a very easy language to use. Almost all of K-OS ONE is written in HTPL.

We considered using assembler, Modula-2, C, Pascal, and FORTH. If we had used assembler, the operating system would have been a little smaller and it could have been placed in PROM. We'd also have spent a lot longer developing the system. (We may still use the assembler for special OEM versions.)

The Modula-2 compiler we looked at generated huge files and required lots of memory. Our 256K Sanyo wasn't up to the task. We would have used Small-C if we'd had it, but we didn't. Same problem with Pascal. FORTH is OK, but it's hard to follow and is not well suited to variables. It's also hard for non-FORTHers to understand.

I wrote HTPL after reading DDJ's special issue on FORTH. HTPL has the structure of Modula-2 (which I like a lot). But it doesn't have the overhead garbage.

We had already written the structural part of a conventional compiler. Plus, we had the lexical and symbol table routines. The hard part of writing a compiler is making it produce good code.

An RPN (reverse Polish notation) compiler can generate good code with a simple, compact code generator.

Even though HTPL uses RPN expressions, it is neither threaded code nor an interpreter. It's a true compiler. HTPL has worked out so well that 95% of the system, including the editor, assembler, and compiler, is in HTPL.

The programs are about the same size as they would be if we'd used assembler. Even with a floppy disk, the compiler is fast.

The runtime library for HTPL is written in assembler. We include the source so you can modify or customize it.

Installation

K-OS ONE is available in two forms: a generic form that's easy to port to a new machine; or ready to boot on the TinyGiant board.

If you purchase the portable version you have to install it. This means, you write a driver program for the disk and for the system console. To make it easier we supply a sample BIOS and boot program in assembly language. You can modify these to fit most machines.

First you have to create a monitor or boot PROM for your hardware. Then you load the binary copy of the patchable system into the machine.

Then you patch in the addresses for the I/O drivers and run the operating system. At this point you can edit the source code to make a better custom installation for your system.

Summary

Since SOG V we've had good response to K-OS ONE, and there's an active group of users (including ourselves) working on new software. We will be publishing any changes made to the operating system so everyone will stay up to date.

We're also working on utilities to make the system easier to use. And we're seeing public domain programs showing up already in the 68000 conference on Micro C's RBBS. (503-382-7643, 24hrs, 3-12-2400 baud, 8 bits, no parity, one stop bit.)

Joe has also spent a good deal of time since SOG looking for powerful, but inexpensive 68000 hardware to be a base for K-OS ONE. He's found it, and the complete unit, with operating system, is $395. Contact him for more information.

■ ■ ■

Controlling The Real World With Your PC

By Bruce Eckel
John Fluke Mfg. Co.
PO Box C9090 M/S 266D
Everett, WA 98206

Build A Simple Digital Oscilloscope

This is one of those beginning articles that looks very advanced. It is. But if you'll find a friend, ask questions, and spend some time with Figures 2 and 3, I think you'll be well rewarded. Follow Bruce through this project and you'll be measuring analog signals with the pros.

An oscilloscope draws a picture of a voltage variation in time. Traditionally, they've been made with a cathode-ray tube (like your computer monitor). An electron beam is swept across (from left to right) in a straight line, and the varying voltage of choice is applied to deviate the beam (up and down) to draw the picture.

Project Oscilloscope

Let's build one. To keep it simple we'll limit it to display a bandwidth of around 100 Hz. It won't be very useful, but it will illustrate how to hook up and use an A/D (analog to digital) converter.

The scope will give you a "feel" for what the converter is doing, and how fast it does it.

I've seen many theoretical articles on A/D's. I think hooking one up will make them easier to understand.

The details of the project are for the Kaypro, because I have one, but I'll try to make the concepts generic so the project can be modified for other computers.

A Three Part Project

1. Hook up a parallel input/output chip to your Kaypro 84. (Other computer owners will have to get creative here.)
2. Hook up an 8-bit A/D converter to one of the ports (very simple — just 2 chips!) and test it.
3. Write the software in Turbo Pascal.

The Parallel Port

I actually went out and bought a Z80-based, single-board controller computer just so I could have parallel ports. I had to program it with assembly language, which meant that while experimenting with hardware, I had two problems: to get the hardware to do what I wanted, and to get the assembly language to do what I wanted (three problems, if you include: "is the bug in the hardware or in the assembly language?").

What I needed was Turbo Pascal, so I wouldn't have to think about software problems.

After spending $200+ on the controller board, I found (albeit with knowledge from putting the board together) that I could spend $10 or so and get the same effect (parallel ports) from my Kaypro, and be able to use Turbo Pascal. So, poorer but wiser, I recommend the Kaypro alternative for this project.

All the Kaypro '84 boards except for the Kaypro 4s have a bunch of empty places for clock and modem chips. One of these is the PIO, which has a port for the clock and a port for the modem (Kaypro 4 people are out of luck here). Kaypro also left the chip select gate for the PIO off the board.

So, to get two parallel I/O ports, you need to install the two missing chips and bring a cable from the parallel ports to the outside world.

Here Are The Specifics

First, pull the board out of your Kaypro. This can be a little scary at first. *(What if it doesn't work when I put it back together? What if all the wires look the same?)* But don't worry, it's virtually impossible to put the cables back on wrong. And if pulling the board out and putting it back in makes the computer fail, it was probably frail of heart anyway.

Look for U27 and U35 on the board — you should see the chip outline. U27 is a 74LS138 address decoder, and U35 is a Z80A PIO.

Those of you with '83 Kaypros have two PIOs on board. One port on each is unused. Look for pads on the board marked E7 - E17 (near U54) and E27 - E37 (near U72). These pads are connected to port B on each chip.

You'll notice that all the little holes where you want to stick pins are filled with solder. That's right, you have to get the solder out first. I used a solder sucker, but solder wick might also work. (Ground yourself to the board and use a small, grounded iron.)

Put a chip socket in U27 and plug in the '138. The tricky part about the PIO is getting the 10 wires for each port (8 bits and two handshake lines) from it to your project.

I used a wire-wrap socket (like a normal chip socket, but with very long pins to wrap wires around — they're used in breadboarding) and soldered the wires onto the wire-wrap pins. Then I started the pins into the holes on the board and soldered them there (see Figure 1).

I suggest using a 26-wire ribbon cable (available at Radio Shack). Pull off wire 26 (not the red one — it's useful for orientation). Then connect 2 wires to +5 V and 3 wires to ground, so you don't need an external supply for the project. The other 20 wires carry data between the PIO and the A/D converter.

To make it REALLY nice, take the metal plate off the fan hole (sorry, K10 owners — you have a fan there) and replace it with a similar piece with a hole for a female DB-25 connector. Connectors which clamp onto a 25-wire ribbon cable are also available at Radio Shack.

Now you don't have to have the wire hanging out all the time; just take a male DB-25, clamp it to another piece of ribbon cable and connect that to your project. Different boards can have their own connectors, allowing you to dazzle people by showing them your projects rapid-fire.

What's An A/D Converter?

The A/D converter takes a voltage from the outside world and converts it to a number which is proportional to that voltage. Converters come with various bit widths (also referred to as resolution); usually the more expensive ones have more bits. Eight bits is most common (convenient for an eight-bit bus).

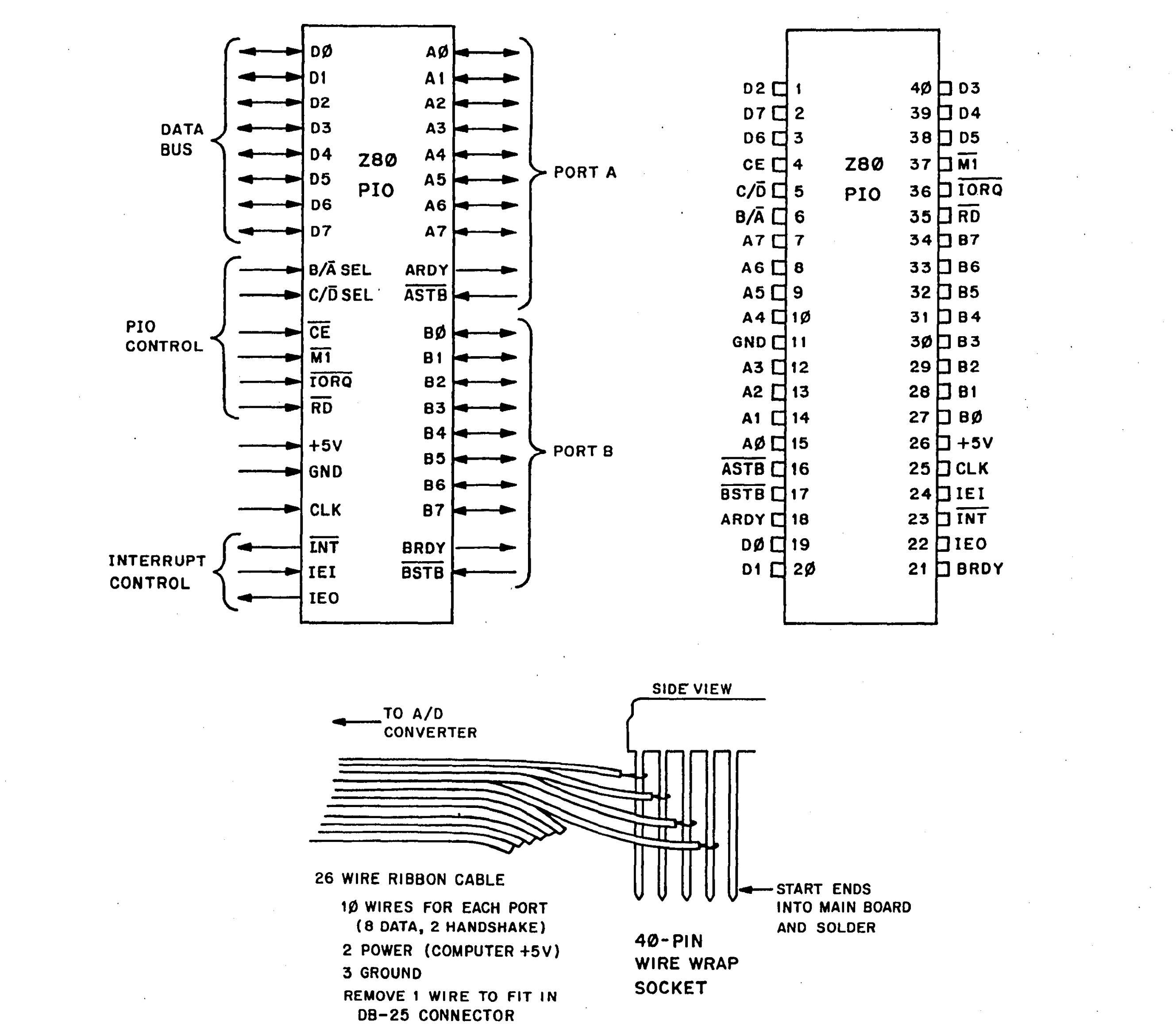
With eight bits, you can divide any voltage range into 2 to the 8th (256) steps. If the input voltage is at the top of the range, the converter will return 255; if it's at the bottom, you'll see 0. The voltage range depends on how you hook up the input to the converter (more on that in the next article). How you interpret the numbers is up to you.

As an example, suppose you hook up the converter so the voltage range is 0-5 V, and you want to know the voltage at the input. Take the input number, divide it by the total number of steps in the converter, multiply it by the range, and add the bottom of the range as the offset (0 V in this case): (input/255) * 5V + 0V = voltage value.

Or, suppose your input is a linear temperature measuring device which gives you 30 V at 30 F and 15 V at 95 F, and you know you'll never go out of that range. Hook up the converter so it returns 0 for 15 V and 255 for 30 V. To calculate temperature from the converter's value: (input/255)*(30F - 95F) + 95F = temperature value. Notice I used 95 F for the bottom of the range since it's the value which causes the converter to return 0.

(continued next page)

Figure 1 - Wiring The PIO Socket (and PIO Pinout)



(continued from page 17)

The Chip

I'm using the National ADC0804 because it's probably the cheapest ($3.95 from JDR) and most common A/D converter (see chip pinout in Figure 2). It's also quite easy to use, partly because it generates its own clock and reference voltage.

The chip needs a clock because it's actually a little state machine. You tell it to start, and it cranks and grinds for about 100 uS, then presents its data and tells you it's done. All that cranking and grinding must be driven by a clock, and we have to provide a resistor and a capacitor for timing components (connected to CLK R and CLK IN on the chip).

The converter needs a reference voltage to measure the input voltage against. The result will be as accurate as the reference voltage. The ADC0804 has a pin called "Vref/2," which means "insert at this pin half of the voltage which will cause the A/D to give a full-scale reading."

Figure 2 - A to D Convertor Circuit

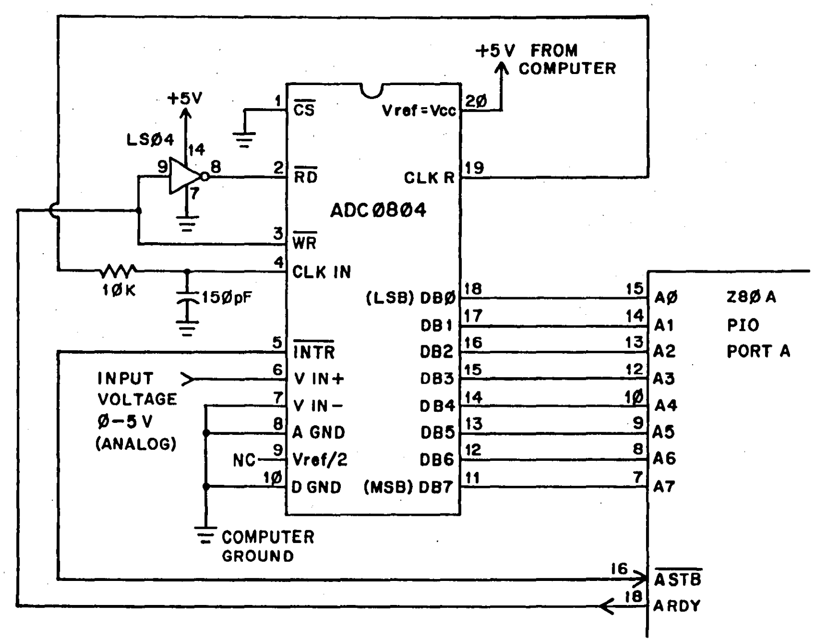
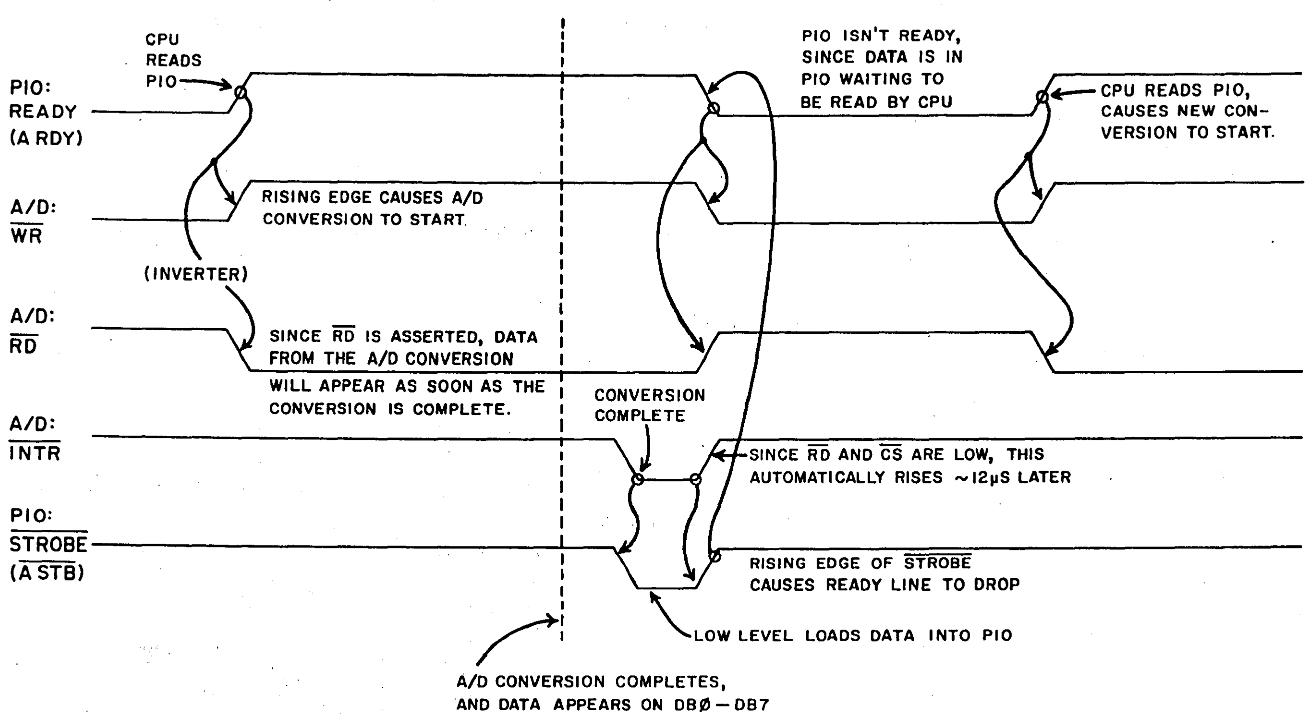


Figure 3 - A to D Timing Diagram



The ADCO804 gives you options: either don't hook up the Vref/2 pin, in which case it uses internal resistors to divide the supply voltage (Vcc) by two, or use some sort of circuit to supply an accurate reference. We'll go the easy route and not hook it up, so our Vref/2 pin will be at 2.5 V (since the supply from the computer is 5 V).

This converter was designed to interface directly with the system bus, and not to go through a PIO. To interface the PIO to the ADC0804, I think it's important to understand the timing of the two chips (but if you really don't want to, skip the next section and just wire it up).

Timing

Figure 3 shows the timing diagrams for the PIO and the A/D. We'll examine the requirements of the A/D converter and see how the PIO is easily adapted.

Since we're not interfacing the A/D directly to the system bus, we can select the chip all the time by tying CS* (I'm using an asterisk here to indicate the line is asserted when low) to ground.

Editor's note:

At this point I found a plea from Margret.

"What are the terms CS, WR, and RD? What is a system bus and what's this about not interfacing to it?"

CS, WR, and RD are control pins. You'll find them on many ICs. CS stands for chip select. When CS is active (in this case when it's low since it's called CS) the chip is active, it expects to receive or send data. What it does depends on the states of its RD (read) pin and its WR (write pin).*

If the line is called RD, then it would be active when its input (voltage) is high (over 4 V). If it's called RD (or has a line over it), it would be active when its input is low (under 1 V).*

We see a rising edge when a line goes from low to high, a falling edge when the line goes from high to low.

A computer's system bus is made up of the internal data, address, and control lines. This A/D interface project won't connect directly to the Z80's busses, but rather will communicate with the processor via a parallel input/output (PIO) chip. The PIO will communicate with the system bus.

In this situation, a new A/D conversion is started by the rising edge of WR*. When the conversion is finished, the chip drops its interrupt (INTR*) line. If the RD* line is low, the INTR* line will rise about 12 uS later.

The RD* line also causes the result of the conversion to be placed on the output data lines of the A/D. Thus, for simplicity's sake it would be most convenient if the RD* line drops as the WR* rises, and stays there throughout the cycle so the INTR* line emits a pulse instead of just going low. Sounds like we want an inverter.

The PIO is a fairly simple beast when used in input mode (mode 1). Each of its two ports, A and B, have two handshaking lines, one which sends signals out (READY) and the other which accepts signals (STROBE*).

When the PIO is empty (after the CPU has read its value), the READY line goes high, to indicate to the peripheral device that it can accept more data. When the peripheral has valid data, it should lower the STROBE* line. This causes the data to be loaded into the PIO. Then STROBE* should be raised, which causes READY to drop (and the PIO to generate an interrupt, if you're using those). When the PIO is read, the READY line rises again and the cycle repeats.

Note that since we're not using interrupts (for simplicity's sake), we have no way of knowing when the data is ready and must simply wait (even Turbo Pascal outruns this A/D converter, though compiled languages are usually slow).

I used the rising edge of the READY line (occurring when the CPU reads the PIO) to start the A/D conversion by raising the WR* line and, via an inverter, to lower the RD* line, thereby enabling the data onto the A/D output lines when it became available, and also causing the INTR* line to reset itself 12 uS after it drops.

The INTR* line is connected to the STROBE* line of the PIO, so when it goes low, the PIO loads the data in, and when it goes high, the PIO drops its READY line (and the WR* line). Thus, when the CPU reads the data

(continued next page)

(continued from page 19)

from the PIO, the READY line (and the WR* line) goes high again, starting another A/D conversion.

The Circuit

As you can see from Figure 2, connecting the A/D converter to the PIO really isn't that difficult once you understand the timing. The necessary inverter brings the chip count up to two.

I suggest using a small superstrip (see your parts house) to breadboard it, and soldering the wires of the ribbon cable onto headers which will just stick into the superstrip (that way you can easily use it for other experiments). Use the power and ground from the computer.

To test the circuit, use a ten-turn potentiometer and hook one end to +5 V, the other to ground, and the wiper as the input to the A/D converter (see Figure 4). The program in Figure 5 will show you if the converter is working correctly.

Note the program ignores the bottom bit. The converter generates its own noise from all that cranking and grinding, so the bottom bit rattles. (Well, it IS cheap.)

Oscilloscope Software

There are several things to know about oscilloscopes before diving into the program in Figure 6.

To be effective, the oscilloscope must give meaningful information. There are many ways to do this, but the most important are scaling and triggering.

Scaling refers to stretching or compressing the picture in the vertical or horizontal direction to show the area of interest. Scaling in the vertical direction means adjusting to different ranges of input voltage; in our simple example we only have one range (0-5 V), so we'll ignore vertical scaling.

Since the beam sweeps from left to right in time, horizontal scaling changes the sweep rate. We can control this scaling by varying the sample rate.

Triggering means looking for a particular attribute of a signal before capturing and displaying it. On analog oscilloscopes, this is usually just a voltage level the scope looks for before starting a sweep.

Figure 4 - Converter Test Circuit

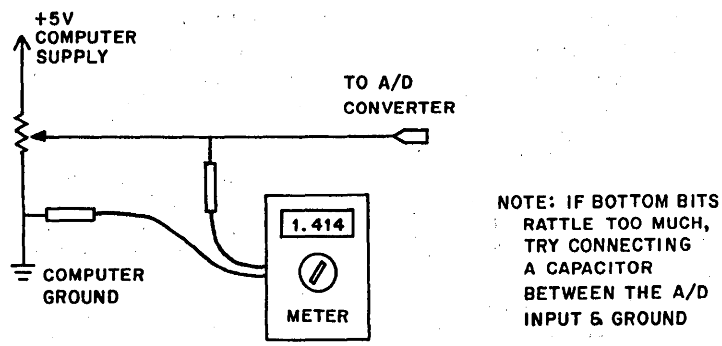
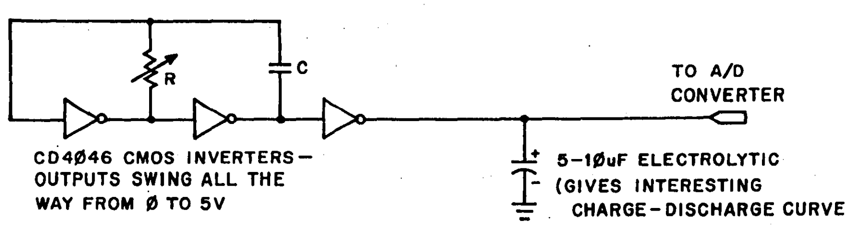


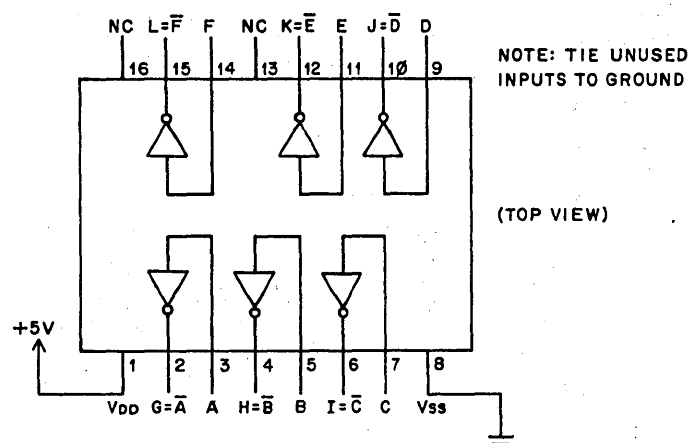
Figure 4a - Oscillator Circuit



R*C SHOULD BE ABOUT Ø.Ø2 (UNITS ARE IN SECONDS)
Example- 1 MEG Ω * Ø.Ø2 uF = Ø.Ø2 S
THIS NUMBER IS ROUGHLY THE PERIOD OF OSCILLATION.

Figure 4b - 4049 Pinout

CD 4Ø49



For instance, if the trigger was set to 1 V, you wouldn't see a trace until the input signal passed through 1 V (you can also choose whether the signal rises or falls through 1 V to cause the trigger). The real advantage of triggering is that it allows you to see a repetitive waveform as if it were just sitting in one place. (If you didn't trigger, it would jump all over the screen.)

But there's a catch — if the event you're looking for occurs before the event which triggered the scope, you're out of luck. Or, if the event just happens once in a while, it will cause the scope screen to flash, and you won't know what the event looked like.

With a digital scope, however, we just zoom along taking samples and sticking them into a buffer. You can do anything (within time constraints) to that data to decide when to trigger (for instance, look for a particular pulse width).

(A pulse is a short high on a line that spends most of its time low, or a short low on a high line. Pulse width is how long the pulse lasted.)

When the trigger occurs, you can display all the points which occurred UP TO the trigger if you want. If it's a one-shot event, you can just leave the results on the screen for the user to examine. (I didn't implement this on this project. "Left as an exercise for the trusting reader.")

So the scope consists of the A/D converter, the procedure which samples until the trigger conditions are met (for simplicity I've used the same kind of level-detecting triggering as analog scopes use), the procedure which gathers the data, and the procedure which formats and outputs the data to the screen.

This is all packaged together with a user interface which allows the time scaling (interpreted in this case as sample rate), trigger level, and other parameters to be changed.

I've tried to use good programming practices and show passing of all variables which are modified, but in one case (point_buffer in procedure get_samples), passing the array by reference caused an irregular time delay when loading it with samples from the A/D converter (it probably required an extra level of indexing). It would be nice if Turbo could show the assembly code it generates so we could see why these things happen.

Figure 5 - Oscilliscope Test Program

```
program test; {oscilliscope tester for Kaypro 2X.  By Bruce Eckel 8/86}

const
     A_CONTROL: byte = $22;          {Control and Data I/O locations}
     A_DATA: byte = $20;             {for each of the Z80 PIO ports}
     B_CONTROL: byte = $23;
     B_DATA: byte = $21;

     MODE : byte = $4f;              {0100 1111  mode 1 = input}
     INT : byte = $07;               {0000 0111 interrupts disabled}
var
     temp,val : byte;

function noise_test : boolean; {this keeps the screen from
flickering because of the noise in the bottom bit of the A/D
converter}
begin
     if (              { part of what this function does is }
     (temp <> val)     { stall the program while the A/D    }
     and               { converter catches up }
     (temp <> (val+1)))
     then noise_test := true else noise_test := false;
end;
begin
     port[A_CONTROL]:= MODE;    {initialize port A of the pio}
     port[A_CONTROL]:= INT;     { port B is done the same way }

ClrScr; write(#27,'C4'); { Turn kaypro cursor off }

{ Turbo's "port array" is used to write to the PIO.  The first
one starts the A/D converter.  Note that most of the programming
effort is just to make the screen output nice.  The following
line is really all there is to running the A/D converter. }

val := port[A_DATA];

repeat begin
       temp := port[A_DATA];
       if  noise_test then begin
              val := temp;
              GotoXY(30,10); clreol; write('binary ',val, '     ');
              { Note I take the value, divide it by the number of
               steps, and multiply by the voltage range (my power
               supply doesn't quite make it to 5.0).  The ":4:3"
               is formatting information for Turbo.  Check the
               results with your multimeter. }
             write(((val/255) * 4.99 ):4:3, ' volts');
             end;
       end
until keypressed;
write(#27,'B4');  { Turn cursor back on }
end.
```

Testing

To test the scope, use the circuit in Figure 4. I've specified a CMOS inverter because its outputs will go all the way from 0 V to 5 V. You can use the rest of the inverters in the 74LS04 (see Figure 2) if you don't have CMOS handy, but you won't get a trace which goes from the top to the bottom of the screen.

The 10 uF capacitor gives an interesting charge/discharge curve to look at. If you leave it out, you'll just get a square wave.

If you increase the delay between samples past the point where the original wave is discernible, and keep going, you'll begin to see patterns which look like the original wave but are sort of repeated on top of each other. This is called aliasing and occurs because you aren't taking samples fast enough to get all the information about the wave.

(continued on page 24)

Figure 6 begins on the next page

Figure 6 - Oscilliscope Program - A/D

```
program scope; { oscilliscope program for A/D example. By Bruce Eckel 8/86 }
type
     direction = (FALLING, RISING); { trigger on rising or falling edge }
     binary = (ON,OFF);          { whether to turn a pixel on or off }
     pixel_control = string[2]; { how to turn pixel on or off        }
     screen_buffer = array[0..159] of integer; { Kaypro screen is 160
                                                  pixels wide }
const
     A_CONTROL: byte = $22;      {Control and Data I/O locations}
     A_DATA: byte = $20;         {for the Z80 PIO port A }
     MODE : byte = $4f;          {0100 1111  mode 1 = input}
     INT : byte = $07;           {0000 0111 ints disabled}
     min_capture_band : integer = 3; {minimum triggering search window}
var
   CH : char;  trigger_edge : direction;
   low_capture_bound, high_capture_bound : integer;
   sample_rate, pause_rate : integer;
   point_buffer, pixel_buffer, old_pixel : screen_buffer;
   loopcntr, dly, dlycntr : integer;
   cursor_off, cursor_on : string[3];
   pixel_off, pixel_on : pixel_control;
   dim, bright, reverse_video, normal_video : string[3];
   revdim,normal: string[6];

procedure terminal_customization; { should make it a little easier to}
begin                             { customize for other computers.   }
     cursor_off := #27 + 'C4';
     cursor_on  := #27 + 'B4';
     pixel_off  := #27 + ' ';
     pixel_on   := #27 + '*';
     dim := #27 + 'B1';            { If you don't have these features, just }
     bright := #27 + 'C1';         { set the strings to ''.                  }
     reverse_video := #27 + 'B0';
     normal_video := #27 + 'C0';
     revdim := reverse_video + dim;
     normal := normal_video + bright;
end;

procedure pixel(on_or_off:binary; vertical_coord, horizontal_coord : integer);
const
     vertical_offset : integer = 131;  { the kaypro's screen doesn't start
                                         at 0,0}
     horizontal_offset : integer =32;  { these offsets start it in lower
                                         left corner}
var pixel_char : pixel_control;
begin
     if (on_or_off = ON) then pixel_char := pixel_on else
                                        pixel_char := pixel_off ;
     write(pixel_char, chr(vertical_offset - vertical_coord),
                       chr(horizontal_offset + horizontal_coord));
end;

procedure refresh_screen(var new_screen, old_screen : screen_buffer);
{ displays contents of point_buffer while erasing old trace on screen. }
var pixel_counter : integer;
begin
     for pixel_counter := 0 to 159 do begin  {Kaypro screen width again...}
         pixel(OFF, old_screen[pixel_counter], pixel_counter);
         pixel(ON, new_screen[pixel_counter], pixel_counter);
         old_screen[pixel_counter] := new_screen[pixel_counter];
         end;
end;

procedure ADC_delay (multiplier:integer); { Tried using assembly language
here, but the overhead of the INLINE statement overwhelmed my timing loop.
Adjust this until you get something from your A/D converter. }
var i,j: integer;
begin for i := 1 to multiplier do begin j :=0; j:= 1; j:=2; end; end;

procedure trigger(edge: direction; low_bound, high_bound : integer);
var presentval,lastval : integer; slope, edge_not : direction;
begin
     edge_not := direction(ord(edge) xor 1);  { invert edge }
     slope := edge_not;
     presentval := port[A_DATA];
```

```
     while (slope = edge_not) do begin   { wait for the right direction }
           while (not ((presentval > low_bound)  { ... and range of values }
                    and (presentval < high_bound))) do begin
                 ADC_delay(1);
                 presentval := port[A_DATA];
                 end;
           lastval := presentval;
           ADC_delay(1); presentval := port [A_DATA];
           if (((edge = RISING) and ( presentval > lastval)) or
               ((edge = FALLING) and ( presentval < lastval)))
           then slope := edge;
    end;
    delay(dly); { trigger delay in milliseconds }
end;


procedure get_samples(rate :integer);
{ tried passing the point_buffer array by variable, but it slowed things
down enough to make the data look bad }
var counter : integer;
begin
     for counter := 0 to 159 do begin
         point_buffer[counter] := port[A_DATA]; { point_buffer is global }
         ADC_delay(rate);
         end;
end;

procedure process_samples(var input_point, output_point : screen_buffer);
{ put samples in a form which can be displayed }
var index : integer;
begin
     for index :=0 to 159 do             {steps in A/D conv \/}
         output_point[index] := (trunc ((input_point[index]/255)*99));
end;                                    {vertical steps on screen /\}

procedure menu; { too many variables to bother passing -- all changed globally}
var i : integer;
begin
   clrscr; gotoxy(1,3);
   writeln('      ',revdim,'Digital Oscilliscope Options Menu:',normal);
  writeln;
   writeln('    t : change trigger delay.  Current delay = ',
                   revdim,dly,normal,' mS');writeln;
     write('    e : rising or falling trigger edge : ',revdim);
   if trigger_edge = rising then writeln('RISING',normal)
   else writeln('FALLING',normal); writeln;
   writeln('    o : change trigger offset : ',revdim,low_capture_bound,normal);
   writeln;
   writeln('    c : change trigger capture band : ',revdim,
                     high_capture_bound - low_capture_bound, normal);
   writeln;
   writeln('    s : change sample rate.  Current rate = ',
                   revdim,sample_rate,normal);writeln;
   writeln('    p : change pause rate : ',revdim,pause_rate,normal);
writeln;
   writeln('          <ESC> : quit');writeln;
   writeln('              any other key returns to sampling');
   while(not keypressed) do; read(kbd,ch);
   if (CH in (['A'..'Z'] + ['a'..'z'] )) then begin
      gotoxy(18,1);
      case CH of
       't','T': begin write('New trigger value : '); readln(dly); end;
       'e','E': begin write('New trigger edge(0 for falling,1 for rising): ');
                readln(i);
                trigger_edge := direction(i); end;
       'o','O': begin write('New offset : '); readln(low_capture_bound);
                if (high_capture_bound - low_capture_bound < min_capture_band)
                then high_capture_bound := low_capture_bound+ min_capture_band;
                end;
       'c','C': begin write('New capture band : '); readln(i);
                if (i < min_capture_band) then
                   high_capture_bound := low_capture_bound + min_capture_band
                   else high_capture_bound := low_capture_bound + i;
                end;
       's','S': begin write('New sample rate : '); readln(sample_rate); end;
       'p','P': begin write('New pause rate : '); readln(pause_rate); end;
      end;
   CH := ' ';
  end;
  clrscr;
end;
```

(continued next page)

(continued from page 21)

Next Time

In the next article, I'll show you how to condition analog signals BEFORE they get to the A/D converter. This will be an introduction to analog electronics, and will handle problems like:

1. How to make an op-amp do what you want. (An op-amp is a very high gain analog amplifier.)
2. Extra-large and extra-small signals.
3. What to do when you have many signals to measure, but only one A/D converter (electronic switching).
4. Frequency limitations of sampling and the "sampling theorem" (how to cope with aliasing).
5. What an analog filter is; how to make a simple one.

■ ■ ■

Figure 6 - Oscilliscope Program (continued)

```
procedure pause_for_input;  { user can change parameters via menu here }
begin
  gotoxy(1,1);write(reverse_video,dim,'PAUSE',normal_video,bright);
  for dlycntr := 1 to 20000 do if (keypressed) then read(KBD,CH);
  if (CH <> #27) and (CH <> ' ') then menu;
  gotoxy(1,1); write ('     ');
end;

{******************** main ***********************}
begin
     port[A_CONTROL]:= MODE;   {initialize the pio}
     port[A_CONTROL]:= INT;

terminal_customization;
write(cursor_off);
dly := 18; low_capture_bound := 2; high_capture_bound := 5;
sample_rate := 1; pause_rate := 5; trigger_edge := RISING;
clrscr;  CH:=' ';
while(CH <> #27) do begin
  for loopcntr := 1 to pause_rate do begin
      trigger(trigger_edge, low_capture_bound, high_capture_bound);
      get_samples(sample_rate); { point_buffer changed as a global here }
      process_samples(point_buffer,pixel_buffer);
      refresh_screen(pixel_buffer, old_pixel);
  end;
  pause_for_input;
end;

write(cursor_on);
end.
```

Concurrent Operating System

Build It With Modula-2

By Brad Justice, Stan Osborne, & Vivian Wills
Computer Science Department
San Francisco State University
San Francisco, California 94132

OK, guys. You want to shake loose the cobwebs? Find out what it's really like in CS401? Burned out on that recipe finder you've been fighting in Pascal?

Step right up! Here's something that'll impress both your boss and your mother-in-law.

Modula-2 was cooked up for writing operating systems, and here's the recipe for doing it. A very interesting look at the guts of a concurrent operating system that you can run on your clone.

Operating Systems have long been considered one of the most difficult areas in computer programming. But in the last few years new analytical methods and software development tools have made it much easier to build operating systems.

We used some of these new tools and methods in a section of the "Operating System Principles" class taught at San Francisco State University in the Spring Semester of 1985.

Our primary goals were to provide students with an understanding of operating system components and practical experience in the design of an efficient and maintainable system application.

In particular, we devised a project that required student interaction, the interface of software to existing hardware devices in a non-standard way, and the use of a preemptive, multiprocessing operating system kernel.

This article describes the operating system kernel and application system developed using the kernel.

Simultaneous File Transfer

Our target application was a file transfer utility, which would permit multiple files to be copied simultaneously to different devices while being controlled from a single menu of options.

We wanted the menu be displayed on one part of the screen while data from a file was displayed on another part. So two or more processes had to do I/O to the screen simultaneously, with each process having its own display window.

We chose Modula-2 because it allows concurrent programming, interface to device interrupts, and the management of software modules developed by a group of programmers.

The system kernel and file transfer application were developed on IBM PCs, with 256K RAM, 2 floppies, monochrome cards, and monitors.

Our application uses the device driver ANSI.SYS, PC-DOS, and ROM BIOS routines. All file operations (open/close, read/write, etc.) and the reading of the current time of day use Modula-2 library routines (supplied with the Logitech compiler). Whenever possible the application has used Modula-2 procedures rather than direct calls to DOS or ROM BIOS.

In the few cases where we needed better performance we used Logitech Modula-2's inline assembly language features to call DOS or the BIOS directly. In particular, we called the BIOS for the printer output routines and the windowed video output.

To run our application on a different hardware system would require changing those parts that are dependent on the BIOS, DOS, or written in assembly language. The operating system kernel has no instructions which are hardware or PC-DOS dependent. The device driver and the application modules are the modules that contain hardware and system dependent routines.

Operating System Kernel

The operating system kernel consists of Modula-2 procedures, types, variables, and processes that schedule the execution of concurrent processes.

When concurrent processes are executed simultaneously on two or more processors, we get true concurrency. But more commonly, these processes are executed on a single processor that switches between the processes. Let's call this "quasi-concurrency."

The IBM PC (or PC, for short) has a single processor, so true concurrency can't occur. But processes can execute quasi-concurrently. One process is given the processor, and some of the instructions of the process are executed.

A process may become blocked, meaning it encounters some condition that makes it impossible for it to continue doing useful work. If so, the current process relinquishes the processor to another process. If the process is never blocked, it will continue to execute until its time limit expires.

If a process becomes blocked, a hardware clock is used to interrupt it and start an interrupt process in the kernel. The interrupt process then gives the processor to a second process. This pattern repeats; the second process then executes until it too relinquishes the processor or times out. Eventually, the first process is again given the processor, and it resumes from where it was interrupted.

The usual alternative to this approach is, of course, sequential programming.

A kernel allows the IBM PC to act like a time-sharing system, where multiple processes occur simultaneously. The processor is switched back and forth between processes so rapidly that it appears as though each has its own processor.

The big advantage of concurrency is that multiple tasks can run simultaneously. For example, our application can interact with the user while it transfers files.

Also, the system is more efficient since it avoids "busy waiting" (waiting for input from a user, for example).

In a sequential program, we could

use a software loop, which repeatedly tests to see whether the event has occurred.

In a concurrent program, the waiting process relinquishes the processor, allowing another process to execute and do useful work. When the awaited event occurs, the waiting process resumes execution. So, while one process waits, the others execute.

In addition to sharing the processor among many processes, the kernel provides the mechanism for controlling access to other shared resources. These resources may be hardware — such as a printer, or software — such as service routines or shared data.

The Kernel

Modula-2 provides many high level features useful in writing a system kernel. Several of these are crucial:

PROC — A data type. A PROC is a parameterless procedure.

PROCESS — A standard type for a process. In Modula-2, a PROCESS is a coroutine for use with a kernel that supports many processes sharing one or more processors.

NEWPROCESS(P:PROC; A:ADDRESS; n:CARDINAL; VAR new:PROCESS); — A standard procedure to create a process. NEWPROCESS is passed "P," the address of the instructions to be executed; "A," the location for the PROCESS workspace; and "n," the size of the workspace. It returns "new," the process created.

ALLOCATE(VAR a:ADDRESS; size:CARDINAL); — A standard process to allocate memory. For example, we need a workspace when NEWPROCESS is called. So, ALLOCATE is passed the size of the area to be allocated, and returns "a," the location of the area allocated.

DEALLOCATE(VAR a:ADDRESS; size:CARDINAL); — The opposite of allocate, DEALLOCATE is passed the address and size of an area in memory to be made available for reuse.

TRANSFER(VAR p1,p2:PROCESS); — A standard procedure to transfer control of the processor to a process. There are two parameters: a source and a destination process. When source PROCESS p1 calls TRANSFER, its execution is suspended and the execution of destination PROCESS p2 is resumed at the point where it was last interrupted.

IOTRANSFER(VAR p1,p2: PROCESS; va:CARDINAL); — Similar to TRANSFER with one important difference: this routine receives control of the CPU from a hardware interrupt. To handle this, it uses a third parameter, "va," the interrupt vector value. On execution, IOTRANSFER passes control between a source and a destination PROCESS. In addition, the source PROCESS p1 is installed as an interrupt handler at the vector "va." The occurrence of the interrupt will cause the source PROCESS (the interrupted process) to resume at the instruction immediately following IOTRANSFER.

Pointers

The kernel uses these Modula-2 facilities along with standard pointer operations for the creation, scheduling, and destruction of concurrent processes.

For the kernel, a new concept of a process is required, different from the Modula-2 PROCESS — a PROCESS plus a process descriptor. The process descriptor is a RECORD that stores information for the scheduling and eventual destruction of the process.

The definition of the process descriptor is —

```
TYPE Processdescriptor =
      RECORD
         Next: Pdpointer;
         Cor: PROCESS;
         Corsize: CARDINAL;
         Sleepcount: CARDINAL
      END;
```

where Pdpointer is a POINTER to a Processdescriptor.

"Cor" can be thought of as a pointer to the PROCESS to which the Processdescriptor applies. "Next" can be used to build linked lists of Processdescriptors.

"Cp," the current process, is a Pdpointer.

The kernel and its relationship to the other components of the system is represented in Figure 1.

Queue Management

An Eventqueue is a linked list of Processdescriptors that are waiting for some event. It's defined as:

```
TYPE Eventqueue = ARRAY
[Top..Bottom] OF Pdpointer;
```

Two Pdpointers are used; one points to the top of the queue (where processes are generally removed from the queue), and one points to the bottom of the queue (where processes are usually added).

There are two Eventqueues used by the kernel — the Ready queue and the Sleepqueue.

The Ready queue consists of processes waiting for the processor. The Sleepqueue consists of processes that have elected to remain inactive for a certain period of time, and are waiting to be awakened by the kernel.

As many Eventqueues can be created as are required by an application program; we'll use many such Eventqueues in our application.

Four operations can be done on TYPE Eventqueue:

Awaited(S:Eventqueue): BOOLEAN; — Used by a process to test an Eventqueue for a Waiting process.

Init(VAR S:Eventqueue); — Initializes the eventqueue.

Signal(VAR S:Eventqueue); — When a process calls Signal, it starts the top process in the Eventqueue S. If Ev-

(continued next page)

(continued from page 27)

entqueue S contains no processes, Signal is a null operation. Otherwise, the top process is removed from the Eventqueue and placed at the bottom of the Ready Queue. The next process in the Eventqueue, if any, becomes the top process.

Wait(VAR S:Eventqueue); — The process calling Wait relinquishes the processor and enters Eventqueue S.

Processes

We call Startprocess to start a process, passing it two parameters: Size, a CARDINAL; and Routine, a PROC.

Size is the amount of memory that must be allocated for the execution of the new process, and Routine is the address of the instructions that will be executed.

By calling ALLOCATE, we can allocate memory locations for the Processdescriptor and the PROCESS. The PROCESS is created through a call to NEWPROCESS. And finally, the new process is placed at the bottom of the Ready queue for eventual execution.

Specifying the workspace size is one difficulty of using Modula-2 for coprocessing. It can be difficult to assess in advance the space requirement of a process. Specifying a space too large wastes memory, while specifying a space too small can cause the system to crash.

The new process waits in the Ready queue while those processes ahead of it are removed from the queue and executed. Eventually it rises to the top of the queue. It's removed from the queue and becomes the Cp, the current process.

It's then allocated the processor, and its instructions are executed. It continues to execute until it times out or calls one of the four PROCEDURES that result in its relinquishing of the processor to the next Ready process. These four PROCEDURES are:

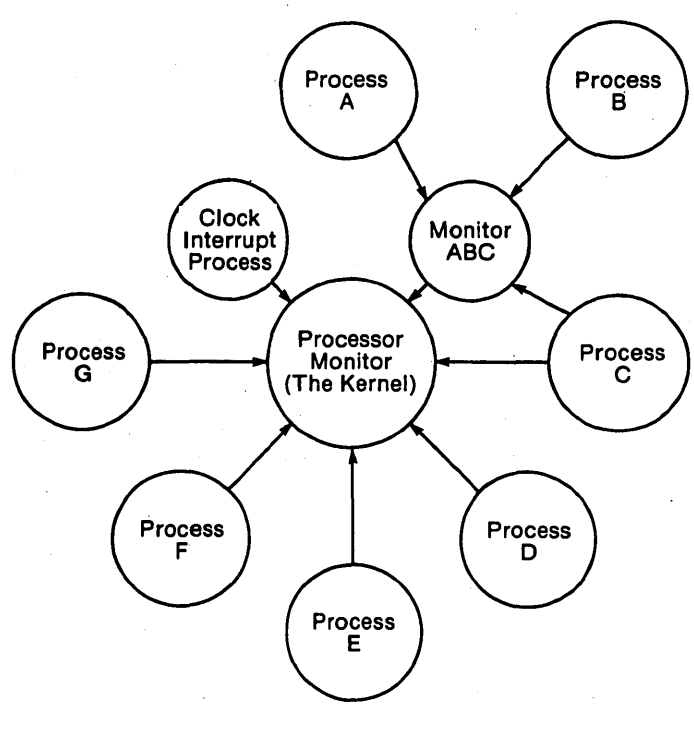
Finishprocess; — The calling process is destroyed, its memory space is made available for reuse through calls to DEALLOCATE, and the process at the top of the Ready queue becomes the current process.

Pause; — The calling process is placed at the bottom of the Ready queue, and the process at the top of the Ready queue becomes the Cp (current process). A call to TRANSFER allocates the processor to the new current process.

Sleep(Count:CARDINAL); — The calling process is placed in the Sleepqueue where it remains for Count clock pulses.

Wait(VAR S:Eventqueue); — The calling process is placed at the bottom of Eventqueue S. The process at the top of the Ready queue becomes the Cp. A call to TRANSFER allocates the processor to the new current process.

Figure 1 - Relationship Of System Elements



Device Interrupts

To keep things from becoming too simple, a kernel must also contend with unscheduled processes. These processes (usually called interrupt handlers) aren't executed because they go to the head of the Ready queue. Instead they're executed in response to an external unscheduled event caused by hardware. When an interrupt occurs, execution of the current process is suspended, and the interrupt handler process is executed.

Interrupt handlers aren't typical processes (like those previously discussed). They have no process descriptor. They wait in no Eventqueue for processing. They cannot be timed out. And, on completion of their task, they do not pass control through a call to TRANSFER; they use IOTRANSFER instead.

(Note: Following the recommendations of Logitech, an effort was made to keep to a minimum the instructions executed by the interrupt processes. This makes particular sense with the clock handler, which is executed 18 times per second. Sometimes the interrupt handler does tasks that could be done through a call to kernel procedures. Instead the instructions are included in-line. This is done to avoid the overhead inherent in a procedure call.)

Clock Device Interrupt

One interrupt handler is required for the execution of the operating system kernel: the clock interrupt handler. Other interrupt handlers can be created as required for the application, for example a keyboard interrupt handler to interpret keyboard input.

The clock interrupt handler provides the time slicing for the kernel. It knows when a process has used up its allotted time period. In this case the current process is preempted if any other processes are in the Ready queue. This interrupt handler also provides the mechanism for managing the Sleepqueue.

Clock Interrupt

The clock interrupt handler is a PROCESS. On the IBM PC the clock interrupt occurs eighteen times each second. Each time the interrupt occurs the instructions of the interrupt handler are executed.

The clock handler counts the number of clock pulses (clock interrupts) that have occurred since the current process began. When the count reaches the limit, the process stops. The clock handler places the current process at the end of the Ready queue and removes the top process of the Ready queue and makes it the Cp. Finally the clock handler executes Figure 2, transferring control to the new current process.

General Timer Service

The clock handler also manages the Sleepqueue. Every time the clock interrupt occurs, the clock handler decrements the Sleepcount of the top process in the Sleepqueue. When the Sleepcount reaches zero the sleeping process is started by removing it from the Sleepqueue and placing it in the Ready queue.

Insertions in the Sleepqueue are handled so only the Sleepcount of the top process needs to be decremented. A Process enters most Eventqueues by calling Wait, and is inserted at the end of the queue.

To see how a process is inserted in the Sleepqueue, let's follow the insertion of three processes in an initially empty queue: the first for six clock pulses, the second for fifteen, and the third for ten.

When the first process calls Sleep(6), it's placed at the top of the Sleepqueue with a Sleepcount of 6. See Figure 3.

The second process calls Sleep(15), and the parameter (15) is compared with the Sleepcount of the first process. Since the second process is to sleep for a longer period, it's placed in the queue after the first process. The Sleepcount of the first process, 6, is subtracted from the parameter, leaving 9. Since there are no other processes, it is placed immediately after the first process with a Sleepcount of 9. See Figure 4.

The third process calls Sleep(10), and the parameter is compared to the Sleepcount of the first process in the queue. Since 10 is greater than 6, the third process will be placed behind the first. The Sleepcount of the first is subtracted from the parameter, leaving 4. This is compared to the Sleepcount of the next process of the queue, 9. Since 9 is greater than 4 the new process will be placed in the queue before this process. It's entered in the queue with a Sleepcount of 4, and the Sleepcount of the process immediately following is decremented by the Sleepcount of the new process (4), making it 5. See Figure 5.

At this point there are 3 processes in the queue. The first process is the first in the queue, with a Sleepcount of 6. The third process is the second in the queue, with a Sleepcount of 4. The second process is the third in the queue, with a Sleepcount of 5. Each time a clock pulse occurs, the Sleepcount of the top process in the queue is decremented.

After 6 pulses the Sleepcount of the first process reaches 0, and it's removed from the Sleepqueue and placed in the Ready queue. The third process is now at the top of the queue, so each clock pulse its Sleepcount is decremented. After four clock pulses its Sleepcount is 0 and it's started. This leaves the second process. For five clock pulses its Sleepcount is decremented, it reaches 0, and the last process is started.

The first process slept for 6 clock pulses, the third for 6 + 4, or 10 clock pulses, and the second for 6 + 4 + 5, or 15 clock pulses. All processes were inactive for the desired period of time, and the updating required for each clock interrupt by the clock handler is reduced to updating and testing a single variable.

(continued next page)

Figure 2 - Invoking New Process After Time Out

```
IOTRANSFER(clkhandlerP,Cp^.Cor,Clkintvec)
```

Figure 3 - One Process In The Sleepqueue

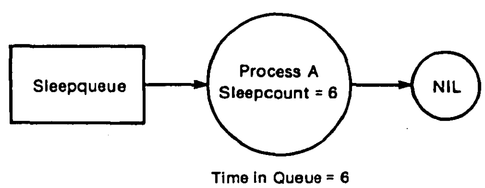


Figure 4 - Adding To The Sleepqueue

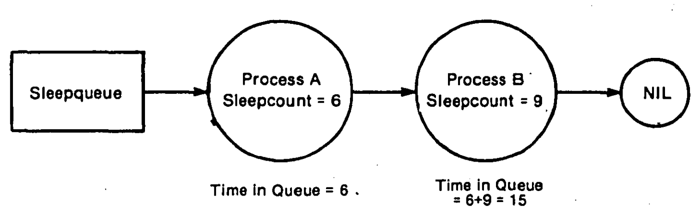
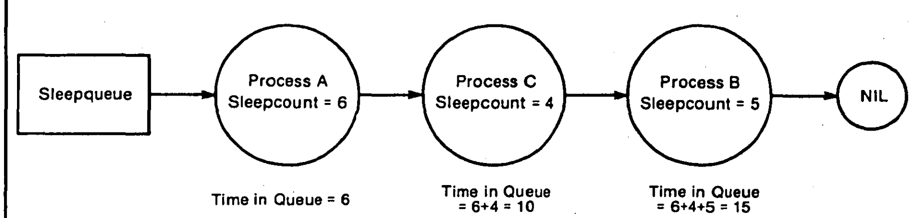


Figure 5 - Inserting A Process Into The Sleepqueue



(continued from page 29)

Countlock

One other utility provided by the clock handler is Countlock. By setting Countlock := TRUE, a process can stop preemption, and the clock handler is locked from timing the process out. The process has the processor until it executes Countlock := FALSE. This turns off the Countlock, allowing preemption.

There are tasks that cannot be interrupted without the possibility of erroneous results. An example might be pointer operations on a linked list. Countlock is one way to insure that a process isn't timed out before the links have been properly rebuilt.

Kernel operations occur with the Countlock on. The Countlock can be invoked by the application as required. One such use of Countlock is the creation of primitive (i.e. may not be interrupted) instructions for semaphores. (Or was it sophomores?)

Semaphores

A semaphore is a flag used by two or more concurrent processes to coordinate access to a shared resource, insuring that only one process has access to the resource at a time.

There are two primitive instructions required for a semaphore — P(s) and V(s) (named after the terminology used by E. W. Dijkstra). The operations required for P and V must be done without interruption if the results are to be guaranteed.

Countlock enforces this restriction with clock interrupts by preventing time out preemption of the current process during the critical sections of P and V.

In our application, we had to be sure that, at any given moment, only one process was using DOS. So P and V are incorporated in the monitor that controls access to DOS.

We use two variables: DOSinuse, a BOOLEAN; and DOSqueue, an Eventqueue. A process calls P before using DOS. The following instructions implement P —

```
Countlock := TRUE;
  IF DOSinuse THEN
  Wait(DOSqueue)
  ELSE
    DOSinuse := TRUE;
    Countlock := FALSE
  END;
```

Flag DOSinuse is checked to see if another process has called DOS. If not, flag DOSinuse is set to TRUE and the process calls DOS. If DOSinuse is already TRUE when P is called, the calling program must wait until the other process relinquishes DOS. It waits in Eventqueue DOSqueue until signalled.

When a process is no longer using DOS it calls V —

```
Countlock := TRUE;
IF Awaited(DOSqueue) Signal(DOSqueue)
ELSE
DOSinuse := FALSE;
Countlock := FALSE END;
```

V checks to see if anyone is waiting to use DOS. If so, that process resumes execution (with access to DOS).

DOSinuse isn't reset to FALSE because DOS is now allocated to the signalled process. If no process is waiting for DOS, DOSinuse is set to FALSE and DOS is now free to be allocated to the next process calling P.

Without the Countlock, P and V may generate incorrect results. For example, Process A wishes to use DOS and calls P. It tests DOSinuse and finds it's FALSE. It then times out.

Process B wishes to use DOS. It too calls P and finds DOSinuse to be FALSE since Process A timed out before it could reset the flag. So, Process B sets DOSinuse to TRUE and proceeds to use DOS. While doing so, it too times out. And Process A resumes execution. It has already tested DOSinuse and still thinks DOSinuse is FALSE. It too sets DOSinuse to be TRUE and proceeds to use DOS.

Because Process A was interrupted between testing and setting flag DOSinuse, the semaphore has failed and two processes are now using DOS.

With the Countlock on, preemption by the kernel is impossible and the semaphore can't fail. In this case Countlock is combined with kernel procedures to create a higher level tool.

Limitations

1. There's no way to have priority access to resources. So, there's only a simple type of mutual exclusion. A process blocks another process from access to a critical section by blocking all other access to the CPU. This can cause performance problems when the time spent in a critical section is long.

2. There's no way for a lengthy interrupt process to lower its priority, so other interrupts may occur. Therefore interrupt processes must be designed carefully.

3. Processes are scheduled using a simple round robin algorithm.

4. The hardware clock interrupt is 17 Hertz. For many applications, this doesn't provide a small enough interval for time outs or shared access to the system with time slicing.

5. Switching between processes takes significant CPU time. Most of the time is lost because IOTRANSFER must be inside the TRANSFER loop. This extra overhead will become less important as processors become faster.

Interactive applications don't switch from interrupt processes to other processes with enough frequency for this overhead to be a problem. Applications that process interrupts frequently must consider the amount of CPU time needed to switch from one process to another.

Our Application System

The Interactive Data Transfer System (IDTS) is an application system using the kernel routines to program simultaneous file transfers between various devices on the PC. Files may be copied from disk to printer, disk to screen, and disk to disk. A split screen holds the menu and keyboard I/O on the top half of the screen.

Keyboard I/O and file transfers are all concurrent operations. While one file is writing to the printer, another may be writing to a disk, and a third file may be writing to the screen. The menu remains active during all file transfers, so the user may type further requests at any time. File transfer requests for active output devices are held in queues and processed as devices become free.

Concurrency in the IDTS is handled by coroutines (called processes in Modula-2). True concurrency occurs when two or more processes are being executed by two or more processors at the same time. The processes in the IDTS are executed by the single processor of the PC in quasi-concurrency. All active processes are given their share of running time by the kernel's Scheduler routines.

Our application system is represented in Figure 6.

System Initialization

When a user starts the IDTS, several things happen before the menu comes up.

During Modula-2's module body initialization phase, three interrupt handlers are installed by the process Inithandlers in the module Intrupts. The Scheduler uses Startprocesses to place Inithandlers in the Ready queue.

Inithandlers installs the Clock interrupt handler (Timeout) used by the Scheduler, and the Keyboard interrupt handler (KBhandler) used by the Menu. Inithandlers also installs a simple Critical Error interrupt hander (CEHandler) that causes the IDTS to break and return to DOS on a fatal error.

Then the screen clears and the menu is written onto the top half. When the module body finishes its initialization, the main program module, Main, calls Startprocess (menu). Then it calls all the file transfer processes (ReaddskP, Toprint, ReaddskS, Toscreen, ReaddskD, Todisk). Finally Main calls Startsystem.

Inithandlers installs the three interrupt handlers and calls Finishprocess. Inithandlers then disappears. Storage space is freed for other processes. Menu and the file transfer processes all Wait in Eventqueues until they're called into action by the appropriate Signal.

The Keyboard interrupt handler responds to a PC hardware interrupt. It's in a high priority module so it's protected from other processes.

The handler routine responds to every key pressed by putting the key scan code into a simple buffer. Then it Signals to the Menu that a scan code is available in the buffer. When it's finished, the keyboard handler IOTRANSFERs back to the process it interrupted.

The module KBHandler contains and exports procedures for reading the buffer and interpreting the key scan codes.

Menu

MENUPROC (Menu Process) reads and processes key scan codes from the keyboard buffer using procedures from KBhandler. It echoes keyboard input on the menu (top) half of the screen. Menu manages cursor control and printing effects (e.g. reverse video) in the menu window via procedures from the module VideoHndlr (Video Handler).

File transfer requests (from the keyboard) also come through Menu. When a request is accepted, MENUPROC starts file transfers using the module DEVmon (Device monitor) for device request queue management.

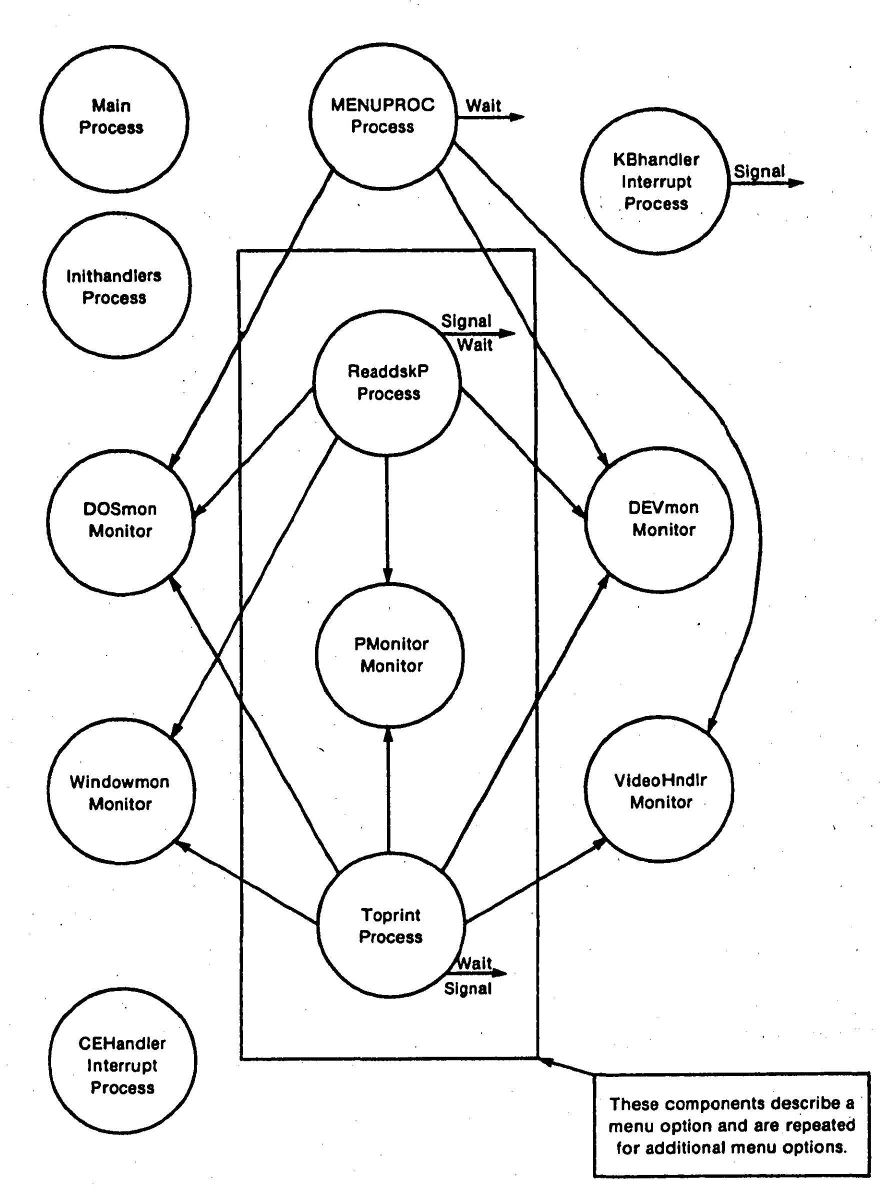
MENUPROC puts the requested file name in the correct device queue and Signals the start of the Read. The Read process then Signals the Write process for the device.

For example, if the user sends a file to the printer then MENUPROC Signals PrinterFreeIn (an Eventqueue for the Read-for-printer process, ReaddskP). The process is put in the Ready queue. ReaddskP, in turn, Signals PrinterFreeOut (an Eventqueue for the Write-to-printer process, Toprint). Now both processes are in the Ready queue where their execution time is managed by the kernel.

File transfers are handled by three separate modules per device:

(continued next page)

Figure 6 - The Entire System



(continued from page 31)

1. A module with a process that reads from the disk and writes to a buffer (e.g. ReaddskP).

2. The buffer and its read and write procedures, Put and Get, are safe in a monitor module (e.g. SMonitor) which is only accessed by the two processes that need it.

3. A module with a process that reads from the buffer and writes to the output device (e.g. Toprint).

Each Read/Write pair of processes runs concurrently by calling the kernel's Signal and Wait.

When a file transfer is complete (EOF), both processes check to see whether any more requests are left in their device queue. If the queue is not empty, both processes Wait, then Signal each other back into action.

A user may request that the IDTS stop when all jobs are complete. When a process finds its queue empty and all other jobs complete, it stops the IDTS and exits to DOS. Otherwise, each process Waits in its Eventqueue for a Signal.

I/O Management

PC-DOS is not reentrant. So, as we discussed earlier, a process must be protected while it's using DOS.

Any process that uses DOS, either directly (using the DOSCALL statement in Logitech Modula-2), or indirectly (e.g. using file commands, disk access, or writing to the screen), uses DOSmon (DOS monitor).

DOSmon protects DOS by allowing only one process at a time to access DOS. Whenever a process calls DOSmon it sets a BOOLEAN, DOSinuse, to TRUE. All other processes that request DOS are put in an Eventqueue by DOSmon, using Signal and Wait from the Scheduler.

DEVmon manages device requests. It insures that only one file at a time is being written to a given device. DEVmon contains and manages the device request queues. A user may request file transfers on already busy devices. These will be started when the current transfers are completed. File requests are kept in first come first served queues.

Wrap Up

Modula-2 has proven itself to be an excellent language for building interactive applications that require system programming.

Separate Definition and Implementation modules guarantee a clearly defined interface between software components. The separate compilation of modules prevents unnecessary recompiling of procedures that haven't changed.

In our project, students familiar with Pascal were quickly able to learn Modula-2 (they were able to complete a complex project in less than ten weeks of part time effort).

A copy of the source files for the Modula-2 software described in this article may be obtained on a 48/TPI IBM PC floppy disk. To obtain the "Modula-2 Kernel & Application" software, send a check or money order for $15 to —

The Association for Computing Machinery
Student Chapter
C/O Computer Science Department
San Francisco State University 1600
Holloway Avenue, Room TH 906
San Francisco, CA 94132

The Kernel and Application are also available for downloading from the Micro C bulletin board (503-382-7643).

We want to thank those folks who helped on this project and the preparation of this paper: John Barr and Phil Rosine of the University of Montana, in Missoula, Montana, for the first version of a kernel; Jeff Clymer for his help during the programming of the class project; Christopher R. Cale, Maurizio Gianola, Alfred Moertlseder, of Logitech, Inc., in Redwood City, California, for donating Modula-2/86 software and technical help; and John Copeland, Howard Ensler, and Brian Hart, of Image Network, in Mountain View, California, for providing technical help and access to laser printers and typesetting software. (Whew!)

Recovering Directories And FATs On 360K Disks

By David Thompson
Staff Hacker &
File Retriever

Winning At Lost & Found

If you stay with me all the way through this massive epistle, you should be able to recover just about anything from a 360K disk. (This is VERY salable knowledge, by the way. Just let half a dozen dealers in your area know that you specialize in recovering data from MS-DOS disks and watch business fall in the door.)

Also, this project costs you nothing: DISKCOPY, FORMAT, and DEBUG come with MS-DOS.

'Twas a week before Christmas and all through the house not a creature was stirring, not even the Logitech mouse. The kids in their nightgowns were all snug in their beds, while visions of video games danced in their heads (seen a sugar-plum lately?). I, in my cap, had settled in at the computer, when from the next room came such a clatter:

"Help."

It sounded a lot more like Sandy than Santa.

"Help!!"

"Rough night with the elves?" I asked.

"Ho ho ho!" she said. She wasn't smiling.

Sandy had been trying to upgrade her Champion from version 1.00 to version 1.05. Something told me it wasn't working. (Champion is the accounting package we're using to keep Micro C's books.)

Mistake

Sandy gets nervous whenever she has to change anything in the system. She gets particularly nervous when upgrading because it seems that there's always a glitch in the process, a glitch that shuts down the system for a week or two. Eventually the Champion folks come up with a solution and Sandy gets going again, but catching up means a marathon at the keyboard.

The upgrade process was really very simple. All she had to do was copy 104TO105.BAT onto her Champion directory (on the winnie) and enter:

```
C>104TO105 A
```

where 'A' is the name of the floppy drive that would hold the upgrade disks. Unfortunately, she entered:

```
C>104TO105 A:
```

The result wasn't the best. The first try didn't work. When she tried again, this time without the offending colon, the installation program complained about a missing file and quit.

Recovery

That's when I got called in. (I never could resist a damsel in distress.)

After poring over the .BAT (batch) file that oversaw the whole operation, I discovered that, sure enough, the program looked for TEMPSYS.DBF. And, sure enough, there didn't appear to be any such animal.

Obviously I was going to have to find it, or Sandy'd be calling Champion and getting behind and...

Anytime I mess with a directory, I make a track copy of the disk and mess with the copy. (If you're going to modify the directory on a hard disk, start with two sets of backups and make sure you know how to do a low level reformat. Just in case.)

The reason I make a track copy (I use DISKCOPY under MS-DOS) is that plain old COPY copies only the unerased files. A track copy program duplicates every bit (literally) of data. If all the files on the original have been deleted, then they'll be deleted on the copy. But they'll be there to undelete.

What happens when you delete a file? If you were in CP/M, you'd see that the first character in the directory entry gets changed from '00' to 'E5'. That's it. (All numbers are hexadecimal.)

Under MS-DOS, the first character in the file's directory entry is the first character of the file name. When the file is erased, that first character is changed to E5. Under DEBUG, an unerased directory entry looks like Figure 1.

Erased, it looks like Figure 2.

Well, there it was. I used DEBUG to change the E5 to 54 (ASCII for T) by doing:

```
-E 280
```

to (E)nter the byte at 280H. Debug responded with 'E5' and I entered '54<cr>'.

Before The Save

Before I write the information back onto the disk, let's see where the information came from.

I read the information into memory with the DEBUG command:

```
-L 0 1 5 7
```

This (L)oads data into memory beginning at memory offset 0, from drive 1 (drive B), starting with sector 5 (the beginning of the directory), 7 sectors (the entire directory on a 360K disk).

Then I entered:

```
-D 0
```

This displayed 128 bytes of data beginning at memory offset 0. Each additional

```
-D
```

displayed 128 more bytes. So I 'D'ed along until I found my erased file. Then I changed the E5 to 54 using the (E)nter command, and saved the whole mess with:

```
-W 0 1 5 7
```

to (W)rite 7 sectors of information

beginning on sector 5 on drive 1. It takes the information beginning at memory offset 0. Easy. (Notice that the format for W is the same as L.)

Then I exited DEBUG and, sure enough, my file was back. All 160 bytes of it. Sandy tried the installation again (with 'A') and it worked!

I had learned something. Sandy was happy. Everything was hunky dory. Well, it was almost hunky dory.

FAT

I've heard a lot about fat. It's squiggly, cushiony stuff that collects where you least need it. Fat is easy to understand and hard to change. MS-DOS disks have FAT too, a FAT that's more difficult to understand but easier to change.

FAT stands for the file allocation table.

Under CP/M, directory entries contain not only the name of the file, but also pointers to all the data blocks it occupies. (Up to 16 of them anyway.) If a file uses more than 16 blocks (usually 1K each), then the file is assigned more than 1 directory entry.

Under MS-DOS, the directory entry only points to the first block of data (called a cluster by Microsoft). The FAT contains the additional pointers, if any.

Fortunately, my TEMPSYS.DBF file was shorter than 1024 bytes. If it had been longer than 1K, I wouldn't have recovered all of the file because I hadn't yet discovered the FAT.

Discovery

After discovering the FAT and realizing that I didn't have the slightest notion what it did, I dug out the *Programmer's Guide To The IBM PC* by Peter Norton. I also grabbed Earl Hinrichs' *Debugging A Disk* article in Micro C issue #33, and the PC-DOS 3.0 *Disk Operating System* manual put out by IBM.

After browsing through Earl's piece, I read the FAT sections in Norton and IBM. Boy, everything you'd ever want to know about FAT was in those two books. Everything.

But I didn't understand it.

Show And Tell

So, instead of the old college brute force method (read it over and over and over, ad sleepium), I dug out DEBUG and discovered for myself how the FAT works.

If you have a clone and a spare disk, there's no excuse for not following along with me. It'll be an adventure, just you and me and your computer probing the darkest secrets of an MS-DOS disk. So brew a fresh cup of tea, dump the cat out of the computer chair, and boot up the system.

Now place your sacrificial disk in drive A and a disk with DEBUG, FORMAT, and a few small files in drive B. (DEBUG, FORMAT and the small files could reside on the winnie instead of B; it makes no difference.)

FORMAT the A disk (with no system) and run DEBUG. (See Figure 3; spurious remarks in parentheses are mine.)

The first block of data is the empty directory. The second is the file alloca-

(continued next page)

Figure 1 - Normal Directory Entry

```
1972:0280  54 45 4D 50 53 59 53 20-44 42 46 20 00 00 00 00   TEMPSYS DBF
1972:0290  00 00 00 00 00 00 59 78-4C 0C 1C 01 A0 00 00 00   ......YxL..
```

Figure 2 - Erased Directory Entry

```
1972:0280  E5 45 4D 50 53 59 53 20-44 42 46 20 00 00 00 00   eEMPSYS DBF
1972:0290  00 00 00 00 00 00 59 78-4C 0C 1C 01 A0 00 00 00   ......YxL..
```

Figure 3 - Directory And FAT

```
B>FORMAT A:

B>DEBUG
-L 0 0 5 1     (Load into 0, drive 0, beginning at sector 5, 1 sector)
-D0            (Dump 128 bytes beginning at 0)
               (sector 5, the first directory sector, follows)

1972:0000  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   .vvvvvvvvvv
1972:0010  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   vvvvvvvvvvv
1972:0020  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   .vvvvvvvvvv
1972:0030  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   vvvvvvvvvvv
1972:0040  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   .vvvvvvvvvv
1972:0050  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   vvvvvvvvvvv
1972:0060  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   .vvvvvvvvvv
1972:0070  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   vvvvvvvvvvv

-L 0 0 1 1    (beginning at sector 1, load 1 sector)
-D0           (dump out first 128 bytes of the file allocation table, FAT

1972:0000  FD FF FF 00 00 00 00 00-00 00 00 00 00 00 00 00   }..........
1972:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
```

(continued from page 35)

tion table (FAT). The FAT byte at location 0 (FD) tells the system that the disk is double-sided double-density. The FF FF in bytes 1 and 2 are unused. The 00s which follow, will hold cluster information, 12 bits per cluster. (12 bits? Arghhhhh!)

Let's write a 1K (or smaller) file onto our test disk and see what happens. (Mine is called ONE-K.TXT, see Figure 4.)

Hey, we've got some data! The directory contains ONE-K.TXT and the FAT contains three more F's. Figure 5 shows a breakdown of the directory entry.

This is our only entry. Numero uno. The bytes 0 - 10 (decimal) hold the file's name. Byte 11 (with the asterisk over it) is the file attribute byte. In this case, only bit 5 is set. (The vertical bars show where a data area starts and stops.) Figure 6 shows the meanings of all the attribute bits.

Note: If you had copied the system files onto this disk during formatting, you'd have noticed that their attribute bytes were 27 hex (bits 0,1,2, and 5 set). That is, they are read-only (bit 0), hidden (bit 1), system files (bit 2), and they haven't been backed up (bit 5).

All those 00's that follow the attribute byte are "reserved by DOS." (That means Microsoft hasn't thought of anything to do with them.)

Time & Date

Immediately following the 00's we have two bytes of time and two bytes of date. (I can hear it now, "Do you have time for a byte or two?")

I didn't bother figuring out the time or date, but it basically requires extracting bits and dividing them up into odd-sized binary numbers. (It's a task that DIR handles quite well, so I'll pass.)

If you're reconstructing a directory entry and want the file to remain in this century, just copy the time and date bytes from a valid directory entry.

Cluster Pointer

The two bytes above "clust" point to the first (and in this case, only) data cluster. This file begins in cluster 0002. Not 0200 as you might first assume. The second byte (00) is the most significant 8 bits, the first byte (02) is the least significant 8 bits. You put them together with the most significant on the left and you get 00 02 or 0002.

File Size

The final four bytes contain the file's size (in bytes). Again, we have to rearrange things a bit. The first byte (80) is the least important, the last byte (00) is the most. So, the hexadecimal size of this file is 00 00 02 80 or 00000280 or 280. Since 280 hex equals 640 decimal, our file is 640 bytes long.

These length bytes are important. The system will read this file until it's read 640 bytes (the number in the

Figure 4 - Directory and FAT With Single 1-K File

```
B>COPY ONE-K.TXT A:

B>DEBUG
-L 0 0 5 1
-DO
1972:0000  4F 4E 45 2D 4B 20 20 20-54 58 54 20 00 00 00 00   ONE-K   TXT
1972:0010  00 00 00 00 00 00 25 24-21 00 02 00 80 02 00 00   ......%$!..
1972:0020  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   .vvvvvvvvvv
1972:0030  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   vvvvvvvvvvv
1972:0040  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   .vvvvvvvvvv
1972:0050  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   vvvvvvvvvvv
1972:0060  00 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   .vvvvvvvvvv
1972:0070  F6 F6 F6 F6 F6 F6 F6 F6-F6 F6 F6 F6 F6 F6 F6 F6   vvvvvvvvvvv

-L 0 0 1 1
-DO
1972:0000  FD FF FF FF 0F 00 00 00-00 00 00 00 00 00 00 00   }..........
1972:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0020  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
1972:0070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
```

Figure 5 - Descreption Of Directory Entry

```
(         |O  N  E  -  K              T  X  T |* |  reserved    )
1972:0000  4F 4E 45 2D 4B 20 20 20-54 58 54 20 00 00 00 00   ONE-K   TXT
1972:0010  00 00 00 00 00 00 25 24-21 00 02 00 80 02 00 00   ......%$!..
(               reserved        |Time  |Date  |clust| file size  | )
```

Figure 6 - File Attribute Description

```
File attribute bits:

0 - If bit 0 is set, the file is read only.
1 - If bit 1 is set, the file is hidden.
2 - If bit 2 is set, it's a system file.
3 - If bit 3 is set, it's a volume label (not a file name).
4 - If bit 4 is set, this entry defines a subdirectory.
5 - If bit 5 is set, the file has not been backed up
                     since it was last opened and closed.
```

length field), or until it reaches an end of file marker in the data (^Z for text files), or until it runs out of clusters in the FAT. Whichever comes first! (We'll cover the FAT in a moment.)

If we reduce this number from 280 hex (640 decimal) to 100 hex (256 decimal) and copy the file to another disk, we'll only copy over the first 256 bytes. Try it!

If we change the length number from 280 to 1000 hex, we won't get 4096 (decimal) bytes in our new file. At most, we'll get 1024.

Let's look at the FAT (Figure 7) to see why.

We've already discussed the first three bytes (FD FF FF). Cluster 2 contains FFF which means that cluster 2 is the final cluster in this file. (Each cluster is two sectors which total 1024 bytes.) Just as the data for cluster 2 (the first cluster) will always lie in disk sectors 0C and 0D, its FAT entry will always lie in the fourth byte (FF) and the lower half of the fifth byte (0F) in the table.

Deletions

This is where I began making noises like: "Wow, I see how it works!" Let's erase ONE-K.TXT. (See Figure 8.)

Ok, the directory data is still intact: everything as it was except that the 4F ('O') in ONE was changed to an E5.

If we change the E5 to any ASCII character that MS-DOS allows in a file name, the file shows up on the screen again and we can copy it, erase it (again), or whatever.

Since this file fits into a single 1024-byte block (cluster) this is all you need to do in order to copy the file onto another disk.

Notice, though, that the FAT is empty again. The pointer in cluster location 2 is 000. When we copy another file onto this disk, its data will be written into the first free cluster (2). Thus its data will be written over ONE-K.TXT.

Let's leave ONE-K erased and copy a 2K file onto the disk. See Figure 9.

COMDEX.TXT is 00000500 hex bytes long (1280 decimal) — just long enough to use two clusters. Notice that its pointer bytes still contain 0002. This file starts in cluster 2 (Figure 10).

FAT cluster 2 now contains 003. It's saying that cluster 2 is occupied (contents greater than 000) and that more of the file resides in cluster 003. Thus, cluster 2 contains a pointer to the next cluster (cluster 3). Cluster 3 must contain either a pointer to another cluster or FFF. In this case it's FFF (last cluster in the file). I wonder what would happen if cluster 3 pointed back to cluster 2 and the file length were very large? (Oh, perverted mind!)

Now, let's copy D.COM to the disk. (See Figure 11.)

Note that D.COM begins in cluster 00 04 (4) and its length is 00 00 0E 00 (E00), which is 3584 decimal bytes. Let's look at the FAT in Figure 12. Clusters 2 and 3 still contain 003 and FFF. That's still COMDEX.TXT. Clus-

(continued next page)

Figure 7 - FAT Description With 1K File

```
                                   clusters
                     { | 2        3 | 4       5 | 6       7 | 8       9 |  ...}
                     (  FFF     000  000    000  000    000  000    000    )
1972:0000   FD FF FF FF 0F 00 00 00-00 00 00 00 00 00 00 00    }...............
1972:0010   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ................
```

Figure 8 - Directory Entry And FAT After ONE-K.TXT Has Been Erased

```
A>ERASE ONE-K.TXT

A>B:

B>DEBUG
-L 0 0 5 1
-D0 1F
1972:0000   E5 4E 45 2D 4B 20 20 20-54 58 54 20 00 00 00 00    eNE-K   TXT
1972:0010   00 00 00 00 00 00 25 24-21 00 02 00 80 02 00 00    ......%$!..

-L 0 0 1 1
-D0 1F
                     { | 2      | }
                     (  000       )
1972:0000   FD FF FF 00 00 00 00 00-00 00 00 00 00 00 00 00    }..........
1972:0010   00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00    ...........
```

Figure 9 - Directory With COMDEX.TXT Replacing Erased ONE-K.TXT

```
B>COPY COMDEX.TXT b:
B>DEBUG
-L 0 0 5 1
-D0 1F
1972:0000   43 4F 4D 44 45 58 20 20-54 58 54 20 00 00 00 00    COMDEX  TXT ....
1972:0010   00 00 00 00 00 00 28 1B-21 00 02 00 00 05 00 00    ......(.!.......
(                                    |clust| file size |  )
```

(continued from page 37)

ters 4 through 7 contain D.COM.

Now let's erase COMDEX.TXT. Figure 13 shows the directory.

To recover COMDEX.TXT we can change its E5 to 43 (in byte 0000), but this file is longer than 1 cluster, so to recover all its data we'll have to get into the FAT (Figure 14).

Clusters 2 and 3 say they are unused. Cluster 4 still points to 5, 5 still points to 6, 6 to 7, and 7 still contains FFF.

From the directory entry for COMDEX.TXT we know that it begins in cluster 2. We also know the file is 1280 (500 hex) bytes long. So it resided in two clusters. Even if we couldn't cheat and look back at the previous FAT this would be easy. Clusters 2 and 3 are free (erased), the directory entry points to 2, and the file only uses two clusters.

(E)nter byte 3, debug responds with 00, key in '03', and hit the spacebar. DEBUG responds with the next byte, 00, and you enter 'F0', spacebar. DEBUG responds with the final 00 and you enter 'FF' followed by a carriage return to end the session.

Finally, look at your handiwork. Figure 15 shows the entire process.

And then write it back into the FAT.

-W 0 0 1 1

and

-W 0 0 3 1

Hold on a minute, you say. The first write (w 0 0 1 1) wrote the FAT, right? Yep, the FAT that begins in sector 1. There are two FATs on every 360K disk — one beginning on sector 1, the other on sector 3. The second write updated the second FAT.

Your file is back. Completely back.

Additional Projects

Hopefully you now have a pretty good feel for the directory structure. For more advanced training, try deleting one file from a full disk. (Use DEBUG and screen dump to print out the directory and FAT before the deletion.) Now try to recover the file without peeking at the printout. Do both the directory and the FAT.

Notice that if you only unerase a

Figure 10 - FAT With COMDEX.TXT (2 Clusters)

```
-L 0 0 1 1
-D0 1F
                         ( | 2     3 | )
                         (  003   FFF  )
1972:0000  FD FF FF 03 F0 FF 00 00-00 00 00 00 00 00 00 00   }...p......
1972:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
```

Figure 11 - Directory With Two Entries

```
B>COPY D.COM B:
B>DEBUG
-L 0 0 5 1
-D0
1972:0000  43 4F 4D 44 45 58 20 20-54 58 54 20 00 00 00 00   COMDEX  TXT
1972:0010  00 00 00 00 00 00 28 1B-21 00 02 00 00 05 00 00   ......(.!..
1972:0020  44 20 20 20 20 20 20 20-43 4F 4D 20 00 00 00 00   D       COM
1972:0030  00 00 00 00 00 00 A1 04-21 00 04 00 00 0E 00 00   ......!.!..
```

Figure 12 - Directory After COMDEX.TXT Has Been Erased

```
-L 0 0 1 1
-D0 1F
                         { | 2     3 | 4     5 | 6     7 | }
                         { |003   FFF|005   006|007   FFF| }
1972:0000  FD FF FF 03 F0 FF 05 60-00 07 F0 FF 00 00 00 00   }...p..'..p
1972:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
```

Figure 13 - FAT With Erased COMDEX.TXT

```
B>A:
A>ERASE COMDEX.TXT
A>B:
B>DEBUG
-L 0 0 5 1
-D0 3F
1972:0000  E5 4F 4D 44 45 58 20 20-54 58 54 20 00 00 00 00   eOMDEX  TXT
1972:0010  00 00 00 00 00 00 28 1B-21 00 02 00 00 05 00 00   ......(.!..
1972:0020  44 20 20 20 20 20 20 20-43 4F 4D 20 00 00 00 00   D       COM
1972:0030  00 00 00 00 00 00 A1 04-21 00 04 00 00 0E 00 00   ......!.!..
```

directory entry and then copy another file to the disk, the second file will begin in the first available cluster, usually the same cluster that was occupied by the "unerased" file. The second file's data will not only be written over the first file's, but you will also have two filenames pointing to the same beginning cluster.

Delete either file and you'll be writing zeros into the entire cluster chain they point to.

Scrambled Directories & Fried FAT

Every once in a while, something will scramble a directory or eat a FAT. I haven't found a program that's smart enough to recover from such major trauma. You can, however.

If the first FAT is gone, try the second. If it's OK, write its information back into the first.

If the directory has been zapped but you still have a FAT, then print out the FAT and mark all the file entry points. (Entry points are all those clusters that aren't pointed to by other clusters. They often follow FFFs.) Then create enough directory entries (use debug to copy samples from another disk and then edit to fit).

I use filenames like A, B, C, D... until later when I can figure out what they are. Let's say you have a 5 cluster-string starting at cluster 2. We'll call it file A (See Figure 16).

The filename is A. The next ten 20's are spaces filling out the filename and extent. The 20 with the asterisk over it is the file attribute byte. On the second line, a borrowed time and date.

Finally, it points to cluster 2 (00 02) and it says the file is 5K bytes long (that's 00 00 14 00 hex).

That's all there is to it.

Bad FAT

If the directory is fine but the FAT's on the fire, then you've got a very different, more tedious project.

Take a large sheet of paper and mark spaces for as many FAT entries as you will need (maximum is about 350). Then mark the entries that are pointed to by the directory and how many clusters each needs.

Those entry points that are spaced exactly the right distance apart (there's

(continued next page)

Figure 14 - Recovering The Cluster in the FAT

```
-L 0 0 1 1
-D0 1F
                   ( | 2    3 | 4    5 | 6    7 | )
                   (  000  000  005  006  007  FFF  )

1972:0000  FD FF FF 00 00 00 05 60-00 07 F0 FF 00 00 00 00   }......'..p
1972:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........


-E 3
1972:0003  00.03   00.F0   00.FF
```

Figure 15 - COMDEX.TXT Restored in the FAT

```
-D0 1F
1972:0000  FD FF FF 03 F0 FF 05 60-00 07 F0 FF 00 00 00 00   }...p..'..p
1972:0010  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ...........
```

Figure 16 - Example Dummy Directory Entry

```
(          A                                   *              )
1972:0000  41 20 20 20 20 20 20 20 20 20 20 20 00 00 00 00
1972:0010  00 00 00 00 00 00 28 1B-21 00 02 00 00 14 00 00
(                           |time | date|clust|   size    | )
```

Figure 17 - How The 360K Disk Is Organized

```
        B  F1 F1 F2 F2 D  D  D  D  D  D  D  Da Da...
Sector  0  1  2  3  4  5  6  7  8  9  A  B  C  D ...

Where

B  - Boot Sector
F1 - First FAT
F2 - Second FAT
D  - Directory
Da - Data (C & D contain the data for cluster 2)
          (E & F contain the data for cluster 3)
          (10 & 11 contain the data for cluster 4)
          .
          .
          .
```

RECOVERING DIRECTORIES

(continued from page 39)

just room for the file) probably represent complete files, in order. Mark these first. After this, try to thread the remaining files into the remaining holes.

Then test, test, test. Check each recovered data file, .EXE file, and .COM file to be sure there are no surprises. (Do it with a copy!) You might well have to peek at the data sectors to see what makes sense. (See Figure 17.)

Finally, if you're up against bad sectors in the directory track, you'll have to try to make a track copy (DISKCOPY). Sure, DISKCOPY will complain loudly, but you have to be able to write your fixes onto the disk. If a directory sector has a hard error you'll have to find a way to move as much as possible to another disk (even DEBUG will help).

And finally, a toast to the next year: may your computer run forever, your business prosper, and may all your FAT problems be light.

■ ■ ■

ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

DIGITAL RESEARCH COMPUTERS (214) 225-2309

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 MOTHER-BOARD 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.

NEW!

DEMONSTRATION SOFTWARE AND A LIBRARY BUILDING PROGRAM ARE INCLUDED ON A 5¼ INCH PC/XT DISKETTE. FULL DOCUMENTATION AND SCHEMATICS ARE ALSO INCLUDED.

69.95 COMPLETE KIT — **89.95** ASSEMBLED & TESTED

HARD DISK CONTROLLER CARD

THE FANTASTIC KONAN KDC-230. WITH MANUAL. SUPER EASY TO CONFIGURE AND USE.

HALF SIZE PCB. $99.00 — ADD $8.00 FOR CABLES

MEX-PC MODEM SOFTWARE PACKAGE $49.95

A FANTASTIC COMMUNICATIONS PACKAGE WITH FEATURES TOO NUMEROUS TO LIST. SEE MAY/JUNE '86 MICRO SYSTEMS JOURNAL FOR FULL REVIEW OR CALL FOR BROCHURE. SUPPORTS COLOR, KERMIT PROTOCOL, Y MODEM BATCH, VT100 EMULATION, REMOTE OPERATION, ON LINE HELP, ETC. RUNS UNDER PC/MS DOS. 180 PAGE MANUAL & BINDER.

MEX-PACK LIST IS $99.95 — **SPECIAL $49.95**

HOT COMPUTER BOOKS!

GET THE MOST OUT OF THE INVESTMENT YOU MADE IN YOUR COMPUTER HARDWARE AND SOFTWARE WITH THE NEW WORDWARE SERIES OF COMPUTER REFERENCE AND TRAINING MANUALS. LEARN HOW TO MAKE ALL THAT EXPENSIVE SOFTWARE YOU BOUGHT REALLY COOK! *BUY ANY 3: TAKE AN EXTRA 10% DISCOUNT!*

ILLUSTRATED TURBO PASCAL	$13.50
ILLUSTRATED AUTOCAD	$18.50
ILLUSTRATED WORDSTAR	$18.50
ILLUSTRATED d BASE II	$13.95
ILLUSTRATED d BASE III	$18.50
ILLUSTRATED LOTUS 1-2-3	$15.50
ILLUSTRATED SYMPHONY	$17.50
ILLUSTRATED C PROGRAMMING	$17.95
ILLUSTRATED MS/PC DOS	$12.95
ILLUSTRATED MULTIMATE	$17.50
WORDSTAR CUSTOMIZING	$18.50
COMMUNICATIONS HANDBOOK	$12.50
ADVANCED TURBO PASCAL PROGRAMMING TECHNIQUES	$19.95

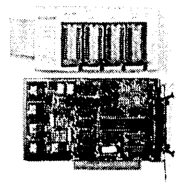
NEW!

S100 BUSS LIVES! WE STILL SELL S100 PRODUCTS AND CP/M SINGLE BOARDS. FREE CATALOG.

SCHEMA: Schematic Capture and Drawing Package

WORKS WITH PC'S AND COMPATIBLES. EASY TO USE WITH OFF THE SHELF PRINTERS, MOUSES, VIDEO CARDS, ETC. COMES COMPLETE WITH A HUGE COMPONENT DATABASE. SUPPORTS MONOCHROME, COLOR, OR EVEN EGA. FREE BROCHURE AND DEMO DISKETTE ON REQUEST. LIST $500 – **SPECIAL $396.15**

NEW! PC/XT EPROM PROGRAMMER $199

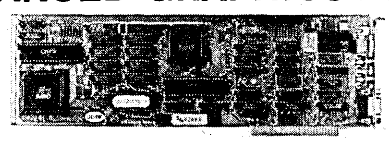


★ LATEST DESIGN ★ PROGRAMS UP TO 4 DEVICES AT ONE TIME ★ FEATURES EASY TO USE MENU DRIVEN SOFTWARE THAT RUNS UNDER PC OR MS-DOS. ★ USES AN INTELLIGENT PROGRAMMING ALGORITHM FOR SUPER FAST (8X) EPROM BURNING. ★ THIS PLUG-IN BOARD ATTACHES TO AN EXTERNAL MINI CHASSIS CONTAINING 4 TEXTOOL Z.I.F. SOCKETS. ★ 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. $199 WITH SOFTWARE.

MICROTEK 2 M.B. EXPANDED MEMORY BOARD $195 (O. K)

MEETS LOTUS/INTEL/MICROSOFT SPEC. FOR EXPANDED MEMORY. FOR USE AS EXPANDED MEMORY (EMS), ELECTRONIC DISK, OR PRINT SPOOLER. WITH A SUPERB MANUAL AND INSTALLATION SOFTWARE. FOR PC/XT OR MOST COMPATIBLES. FOR 2 MEGABYTES OF 150 NS RAM (72 DEVICES) INSTALLED AND BURNED IN ADD $225.

PC/XT AT ENHANCED GRAPHICS ADAPTER



$249

- ★ 3 CARDS IN ONE!
- ★ 100% IBM COMPATIBLE.
- ★ COMPATIBLE WITH IBM EGA, COLOR GRAPHICS ADAPTER, OR MONOCHROME ADAPTER.
- ★ 256K VIDEO RAM! (4 TIMES MORE THAN IBM!)
- ★ DUAL FREQUENCY OUTPUT FOR EITHER EGA OR STANDARD RGB COLOR MONITORS.
- ★ PERFECT MATE FOR NEC MULTISYNC COLOR MONITOR!
- ★ LIGHT PEN INPUT.
- ★ FULL 16 COLORS.
- ★ PARALLEL PRINTER PORT.
- ★ A SUPERIOR BOARD AT ABOUT 1/3 THE COST OF IBM EGA!
- ★ ASSEMBLED, TESTED, AND BURNED IN. WITH MANUAL.
- ★ USES CHIPS & TECH. VLSI 4 CHIP SET.
- ★ FIELD PROVEN BIOS.

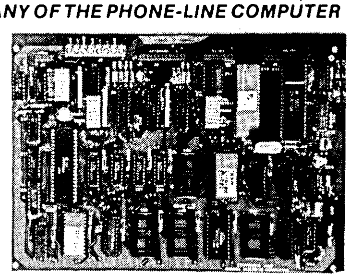
ZRT-80 CRT TERMINAL BOARD!

A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.

FEATURES:

- ★ Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
- ★ RS232 at 16 BAUD Rates from 75 to 19,200.
- ★ 24 x 80 standard format (60 Hz).
- ★ Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
- ★ Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
- ★ Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
- ★ 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
- ★ Composite or Split Video.
- ★ Any polarity of video or sync.
- ★ Inverse Video Capability.
- ★ Small Size: 6.5 x 9 inches.
- ★ Upper & lower case with descenders.
- ★ 7 x 9 Character Matrix.
- ★ Requires Par. ASCII keyboard.

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD $10



$89.95 #ZRT-80 — A&T ADD $50

(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716 CHAR. ROM. 2732 MON. ROM
$49.95
SOURCE DISKETTE - ADD $10
SET OF 2 CRYSTALS - ADD $7.50

THE NEW 65/9028 VT ANSI VIDEO TERMINAL BOARD!

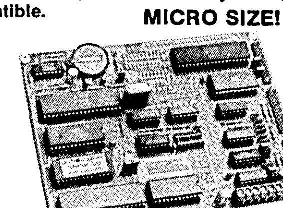
★ FROM LINGER ENTERPRISES ★

A second generation, low cost, high performance, mini sized, single board for making your own RS232 Video Terminal. This highly versatile board can be used as a stand alone video terminal, or without a keyboard, as a video console. VT100, VT52 Compatible.

MICRO SIZE!

FEATURES:

- ★ Uses the new CRT9128 Video Controller driven by a 6502A CPU
- ★ On-Screen Non-Volatile Configuration
- ★ 10 Terminal Modes: ANSI, H19, ADM-5, WYSE 50, TVI-920, KT-7, HAZ-1500, ADDS 60, QUME-101, and Datapoint 8200
- ★ Supports IBM PC/XT, and Parallel ASCII Keyboards
- ★ Supports standard 15.75 kHz (Horiz.)
- ★ Composite or Split Video (50/60 Hz)
- ★ 25 X 80 Format with Non-Scrolling User Row
- ★ Jump or Smooth Scroll
- ★ RS-232 at 16 Baud Rates from 50 to 19,200
- ★ On Board Printer Port
- ★ Wide and Thin Line Graphics
- ★ Normal and Reverse Attributes
- ★ Cumulative Character Attributes: De-Inten, Reverse, Underline and Blank
- ★ 10 Programmable Function Keys and Answerback message
- ★ 5 X 8 Character Matrix or 7 X 9 for IBM Monitors
- ★ Mini Size: 6.5 X 5 inches
- ★ Low Power: 5VDC @ .7A, ± 12VDC @ 20mA.



$99.95

ADD $40 FOR A&T

(Full Kit w/100 Page Manual)

SOURCE DISKETTE: PC/XT FORMAT 5¼ IN. $15

NEW OPTION

PC/XT keyboard ROM. Allows use of IBM style keyboard. $15

Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

TERMS: Add $3.00 postage. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCharge. Tex. Res. add 5-1/8% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance. Prices subject to change without notice.

*TM OF DIGITAL RESEARCH INC. (CALIF.) *IBM TM OF INT. BUS. MACHINES. WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

Micro C Contest Results

By Larry Fogg
Staff

Solid Code From Solid C'ers

It's contest judging time again. Around the holidays, the Micro C office takes on a video game arcade atmosphere, with beeps and whistles in the air and crowds of people surrounding shy computers. Join us for a look at the winners.

This year the focus turns to C. We had 20 entries, considerably fewer than the 100+ for the Pascal Runoff last year. When you think about it, this makes a lot of sense. Pascal is an "easy" language. You can be away from it for a while and then sit down and knock out a useful program without major relearning.

On the other hand, it takes a constant and (as one unnamed individual might say) C-rious effort to stay on top of C. As a result, although we had fewer entries, they were generally very high quality.

It's also interesting that half of the winners run on both MS-DOS and CP/M systems. Hooray for portability.

And The Winner Is ...

I think the envelope is in here somewhere.

The Grand Prize of a Definicon Systems DSI-32 board, an optimizing C compiler from Manx, Essential Software's C library, and a three year subscription to Micro C goes to:

Kirk Bailey
P.O. Box 1702
Corvallis, OR 97339

Kirk's entry form said his goal was to "win a subscription to your fine magazine (flattery intended)." We're so flattered that we're giving him the whole ball of wax for MORSE, a Morse code training program. In a world where C programming is maligned for its obscurity, MORSE stands out as an excellent example of clear, functional code.

Several different modes are available. Input from the keyboard or from a text file can be transmitted from the speaker. Or the program can test your keyboard responses to random characters output on the speaker.

Congratulations, Kirk, on a beautiful piece of work.

First Prize

The First and Second place finishers receive the optimizing C compiler, the C library, and a two year subscription to Micro C.

Bill Mahoney
7610 Redick Ave.
Omaha, NE 68112

Bill's game of Backgammon takes First Prize. It runs on both MS-DOS systems and the Kaypro, and plays a very intelligent game. It even beat Dave (who thinks he's an intelligent player.)

Second Prize

Harvey Miller
428 Yosemite Ct
Petaluma, CA 94952

Clear out all those unwanted duplicate files that are clogging up your hard disk with Tree Surgeon. Harvey's entry shows locations of duplicates, compares them to see whether they really are the same, and allows deletion or renaming of individual files. Very useful utility for anyone with a winchester.

Third Prize

Third through Fifth place finishers get the Aztec C Prime Package from Manx and a one year subscription to Micro C. Third Prize belongs to:

Gary Oliver
P.O. Box 826
Corvallis, OR 97339

Gary's MS-DOS / CP/M C Preprocessor expands macros, does conditional compilation, and checks include files. An excellent addition to Small C and other compilers which do not support macros.

Fourth Prize

Dave Regan
P.O. Box 601
Corvallis, OR 97339

Word Search is yet another entry from the Corvallis connection. (Is there something going on over there that we don't know about?) A very generic program (MS-DOS, CP/M, and UN*X), Word Search generates those hidden word puzzles you see in the Sunday paper. Feed it a bunch of words and the program will create an array of characters with your words hidden inside.

Fifth Prize

Noriaki Hosoya
1600 W. Plum 23-D
Fort Collins, CO 80521

Noriaki's LOG brings time and date stamping of files to Kaypros equipped with real time clocks. This is a useful program with really nice looking code.

Very Honorable Mention

No C contest would be complete without an entry from Sigi Kluger. Sigi wrote a communications package for the DSI-020 called COMM20. Written in his usual fine style, we nonetheless had to exclude it from the prizes due to DSI's connection with the contest.

Till Next Year

Our thanks to all the folks who put so much into their entries. We had a great time judging them. (Disks will be announced next issue.)

PROLOG will be the language for next year's contest, so put on your AI hats and start writing.

■ ■ ■

C Contest Entries

Kirk Bailey	MORSE	Morse code trainer
Robert Barcus	GOLFCAP	Handicap system for 18 hole golfers
Steve Bloom	JW	HP Laser Jet printer utility for CP/M
Graham Collins	DO	Automates compilation of CP/M C programs
David Fox	NAG	Periodic event reminder
Adam Fritz	HB	Banner printer
A. W. Gustafsson	DSKED	MS-DOS disk editor
Cameron Hall	PICK	List manager
Noriaki Hosoya	LOG	File time/date stamper for CP/M
Sigi Kluger	COMM20	DSI-020 communications software
John Lucas	BAC	CP/M backup utility
Bill Mahoney	BKG	Backgammon
Harvey Miller	TREESURG	Cleans up duplicate files on MS-DOS hard disks
Gary Oliver	PP	C preprocessor
Timothy Prince	DRVLINR	Floppy disk drive alignment
Dave Regan	WORDSRCH	Generates word searching puzzles
J. B. Robertson	CPRN	Smart file printer for C source code
William Robertson	INTERMOD	Calculates transciever intermodulation products
Michael Rovak	DEL	CP/M smart file erase
Austin Taube	DSWP	CP/M directory maintenance utility

Solution to last issue's "Trademarks" puzzle

B	O	A	R	D		C	O	D	E		A	L	T	
A	D	L	I	B		O	P	E	N		M	O	O	G
D	O	G	M	A		M	U	L	T	I	M	A	T	E
E	R	A		S	E	A	S			M	O	D	E	M
			W	E	A	L		D	E	S				
A	L	T	A	I	R		S	E	T	A		R	I	B
C	O	A	T	I		A	T	A	R	I		A	L	L
O	R	B	S		D	I	A	N	E		T	R	I	O
R	E	L		P	O	R	T	S		W	R	E	A	K
N	N	E		O	N	E	S		S	O	U	R	C	E
				L	E	D		M	A	Z	E			
A	C	C	R	A			A	D	E	N		L	I	I
S	U	P	E	R	C	A	L	C		I	C	O	N	S
P	L	/	I		R	I	S	C		A	L	G	O	L
	L	M	N		C	R	O	C		K	R	O	N	E

Technical Letters We Didn't
Get Around To Running:

To:
The Technical Head
Micro C

Dear Head,

I was more or less impressed with Micro Cornucopia, and your speedup articles on the Kaypro and the PC clone sounded so exciting I bought one of each. They did OK at the store.

In fact, the Kaypro was working fine, mostly, before I pried the lid off. Didn't notice the screws in time. Finding the right place to do something was even more interesting. Anyway, after I cut the traces, it hasn't been the same.

Emily found the right chips (she can see those teeny little numbers) and I worked very carefully. I counted pins around one way and cut traces. Then, just to be sure, I counted around the other way and cut traces. You can't say I didn't get the little buggers. Then I counted both ways and did the jumpers.

IBMs are great to work on — they have more parts (no doubt in case some don't work). I did the crystal change in issue #28. I fired it up and though it didn't light, at least it didn't go "beep, beep." Then I did the RAM resident speed up. Fortunately the "beep, beep" didn't return.

You might have guessed that I'm not real experienced around computers, but I used to be a plumber so soldering is no sweat.

In fact, your instructions have been OK except for one thing: you screwed up in your choice of soldering irons. (You didn't say how you used those wimpy little things.) I can't do anything with them. I had just started heating up two of them with my torch when they both sagged and made a blob on the table.

Anyway, enough of this blathering. The reason I'm writing is to get your suggestion on another computer that I can speed up. Should I try again on a Kaypro or is there something easier for starters?

Horace P. Willingham III
Lompac, TX

Dearest Horace,

Don't speed-up another Kaypro. Please! There might not be enough old 2's around for you to get one running.

There are lots and lots of Apples out there, and they could really use speeding up. However, we think you should be careful, Apples have plastic cases. We noticed from the picture you sent that your Kaypro's cabinet is sagging.

■ ■ ■

By Don Thompson
MicroSphere
PO Box 1221
Bend, OR 97709

Running A Computer Store : Selling Hardware & Service

Starting and running a successful computer store is not as easy as purchasing a ComputerLand franchise and watching money roll in the door. A quick look at the Wall Street Journal auction section demonstrates clearly that success in a computer store (including ComputerLand) requires much more than a good selection of computers. This is the story of a successful new computer store.

I remember a few years ago (just before we moved to Bend) when Roth's groceries moved into Albany. Albany was already overpopulated with grocery stores. Roth's prices were equal or higher on most items, and the store location was convenient but not on the main drag. Within two years they were the most popular store in town. Even the large chain stores like Safeway or Fred Meyer's couldn't compete with Roth's Friendly Foodliner.

Examples Of Service

Here are some examples of the type of service provided by Roth's. The manager would carry out groceries, clean up broken bottles, or run a cash register, whatever needed to be done. The store was spotless and the lines were usually short.

He greeted regulars by name. He cashed $50.00 checks on Friday night without a purchase. He expected employees to work as hard as he did. Since he was working along side of everyone else, he knew which people were keeping up. The wages were the best in town.

Customers really respond to good service. The other stores just couldn't compete with that kind of service.

Computer Business

The computer business is very similar to the grocery business. It is very competitive, and the easy money selling sealed boxes to yuppies for $5000.00 a shot is gone. Now customers demand service.

There are never enough people willing to put forth the time and energy to provide excellent service. Almost any town of reasonable size is ripe for a competent new computer store.

In fact, people now drive 200 miles just to do business with us. That's a pretty good indication that we're doing something right.

Now I'll get off my soap box and get down to practical suggestions.

Computers Are A Unique Business

New customers rarely know what they need. If you're lucky, they know what they want to do.

It's easy to take advantage of people who are buying their first system. Easier, in fact, than taking the time to really understand their needs. It's also easy to throw a bunch of expensive software at the problem.

Until they use their computer and talk to friends who also have one, they won't know the difference. However, if you sell them the wrong system or overcharge them, they'll eventually find out and business will vanish like a scared rabbit.

If you treat customers with respect and work hard educating them so they can make informed choices the first time, their friends will hear about it. Those friends will travel great distances to walk in the door and say:

"Sell me a system like the one you sold Joe. He likes it, and it works."

Another advantage of an educated customer is that the decision on what to purchase is a shared decision. The customer has some responsibility as well as the dealer. If it turns out that the choice wasn't perfect, the two can work together on a solution.

For instance, showing two printers in the customer's price range and letting him chose is much better than showing him one and insisting that it's the best one for him. Computers and monitors fall in the same category.

Salesmanship

You can't use a used car salesman's hard sell. You've got to have an honest desire to do the best job for your customer. (The customer will pick up on it right away.)

It's also important to know when to ask for the order and when to let the customer think things over. For example, quit talking when the customer digs out his checkbook. It is just as easy to talk yourself out of a sale as into one.

Unfortunately, good salesmanship is learned in the school of hard knocks. Try it before you bet $50,000.00 that you can do it.

Be open to your customer. Why is he buying? On what is he basing his decision? Make no assumptions. Ask questions until you really understand. Not all people are shopping just for price.

A corporate buyer might really appreciate a 2400 baud modem or an enhanced multi-video card. They raise the price of the system, but they also make the unit more valuable for his use.

Understand Computers

Learn everything you can about computers, both the hardware and software. It really helps if they fascinate you.

If you don't really enjoy computers, then you'll find it more difficult to sell them and make them work. When computer salesmen give out inaccurate information it usually means that they don't care.

It does not inspire confidence when a salesman takes 30 minutes to figure out that an Apple II+ disk will not boot an IBM. Customers have told me that they are glad to finally find someone who can make a computer do what the customer knows it should be able to do.

Be familiar with the internal operation of the computer. Be able to explain why hooking a serial printer to a parallel output won't work even though the connectors look the same.

Reserves

Have enough money in the bank so the success or failure of one sale does not affect whether the business will open the next week. Customers can immediately detect when the salesman is desperate for money rather than interested in their needs.

Making The System Work

You need a knack for making things work right. It is truly amazing to see computer systems that other dealers have put together. It's immediately obvious that they'll never work.

Make Friends

Being able to communicate seems to be a lost art in America. It takes time to put yourself in the customer's shoes. If you don't take the time to let him like you and trust you, things will be much more difficult when the customer is in the midst of figuring out his new system.

Who Your Customers Really Are

Spend your time with the people who can make the decision to buy and with the hackers who have figured out new and easier ways to format hard drives or use applications packages. There are other people who have unlimited time, but nothing to contribute. They'll be glad to take your time if you let them.

Choose Your Brands Carefully

Does the equipment work reliably and can it be repaired easily? If it takes three months to get a bad part replaced, I guarantee there will be a few unhappy customers in your store every day waiting to hear about their systems. Not all companies provide good support or quality.

Never totally believe any distributor trying to sell computer equipment. Try it before you buy it. If you wouldn't use it, don't sell it. Think of all the computer stores that sold IBM PC Juniors. Their unfortunate customers bought their second computer (first real computer) somewhere else.

Can you make a little money selling it? If you have to compete with mail order companies on your entire product line, things get much more difficult. For example, Kaypro and Leading Edge will discontinue a dealer who sells systems mail order. Thus, price and profit margins can be maintained. If you can't make money selling a system, do not sell it. Put a price on your time and energy. There are lots of customers out there who understand its value.

Will customers accept the brands of equipment you sell? Are they worthy of the investment you're asking them to make? Do you have magazine reviews and product documentation to support your recommendations?

One of the easiest sales is to a large company, especially when I hand them 20 pages of reviews and information on the products I'm recommending. It's impossible for any single customer to try every item on the market. And it's hard to argue when Byte, PC World, PC Magazine, and Consumer Reports all like a system.

Don't Forget Price

Since I'm not an authorized IBM dealer, reviews are critical if I want to win over sales that were scheduled to go to IBM. There's an old saying, "No one ever got fired for buying Big Blue." But I can compete on PRICE.

After I sold the local hospital a system and a laser printer for the price they'd budgeted for an IBM alone, everyone went away happy. The system was for the purchasing department. They gave the laser printer to their boss.

Finally

I guess the most interesting thing, and most basic business requirement, is figuring out how to sell the right equipment at a fair price with a reasonable profit — and then making it do what the customer needs. Figure that out and, bingo! — you'll succeed.

■ ■ ■

PC SCHEME

Reviewed By Steve Witkowski
23265 Cindy Ct.
Bend, OR 97701

A Great Way To LISP On The PC

LISP is back out of the closet. It's not new. Years ago it overwhelmed 8080s and 6800s with its memory demands, and overwhelmed programmers with its parentheses. Now, because of the interest in AI and Prolog, Steve takes a look at a new Scheme of this ancient language.

The resurgence of LISP as a language for creating expert systems, and the arrival of fast, powerful PCs with enlarged memory, have led to the introduction of several new, very complete, LISP interpreters.

One of these LISP dialects, PC Scheme (version 2.0), can access memory above the DOS 640K barrier, and is an excellent package for learning LISP as well as for creating professional applications.

A Little Background

PC Scheme is a close relative of the Common LISP standard, and anyone familiar with any standard LISP dialect will have little trouble using it. If you're new to LISP, PC Scheme is a great way to learn the language.

Scheme was developed at the MIT Artificial Intelligence Laboratory for educational environments. It's easier to learn than Common Lisp, and in many ways offers greater flexibility, since it isn't bound to the rigid confines of a standard. (Incidentally, if you venture into the LISP world, don't be surprised to find more than a little anarchy.)

One problem with all of the PC LISP dialects is speed. And although PC Scheme is an incrementally compiled language, don't expect the lightning performance of Turbo Pascal or Logitech Modula-2/86. However, it's adequately fast on my computer — an AT&T 6300 (NEC V30 CPU).

LISP is memory hungry, so I've added a HiPage extended memory board and a Maynard Accent expanded memory board, for a total of 2.64M.

PC Scheme: What You Get

Texas Instruments (TI) has provided three different versions of PC Scheme 2.0. One for expanded memory systems, one for extended memory systems, and one for systems with up to 640K.

TI claims that the expanded and extended memory versions can access "close to" 2M of RAM. I've been able to use 1.87M, which is close enough.

SCHEME is available from:

Texas Instruments Inc.
Data Systems Group
SCHEME Product Center
P.O. Box 2909 M/S 2244
Austin, TX 78769-2909

The price is $95, and includes a User's Guide and a Language Reference Manual. Both volumes are complete, but if you're brand new to LISP, you'll need some additional help.

Structure And Interpretation Of Computer Programs by Abelson and Sussman (from MIT Press for $35) is an excellent tutorial. It's one of those rare computer books that's a joy to read. And coupled with PC Scheme, it provides an ideal LISP learning and programming environment.

You need at least 320K to run PC Scheme, and at least 512K if you want to use the EDWIN text editor that's part of the package.

I recommend a minimum of 640K for serious programming, and it doesn't hurt to add an additional 2M of expanded or extended memory if you want to see the full power of the language.

Scheme & Common LISP

Scheme and Common LISP are very similar. Consider the following code to define a procedure SQUARE. In Scheme —

```
(DEFINE (SQUARE X) (* X X))
```

And in Common LISP —

```
(DEFUN SQUARE (X) (* X X))
```

The major difference in syntax is the DEFINE in Scheme versus DEFUN in Common LISP, and the placement of parentheses around variables. Those are about the most important differences between the two dialects.

Editor, Graphics, Procedures

Scheme comes with an EMACS-like editor, EDWIN. With this powerful editor you can split the screen between Scheme and EDWIN.

A blinking left parenthesis tells you where you are with respect to the proper closure of parentheses. Control keys move you back and forth between the two environments, and let you save and retrieve files. The text editor is more than adequate for most programming.

If you aren't familiar with an EMACS editor, you may find EDWIN a bit confusing. I did. Once you learn how to use it, however, you'll appreciate EDWIN's power and flexibility. (Or you can choose any editor that generates ASCII text.)

PC Scheme includes a variety of color and graphics processors: SET-PALLETTE!, SET-PEN-COLOR!, and MAKE-WINDOW.

PC Scheme has transcendentals, DRAW procedures for drawing lines and boxes, a DOS-CALL to execute .COM or .EXE files, as well as class, vector, and string operators.

The Power Of Any LISP

LISP's power comes from its object-oriented nature. LISP treats all objects as data — numbers, symbols, other procedures. All are arguments which can be operated on to create new objects.

This ability to treat objects as data is where Scheme excels. Its object-oriented system, SCOOPS, creates a first-class programming environment with multiple and dynamic inheritance.

The variables define the properties of objects, and the methods define their behavior. In this object-oriented programming world, different levels of abstraction define different classes, and inheritance defines how properties are transferred to other classes.

I know this may sound a little strange, but it's easy to get the hang of. For example, let's define a new procedure, CUBE —

```
(DEFINE (CUBE X) (* X X X))
```

Or, in terms of the previously defined procedure SQUARE —

```
(DEFINE (CUBE X) (* X (SQUARE X)))
```

which illustrates LISP modularity. Once you define a procedure, you can use it to build other procedures. In the global Scheme environment, your library of procedures floats around, ready to be used as data or to help you build new procedures. A procedure SUM-SQUARE-CUBE could be defined as in Figure 1.

Then you could obtain the cube of the result by finding —

```
(CUBE (SUM-SQUARE-CUBE X Y))
```

No Compile To .EXE

PC Scheme is an incrementally compiled language, so it behaves like an interpreter. Scheme compiles to a low level code, which speeds up execution, but requires PC Scheme's presence for execution. This is easily Scheme's biggest weakness. If Scheme could generate .EXE files, it would be a super LISP for the PC.

SCHEME.INI loads procedures for you when you first enter Scheme, eliminating the need to tediously load each file. You can also use macro definitions and a LOAD FORMAT to speed things up.

Wrap Up

The package is about as bug-free as any software I've used. I had some initial problems with the compatability of version 1.0 and the AT&T 6300 graphics board, but these have been resolved in version 2.0. Also, TI's technical support has been outstanding — down to returning my calls.

I've found no problems with Version 2.0 except for the two shortcomings I mentioned. The documentation isn't adequate for a novice, and the package can't generate an .EXE file.

Since you'll probably have to pay $1000 or more for a PC version of LISP that creates stand alone code and uses more than 640K, PC Scheme is a bargain.

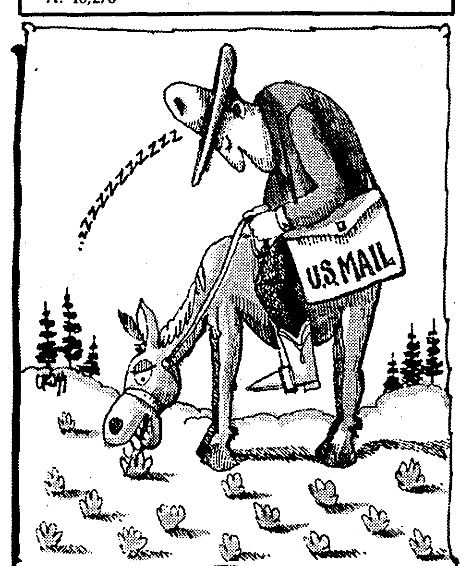
Editor's note: I'm not sure how actively TI is pushing PC Scheme. I contacted TI's Oregon office to find out about ordering it. They hadn't heard of it and didn't know who to contact. When I asked if someone could find out and let me know, the answer was 'no.'

■ ■ ■

Figure 1 - Procedure SUM - SQUARE - CUBE Defined

```
(DEFINE (SUM-SQUARE-CUBE X Y) (+ (SQUARE X) (CUBE Y)))
```

U.S. Postal Service Statement of ownership, management and circulation (Required by 39 U.S.C. 3685) 1A. Title of Publication: MICRO CORNUCOPIA 1B.Publication Number: 0747-587X 2. Date of Filing: 9-30-86 3. Frequency of Issue: Bi-monthly 3A. Number of Issues Published Annually: 6 3B. Annual Subscription Price: $16.00 4.Location of Known Office: 155 NW Hawthorne, Bend, Oregon 97701-2917 5. Location of the Headquarters or General Business Offices of the Publishers: 155 NW Hawthorne, Bend, Oregon 97701-2917 6:Name and Complete Address of the Publisher, Editor, and Managing Editor: Publisher: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917; Editor: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917; Managing Editor: David J. Thompson 155 NW Hawthorne, Bend, Oregon 97701-2917 7. Owner: Micro Cornucopia, Inc. PO Box 223, Bend, Oregon 97709-0223; David J. Thompson 1259 NW Iowa, Bend, Oregon 97701-1001; Sandra S. Thompson 1259 NW Iowa, Bend, Oregon 97701-1001 8. Known Bondholders, Mortgagees, and Other Security Holders Owning or Holding 1% or More of Total Amount of Bonds, Mortgages or Other Securities: None 10. Extent and Nature of Circulation: Average Number of Copies Each Issue During Preceding 12 Months A. Total Number of Copies Printed: 16,607 B. Paid Circulation 1)Sales Through Dealers and Carriers, Street Vendors, and Counter Sales: 4,152 2)Mail Subscription: 9,020 C. Total Paid Circulation: 13,172 D. Free Distribution by Mail, Carrier, or Other Means, Samples, Complimentary, and Other Free Copies: 1,816 E. Total Distribution: 14,988 F. Copies not Distributed: 1)Office Use, Left Over, Unaccounted, Spoiled after Printing: 1,352 2)Returns from News Agents: 267 G. Total (Sum of E, F1 and 2)-Should Equal Net Press Shown in A: 16,607; Actual Number of Copies of Single Issue Published Nearest to Filing Date: A. Total Number of Copies Printed: 16,276 B. Paid Circulation 1)Sales Through Dealers and Carriers, Street Vendors, and Counter Sales: 5,257 2)Mail Subscription: 9,011 C. Total Paid Circulation: 14,268 D. Free Distribution by Mail, Carrier, or Other Means, Samples, Complimentary, and Other Free Copies: 280 E. Total Distribution: 14,548 F. Copies not Distributed: 1)Office Use, Left Over, Unaccounted, Spoiled after Printing: 1,650 2)Returns from News Agents: 78 G. Total (Sum of E, F1 and 2)-Should Equal Net Press Run Shown in A: 16,276



A Quiet Morning At Micro C

Tech Calls Mirror Tech Needs

By Larry Fogg
Micro C Technical Person
And Keeper Of The Towel

You really have to be here for a while to understand what it's like to spend a morning helping people with their technical problems. (In fact, if you want to volunteer some time...)

Many mornings are fun. The problems are interesting, the solutions straightforward. It's puzzle solving at its best.

This is a look at one morning.

The Day Begins

"Hi, this is Larry."

Actually, this statement is not altogether true. Most folks can tie their shoelaces and speak in complete sentences by 9:00 am. This morning (like most others) these complex tasks elude me.

What we really have here is part of Larry. While this part tries to answer the first tech call of the day, the rest of me remains blissfully oblivious to the morning. After a couple of tech calls I'll get it together and be ready for the day.

This morning's opening call awakens me prematurely.

Shock Treatment

"Hi. I'm calling from Martha's Vineyard. You probably read about the storm we just went through. (It had been national news.) Seems my Kaypro got blasted by a lightning strike. I see lots of charcoal when I pop the top. Any suggestions?"

"Punt," I reply in my best high techese. "Call ERAC in San Diego and pick up a surplus main board. It doesn't make sense to get into an extended battle with a barbecued board."

Miami Advice

"Tech on three."

I punch line three and pick up the phone. Miami calling. The mercury's way below freezing in Bend and I just know the caller's basking under a mid lunch hour sun.

"How's the weather there?" gloats Miami.

"Fine. How's the skiing in Miami?"

Pleasantries aside, we get down to business. "My drives have gone south (To Cuba?) and I need to replace them. What do you recommend?"

"We still like Mitsubishis and Teacs. Never had one fail and we've sent plenty of drives (mostly Tandons) to the great warehouse in the sky.

"But before you give up on them, try a little routine maintenance. It's not uncommon for older drives to fade away due to a sticky head transport mechanism. Sometimes a drop of light oil (like TriFlo) on each rail will loosen things up and bring the drive back to life."

More Drive Problems

"I was futzing around with my PC over the weekend (weekends are dangerous if you're a computer) and accidentally fired it up with the drive cable on backwards. Now I can't boot the system. Have I lost the drive or can it be fixed?"

A lot of questions have obvious answers. Obvious after the caller hangs up, that is. My reasoning on the phone should have been:

The odd numbered lines on one side of the drive connector are all grounds. The signals live on the connector's other side in the even numbered lines. With the connector on backwards, every signal line is connected to ground.

Since the signals are active low, all drives are selected, the motors spin up, the heads home and are loaded, and voila! All of a sudden we're writing garbage to track zero. Garbage is certainly not the breakfast of champions for a floppy disk.

A new system disk would have fixed this gentleman's "dead" drive. I hope he's reading these words.

MAX Attack

"Help! I stuffed your Pro-884 MAX ROM into my Kaypro and now it has terminal indigestion. No life at all."

"Sometimes ROMs get zapped in transit but this sounds more like a problem with the main board. Kaypro's ROMs are 24-pin 2732s. However, Kaypro provided for the use of a 2764 ROM by putting in a 28-pin socket. Our ROMs are 2764s.

"Unfortunately, on some boards the extra pins aren't wired correctly. The problem doesn't show up until a larger ROM or EPROM, like ours, is installed.

"EPROMs need to know whether they're being programmed or used. A high (+5 V) signal on pin 27 tells the EPROM that some work is expected of it. A low (0 V) signal means it's going to be programmed. If +5 V isn't being supplied to the EPROM, it will just sit there. Most computers look pretty unimpressive when their monitor ROMs are on a coffee break."

It's very satisfying when a problem gets resolved while I'm still on the phone. This caller has his machine and a voltmeter handy. A quick check of pins 27 and 28 shows low voltage on 27, and a jumper to +5 V gives us a live Kaypro.

Power Supply Connections

Micro C is hardly your typical magazine. Our readers (yes, you) have always made major contributions to our common store of knowledge. When a particularly nasty problem gets solved, you let us know.

I've heard the following symptoms several times. But until a reader sent in a tip on the subject I was totally in the dark.

"It happens for no reason. I'll just be sitting there reading the screen when both drive select lights come on, the motors spin up and the keyboard heads for Never-Never Land. Is my Kaypro possessed?"

Next time I'll be able to answer, "No gremlins involved. Take a look at the solder joints where the power supply cable attaches to the power supply board. Probably one or more of them will have worked itself loose. Resolder all the wires and the problem should disappear."

Science Break

I don't take tech calls from 11:00 to 11:01. That's when Margret and I challenge our minds by tuning in *Dr. Science* on the local Public Radio station. This morning's question has to do with ground water.

"What is it and how do you grind water anyway?"

I'll let you mull that one over while we get on to the last hour of tech calls.

Bad Timing

A very worried voice on the far end of the line says, "I did your 5 MHz speedup on my old Kaypro, and now all I get is flashing garbage on the screen. What happened?"

The Flashing Garbage Syndrome points strongly towards a bad system clock. At this point a picture (on an oscilloscope) is worth a thousand words. We're looking for a nice 5 MHz square wave on pin 6 of the Z80. Instead, the Z80 probably sees a wimpy, broken down ripple.

But this caller has no scope, so all I can say is, "Check your connections carefully. Look for cold solder joints (very dull looking) and be sure that the output pins of U86 (4 and 5) are pulled out of the socket. If the pins remain in the socket, the Z80 sees both timing signals at once and gets confused. A processor can run at 5 MHz or 2.5 MHz but not both at the same time.

"If everything looks good and the Kaypro runs at 2.5 MHz but not at 5 MHz, be suspicious of 5 MHz. There is a 4 MHz signal (U87 pin 6) which will be cleaner. Try using it instead. The Kaypro does use 4 MHz elsewhere so be sure to leave pin 6 in the socket."

Bad Media During Format

"I keep getting 'disk unusable' messages when I try to format a floppy on my PC."

"You probably have a fast clone of some sort. A lot of sped-up systems don't like to format at the high speed. The drive head doesn't have enough time to settle over a track before it tries to write. Slow down to 4.77 MHz and try again."

Last Call

It's 11:59. The old stomach is sounding like Mt. St. Helens. Think I'll slip away and ...

"Tech on one."

"Uniformed agents of the government just delivered a couple of your MS-DOS public domain games disks. Most of them work well, but several are behaving rather strangely. They clear the screen and perhaps play some music, but the screen stays blank. I have to reboot the system to regain control. Could this be a plot to take over my computer?"

"Probably. The other possibility is that you are trying to run programs designed for a color system on a monochrome machine. Color and monochrome data are stored in different areas of memory. So if a program writes to color memory but the computer reads from monochrome memory (or vice versa) nothing ever makes it to the screen. The computer still lives, but the only way it can talk to you is through the speaker."

Fini

So ends another Micro C morning on the phones. It's great fun, really. I get to talk to folks from all over the world and, occasionally, we even solve some problems.

■ ■ ■

Slowing Your Fan

A Quiet Introduction To Mr. Ohm

By David Thompson
Manager
Micro C Hot Air Dept.

If you've been no fan of Ohm's Law. If volts and amps aren't current topics at your dinner table. Then resist no longer. The following should put you and your system back on whispering terms, and make you smarter to boot.

I think best when it's quiet. I write best at a computer. So I really appreciate a quiet computer.

The major distraction in a non-winnie system is the power supply fan, and that fan is very, very buzzy.

I've found some fans that are pretty quiet. They usually draw about half the power of their noisier brothers. (Sometimes they're marked "half speed.") However, they're usually hard to find (being so quiet and all) and can be fairly expensive ($10+).

So, the following is how you can quiet your noisy fan without causing a family revolt. (And besides, you get to meet Mr. Ohm.)

How To Quiet Your Own

First, let's dig the power supply out of your clone. It contains the fan.

Disconnect the power cord from the system, open the cabinet, and remove the four screws from the back of the cabinet that hold the power supply in place.

Before disconnecting all the power leads, note how the main power connector (it may be split in half) plugs into the processor board. The ground wires (black) lie in the center of the whole mess. (The connector has a notch on one side which snaps into a lip on the socket, but it can be plugged in backwards if it's forced.)

Disk drive power connectors are pretty foolproof; they have little polarizing corners that do the job.

Inside The Supply

Open up the power supply (a half-dozen or so screws). The fan is mounted to the top of the cabinet with two leads going down to the main supply board. They supply 12 VDC to the fan. The red lead is plus.

Look closely at the center of the fan motor. It'll be marked with either the current draw (.2 amps is common) or the wattage. The wattage will usually be between 2 and 5 watts.

Obviously, the purpose of the fan is to move air, and if your system is packed tight with full-length boards and winnies, you might want to speed it up rather than slowing it down. (But that, if it's possible, is not the subject of this article.)

Assuming your system is running comfortably cool, there shouldn't be any problem with slowing the fan just a bit. You'd be amazed how quiet most fans become after just the slightest speed reduction. The key is to reduce the speed of the blades to just below the point of irritation.

Volt, Ampere, Watt, & Ohm

Before we cut a wire, add a resistor, and solder a joint or two, let's look at a little theory. Volt, Ampere, Watt, and Ohm were electrical pioneers. (That means they got to be electrical engineers without struggling through AC theory class.)

Since these discoverers needed names for their discoveries they uh... Well, they used their own. (What else could they do?)

Volts are a measure of electrical push, how hard electrons are trying to get from plus to minus (or vice versa). Lightning is a good high voltage display. The 12 volts (12 V) that drive our fan don't push very hard.

Amperes (or amps) are a measure of electrical quantity. That's the number of electrons which are flowing past a point (or through a fan motor) per hour. Our fan motor will probably draw about 1/5 ampere (.2 A).

Watts (not George Watts) are a measure of power, the ability to move something or heat something. The fan motor will probably consume about 3 watts (3 W). 60 watt light bulbs consume about 60 watts (surprise).

There's a very simple formula that ties these three together.

Watts = Volts * Amperes

The power consumed by a device (the motor, in this case) is calculated by multiplying the voltage it's receiving by the current (in amperes) passing through it. We'll get down to specific cases a moment.

Ohms are a measure of electrical resistance, and are the fourth leg of this trio.

A Liquid Example

Let's say we have a high dam. The height of the water behind the dam determines its pressure against the dam. If we put a large outlet in the bottom of the dam, then the only thing to keep the water from immediately draining through the outlet would be some kind of impediment (resistance), such as a turbine.

Given a fixed impediment (resistance), the amount of water (current) flowing through the hole would be determined by the height (voltage) of the water. The greater the water pressure (voltage), the greater the water flow (amperage). The greater the resistance, the lower the flow.

Current Flow = Pressure / Resistance

or

Amperes = Volts / Ohms

This is Ohm's law. And you can swap this formula around any way that algebra allows, such as:

Volts = Amperes * Ohms

or

Ohms = Volts / Amperes

Meanwhile, Back At The Fan

I've found that reducing the current through the fan by 10% to 20% makes quite a difference. So, first I calculate the current flow (if it isn't marked on the fan). My fan says it consumes 2.4 watts.

Watts = Volts * Amperes

or, rearranging things a bit:

```
Amperes = Watts / Volts
.2 amps =  2.4  /  12
```

My 2.4 watt, 12 volt fan draws .2 amps.

Now I'll figure how much resistance my fan has to the flow of current (remember the dam?). It receives 12 V from the power supply, and we've found that it draws .2 amps, so:

```
Resistance = Volts / Amperes
  60 Ohms  =   12  /  .2
```

The resistance in our 2.4 watt fan is 60 ohms. When you put 12 volts across 60 ohms, .2 amps will flow.

Let's add enough resistance in series with the fan so that only .16 amps will flow through it (that's 20% less current).

Series?

A series circuit is one where there are two (or more) elements, and all the electrons which flow through one have to flow through them all. (See Figure 1.)

It's like our dam. If we were to put two generators one after another on the same pipe, so that all the water flowing out of one had to flow through the other, then the two would be in series. If one stopped the water, the other would stop also.

Let's take a little liberty with dams. Suppose you have one generator (with, say, 60 ohms resistance to water flow) and the flow is .20 gallons/sec. If you add another generator (with 15 ohms resistance to water flow) in series with the first, the total water flow will be .16 gallons/sec. See Figure 2.

Back At The Bench

We know we want a current flow of .16 amps through the fan, and we know we have 12 volts, so let's see how much resistance we should add.

```
Resistance = Volts / Amperes
  75 Ohms  =   12  /  .16
```

The fan motor is 60 ohms, so we'll have to add about 15 ohms in series with the fan to get the total up to 75 ohms.

What wattage resistor? Good question. Current flowing through a resistance gives up energy (watts) in the form of heat. Too much heat in too small a resistor destroys the resistor. We already know that:

Watts = Volts * Amps

We know the voltage across the whole circuit (12 V), but we'll have to figure out how much of that is across the 15 ohm resistor. We know the resistance (15 ohms) and the current (.16 amps). Let's figure out the voltage across the resistor:

Volts = Ohms * Amps

2.4 V = 15 Ohms * .16 Amps

Now let's figure out the wattage:

Watts = 2.4 V * .16 Amps

Watts = .384

A half watt resistor would work, but it would get pretty warm — I'd use a 1 watt model. (It shouldn't cost more than 25 cents at a parts house.) Figure 3 shows how to connect it.

Make sure you leave no bare metal, it could short to the cabinet or other components.

(continued next page)

Figure 1 - Resistor In Series With Fan

```
                    Current Flow -->

+12 V ---------- Resistor ------------ Fan ------------ -12 V
```

Figure 2 - Two Generators In Series

```
             Generator1              Generator2
-----------XXXXXXXXXX-------------XXXXXXXXXX------------------
Pipe         60 Ohms                 15 Ohms             Outlet
-----------XXXXXXXXXX-------------XXXXXXXXXX------------------
```

Figure 3 - Connecting The Resistor

```
             red wire           black wire
+12 V ---------------- fan ----------------- -12 V

Connect the resistor in series with either fan wire.

                                black wire
+12 V --- resistor --- fan ----------------- -12 V
```

(continued from page 51)

Use heat-shrink tubing (available at most electronics supply houses) or good electrical tape to cover any bare wire. Don't cover the resistor. It'll make it hot.

Reassemble the power supply and fire it up (disconnected from the system).

The fan should run, but it should be much quieter. If it's not quiet enough, you might want to increase the series resistance slightly (to 20 ohms or so). Watch the wattage. If the fan seems too slow, then try 10 ohms. (Resistor values don't have to be exact, one or two ohms either way won't make much difference.)

When you put two resistors in series, you sum their resistances.

```
Series Resistance = R1 + R2
```

Series resistors:

```
---- Resistor1 ----- Resistor2 ----
```

If you put two resistors in parallel, it's:

```
                           R1 * R2
Parallel Resistance = ---------
                           R1 + R2
```

Parallel Resistance?

OK, I brought it up, and it's a really easy concept. Two resistors in parallel are like a dam with two separate outlets. Each outlet has its own resistance to water flow. You can shut one off completely and not affect the flow through the other.

If the resistance of the two outlets is the same, then the total resistance is half. With resistors, that means that two 30 ohm resistors in parallel would act like a single 15 ohm resistor. (Give you an idea?)

This parallel circuit of two 30 ohm resistors:

```
       --- 30-ohm Resistor ---
       |                     |
 ------|                     |------
       |                     |
       --- 30-ohm Resistor ---
```

Is equivalent to the following:

```
--------- 15-ohm Resistor --------
```

Finally

Reverse the disassembly and button her up. You'll enjoy the quiet, and for years all the relatives will enjoy hearing how you custom designed a hardware mod.

■ ■ ■

By Laine Stump
Development Foundation of Turkey
Tunali Hilmi Cad. 22
Ankara Turkey

Foreign Characters, Foreign Problems

Laine takes us Englishers to task for making the support of foreign character sets an almost insurmountable (unsupportable) task. In Laine's letter he notes the following:

"By the way, you think YOU get bothered when you pull out your Kaypro laptop on a plane to Florida; you should see the commotion when I fire up my little Toshiba on the bus to Istanbul!!"

Still more casualties in the clone department. The final tally is 5 completely useless machines and 10 that only run at 4.77 MHz (out of 35). The more I test, the more that fail. I'm now recommending X16s to anyone naive enough to ask my advice.

Interesting New Stuff

The following is a list of new equipment and software that I've seen press releases for — and would have tried out by now. But I can't because it's not available down at the local vegetable market.

Hercules Graphics Card Plus

In case you haven't noticed the huge ads in BYTE the last few issues, this is a new card with the same features as the original Hercules card. Plus, it accepts downloadable character fonts. When in the "RamFont" mode (as Hercules calls it), it can hold up to 3072 characters, or 12 different fonts of a standard 256 character set.

This looks like the answer to my wishes for a fast display system for word processing and certain publishing jobs, but I'm not sure, since I've only read about it.

Compaq 386 (or any 386)

I really don't need this much power, but just think of the idolistic implications of such an omnipotent hunk of silicon! Not to mention the cranial ecstasy created by such a high system "macho index."

MS-DOS 5.0 (multiuser DOS 386)

I don't even own a machine that this would run on, and it's not available yet anyway. But I still keep falling into daydreams about the possible cost savings, simplicity of design, and flexibility (compared to a minicomputer or a network) if you stuck it on a 386 machine with several terminals.

Microport Unix System V

See above. And the runtime system is only $150!!

The X?? From PC Tech

Come on guys! I know you're working on something up there in your little secret valley, and I have a feeling it isn't a sequel to "Lake Wobegon Days."

Microsoft Codeview Or DSD86

I don't care which (I'd love to be given a choice!). I use SYMDEB (big brother of DEBUG) more than any other program in my arsenal, with the exception of my text editor (EXPRESS, of course). The thought of a screen oriented debugger with a dynamic stack display, register display, and code display, as well as breakpoints on memory references and other such goodies, really sets my head spinning. Too bad I can't justify the cost of MS C (Codeview is included).

Anything With Four Wheels

As long as it has bucket seats and 1500 CCs — built in Britain: I don't spend all my time thinking about computers, you know!

International Software

Since coming to Turkey, I've been asked by several acquaintances to recommend a software package for one thing or another (the most common seems to be word processing). Even before considering cost, I've found that I must immediately cross several candidates off my list of possibles just because they can't function properly in a multilingual environment. This happens not only when I'm searching for software for other people, but also when I'm looking for software, and hardware, for myself.

Thinking about all those companies out there who are unknowingly throwing a huge potential down the drain, I've been building a list of "do"s and "don't"s in my mind for several months. Now I believe I've thought long enough and should set the list in stone (or at least newsprint) somewhere.

Guidelines

There's a big market for software and hardware in non-English speaking countries, and following these suggestions may just make your wares more salable than a competitor's.

Even if you don't currently plan on selling your software directly to overseas markets, you should consider following as many of these guidelines as possible. It would really help people like me.

Here Goes:

1. Don't do anything special with the high or 8th bit when dealing with characters.

Allow a character to be any value from 20h (32) to FFh (255). NEVER use the statement "ch := ch and 7Fh."

You may think this is silly, since all the characters in the alphabet lie below 7Fh (127). You must remember, though, that most foreign languages, even if they do use the same alphabet, have at least a few additions, usually for characters with accents.

For instance, modern Turkish contains 16 characters (counting upper and lower case) that are not in the standard ASCII set. They're placed above 7Fh along with other characters in the "extended" character set.

Even on the IBM PC, some of the

Turkish characters are not represented, but since IBM allows using all 255 positions for characters, I have easily modified all of our IBM machines so they display the complete Turkish alphabet.

At the other (poorly designed) end of the hardware scale are terminals such as the Wyse 50 which (although otherwise perfect) use the high bit of the character as a "protected" bit.

Characters with the high bit set display as normal "character-80h" characters. However, they have a special status. This makes it extremely difficult to use the terminal in a foreign country (without drastic hardware modifications). Wyse seems to have learned its lesson, though. The new Wyse 60 has a full 256 position character set (IBM compatible, even).

Epson has also learned the advantages of a full deck. Older model Epson printers had only the standard ASCII set, plus a few extra foreign characters accessed by changing into different modes. Their newer printers either come with a full 256 character IBM set, or can accept a plug-in module which supplies the expanded set.

Of course, if the software can't deal with these characters, it doesn't matter whether the hardware can display them. Many programs also use the high bit as a "flag" of some sort. WordStar, for example, uses the high bit to indicate the final letter of a word. This makes WordStar Worth-Less. *(I thought it was worthless anyway.)*

Another misuse of the high bit is to signal the end of a string, e.g. "this is a strin","g"+80h. This was quite common in older CP/M programs, where it was used to save that extra byte of memory (you don't need a length byte or a terminating null).

Then, of course, there are the programs that don't use the high bit for anything, so they conscientiously AND it off and then tell you that you're trying to give them bad input. One notable program which does this is ROFF4, which I used to use daily. Then I discovered that it couldn't handle anything with a Turkish character in it. It can't even print my address! (And Dastardly Dave won't send me the source code for the DOS version, even though I've asked for it...) *(Dave's note: Dastardly Dave will have to find it before Dastardly Dave can send it.)*

2. Use the highest level keyboard input routine possible.

Many foreign characters are not represented on a standard keyboard and must be generated by typing alt (or 2nd) and another key. This "function key" combination must be translated by the system software into the proper code number for the desired character.

Problems may occur when, for instance, the system software is doing this translation at the level of a DOS device driver, and your application program is getting its input through the ROM BIOS. You'll be bypassing the translation code and so you'll not receive the desired characters. You'll also have a problem if the translation is done as a keyboard interrupt service routine, while you're reading the hardware directly.

Drop to a lower level of service routines only when absolutely necessary. Use the operating system for keyboard input, if you can. (However, that means you can't make the tab key into an alt key, distinguish alt+j from alt+J, etc.).

Many programs have given me problems in this area. Fortunately, I've been able to overcome nearly all of them so far. My first pass at supporting the input of Turkish characters was with a CON device driver that used the ROM BIOS INT 16h to read the keyboard, and then translated certain alt keys into Turkish characters. This worked fine for all my own programs (because I use DOS to get input), and I was smugly happy for awhile until Ron informed me that the IBM version of Turbo didn't accept Turkish characters.

I then went to the next lower level, trashed the device driver idea, and wrote a program which intercepted all calls to INT 16h. This worked fine with almost all the software we use, including Turbo and Multiplan. I thought the problem was solved. Then I tried Microsoft WORD. Every time I typed alt+i (to give me an undotted lowercase I), it immediately placed me in italics mode.

I didn't ASK to be born, you know.

Knowing the drudgery of writing a program to translate IBM keyboard 'scan codes' into ASCII, function codes, and the like, I gave up on WORD for awhile. Then one day I was playing around with some little experimental "stay resident" programs. I learned that WORD actually does use INT 16h, the only problem is that it checks the "shift key status" first to see whether the alt key is down, and if so it only looks at the scan code (in register AH) instead of the character value (in AL).

In the latest version of my key

(continued next page)

(continued from page 55)

translation program, I intercept the "shift key status" function of INT 16h also, returning "no alt" unless both the alt AND the ctrl key are pushed. Are my worries over yet? Of course not! Now I've just heard that 1-2-3 does not work with Turkish characters, even when the hardware service interrupt is intercepted! This may be a problem of type 1, but since I never use 1-2-3 and neither does anyone at the office, I probably won't try to make it work.

3. If you're writing an operating system or BIOS for a machine, include "hooks" to allow character input and output translation.

This feature became important when I tried to modify a Wang PC to translate WISCII (Wang International Standard Code for Information Interchange) into something a "foreignized" Epson printer could digest.

The problem was that, while Wang's printer driver provided a wonderful character translation table for all 256 possible characters to be output to the printer, it was a 1:1 translation. I needed a 1:7 translation — i.e. to print a "I" I had to change modes (3 characters), type the character, and change back.

Since Wang doesn't freely give out information on any level lower than calling DOS, I couldn't just write my own device driver or do the translation at the ROM BIOS level. Instead, I was forced to spend two days disassembling the device driver to figure out what it did and modify it.

In sharp contrast to this is the IBM PC and its compatibles. All of the BIOS level interfaces are explicitly documented. All the information necessary to write your own device driver can be bought on practically any street corner, as well as from the neighborhood drugstore. When faced with a similar problem on an IBM, I had a translation program written and debugged in under an hour.

4. Allow for a special "Upcase" routine.

Use your operating system's "character upcase" function (if available) to switch the case of all characters. If this function is not available, allow for a special user (or dealer) installed routine to handle switching cases.

Upper casing in standard ASCII is easy, you just see if the character is between 61h and 7Ah and, if so, subtract 20h. This works because the lower case equivalent of a character is positioned exactly 20h above the upper case.

In foreign language character sets, this is not always the case. Some are similar (such as Wang's "WISCII" set where upper case letters are all offset by 10h). But IBM is a mess.

Some character pairs are right next to each other, others are offset by 19h, one by 5h, etc. To make it all even more complicated, the contents of the character set change from country to country. For instance, I mentioned earlier that I added several characters to the IBM foreign character set; this was done by replacing some of the unused (in Turkish) characters with Turkish characters.

An extension of this same idea is the problem of determining word boundaries. Just checking for 'A'..'Z' or 'a'..'z' isn't enough. Probably the easiest is to just figure that any character over 80h is part of a word, or check for being "out" of a word instead by looking for spaces and punctuation.

5. Let programmers create an "alternate collating sequence" for all sorting, indexing, and range checking routines.

This is most easily done by having an array [chr(0)..chr(255)] of char which contains the "position number" of each character in the set. For instance, if the user wanted the "c with a cedilla" to come immediately after c and before d, he might make collate['c']:=65h, collate['c w/cedilla']:=66h, and collate['d']:=67h. Then, instead of using "IF (ch1 > ch2)", just use:

```
IF (collate[ch1] > collate[ch2])
```

In this area, most programs fail miserably, including Turbo Toolbox (although you can fix that yourself, since Toolbox comes with source code included). One program that includes this facility is the BTrieve file management package; although I haven't tried it, I have a feeling that if they went to these lengths, it must be at least semi-decent.

6. If you display text in graphics modes, read the character fonts from a separate data file and include instructions (and maybe even a font editor program) so the characters can be modified to fit the local standard.

WORD is a real loser in this category (at least the version I've tried is). Not only is the character set hidden somewhere deep inside the program, but they used all kinds of programming tricks to keep me from tracing through it with DEBUG to find the appropriate table.

Turbo Graphix Toolbox has done this part half way. It's good that they put their character fonts in a separate file, but they should have included a program to edit the fonts. I ended up having to write font editor programs for Turbo Graphix.

This rule also applies to any hardware that displays characters. If you're designing a printer, a terminal, a display adaptor, or whatever — put the character generator in a separate ROM and include information on the ROM's content and how to modify it.

Many companies, such as Wyse, have wonderful tech support departments that will gladly give you complete information on this type of thing. Others, like Epson, refuse to reveal anything at all, and even seem to think that you're somehow "stepping beyond the bounds" of honest business if you try to modify their ROMs.

7. Support European format dates.

In the U.S., we ("you," I should say) display dates in the form mm/dd/yy, but in most European countries they're displayed as dd/mm/yy.

MS-DOS has a system call (38h) which returns country specific information such as this to your program. Besides the format of date, it also tells you whether to use 12 hour or 24 hour time, as well as what the country's currency symbol is and whether to put numbers in the form "#,###,###.##" or "#.###.###,##".

If the operating system you're working with can't provide this information, then let the user read it in from a data file and give instructions in the manual on how to change it.

8. Place all menus, prompts, and error messages in an easily modified text file. Include directions in the documentation on how to change these messages.

This should even extend to the proper responses to "yes/no" questions. In Turkish, for example, Yes is "Evet" and No is "Hayir." All questions that usually look for "y" or "n" must be able to reconfigure themselves to accept (in our case) "e" or "h." (Make sure you allow for the unlikely case of both "yes" and "no" starting with the same letter.)

Turbo Pascal wins in this category. They put all their messages in a data file, and even give instructions in the manual on how to modify the message file. The only thing they forgot was the ability to change "Y" and "N."

Most people think that menu translation is the main problem in converting a program to work in a non-English environment. By now you probably see that it's one of the most trivial parts of the process, and may even be the least important.

A high percentage of people using computers in foreign countries have at least a basic knowledge of English. Many of them speak it fluently. They can read menus and help files in English, but they will not (or at least should not) even consider using your program if it can't handle the complete alphabet of their native language.

9. If you want to get really tricky, allow for right to left input and output, as well as left to right.

This ability would make your display terminal, word processor, or database program much more useful in those countries immediately south and east of me (whom I won't mention together in the same paragraph for fear that they'll all decide to hate me. If they have enough time to take out from hating each other, that is...)

Reasons

You may not pay attention to all of my suggestions in the programs you develop; you may not even pay attention to any of them. But at least you'll be aware of the implications of your design. I sure wish someone had told MicroPro about these things before they wrote WordStar. (No matter how much I despise the program otherwise, it does seem to be an industry standard. In English speaking countries anyway.)

By the way, in case you're wondering, I'm not exactly a saint in following all these suggestions either. Well, NOW I am, but EXPRESS (which was written before I came to Turkey) breaks almost every one of the rules. Fortunately I have the source code, so I modified it to solve my problems. EXPRESS is now the only editor I've found that can input, display, and print the complete Turkish (or any other) character set with no problems. I have learned.

I take that back. I guess the Turbo Pascal editor works too.

Limitations

Even if you pay attention to these, your program will still be unusable in many countries, namely the ones with more than 256 characters in their alphabet and ones that read down the page instead of across. I haven't spent enough time in those parts of the world to consider myself any kind of an authority on that subject. Yet.

■ ■ ■

By Stephen Leon
200 Winston Drive
Cliffside Park, NJ 07010

Who Owns Public Domain Software?

More public domain releases from both PC/BLUE and SIG/M, but Steve also takes a look at the copyright protection versus public ownership of public domain software. Who owns what?

The Turbo Exchange BBS in Syracuse, New York is not a public system. Clarence Rudd, the SYSOP, maintains it strictly for a close group of scientists and engineers. It seemed a shame that such a wealth of information was unavailable to the public. So we got Clarence to send us a backup of the system, and we've produced six volumes of Turbo Pascal programming. Three of them are SIG/M Volumes 291, 292, and 293. The other three are PC/BLUE Volumes 260, 261, and 262.

The SIG/M disks contain 44 files, and the PC/BLUE volumes have another 40. Notwithstanding the SIG/M policy of releasing both CP/M and MS-DOS programs, we thought it advisable to put the generic and Z80 type material out on SIG/M and the MS-DOS material on PC/BLUE. There was just too much to do otherwise.

SIG/M Turbo Pascal Releases

Volume 291 contains a wealth of Turbo routines. Included are routines to access CP/M directories from Turbo, clean up your source code, read dBASE II files from Turbo, convert between number bases, trap I/O errors, and get English answers to Turbo error codes.

Also on the disk are a Turbo line editor, Epson and Okidata print formatters, a WordStar control code remover, a math expression evaluator, a memory adjuster, a tutorial on using linked lists and pointers, an explanation of how to make Turbo leave your CP/M CCP alone, a library of clock routines, and a few other things including an adventure game and a Master Mind game.

Volume 292 includes a 103K library of Turbo programs for engineers and scientists, a screen generator, and more. Volume 293 has three utilities packages, two screen generators, and a WordStar table of contents generator.

PC/BLUE Turbo Pascal Releases

PC/BLUE 260 includes routines to change file attributes, break Turbo's 64K limit, do ANSI graphics, and copy files. It also has speed boosters, windows for Turbo menus, time and date routines, com port routines, and a duplicate file finder.

PC/BLUE 261 lets you format disks from Turbo programs, branch to and from Turbo programs, stop 'Abort, Retry, Ignore?' in Turbo, execute DOS and other commands in Turbo, and manage your windows. You also have a quick editor, a way to spell check a full file with Lightning, Turbo ZAPS, random numbers help, an 'almost' text compare, and a how-to on windows in Turbo.

PC/BLUE 262 has a window manager with background tasking, a 160K collection of Turbo routines, high speed video I/O, a source code debugger, and a few other odds and ends.

Clarence has assembled quite a collection of Turbo material and we thank him for making it available.

For Historians & FORTRAN Fans

David A. Danello has contributed to the SIG/M library (Vol. 288) a collection of utilities for Microsoft FORTRAN-80. They consist of a 59K library of FORTRAN subroutines and a 60K library of FORTRAN tools. Also on the disk is a 99K library of miscellaneous programs, including astronomy programs and Morse code.

Ted Campbell has provided Stardate (Vol 290). It is an accurate calendar, almanac, and limited astronomical data program for historians, writers, and others, covering the period from 2500 BC to 2500 AD. The disk contains both CP/M and MS-DOS versions of the program. Also on Volume 290 is a Kaypro-to-Mainframe program by Dave May.

Other New SIG/M Releases

The ACGNJ-BBS that I've been running (201-886-8041) is now up to 122 megs. I'm a great one for utility programs, and recently did a sweep of the system to produce SIG/M Volume 289 — 'Utilities on the ACGNJ BBS System.' The disk contains COMPARE, a Z80 disk compare; EDFILE, a Z80 HEX and ASCII file editor; FBAD, a bad sector locator which improves upon FINDBAD; FILE, which searches all drives for a file; SFILE, which not only searches all drives for a file, but also searches .LBR files, all user areas, etc.; NEWCOPY, a fast copy program; and SETDRU, to set drive and user path in CP/M 80. Also on the disk is an MDRIVE program for a Rainbow 100 and StarTrek in Nevada BASIC.

The final volume in this release series is SIG/M 294. It includes CRUNCH, an ARC-like Z80 file compression utility; and LBLMKR, a Turbo Pascal label maker.

Contest For Students

The Trenton Computer Festival will award a computer system to the high school or junior high school student who submits the best essay on the topic "My Computer's First Essay." The prize is a Datafox model 88-2 donated by Pierce Phelps of Philadelphia. The Datafox is a full IBM compatible system with a monitor and keyboard. It has a dual speed processor (4.77/8 MHz switchable), 2 disk drives, I/O ports, clock, Hercules graphics, etc.

Each entry should contain (on the back) the name and address of the entrant, the grade of school attended, and the date of birth. Essays should be the work product of the student, but teachers are encouraged to conduct class contest entry exercises, and teachers and parents may assist the entrant. The essay can be as long or as short as the contestant feels is necessary (although a length of no more than one page is suggested).

This is the third year TCF has run a student essay contest. It has been quite a success in the past. This type of essay contest is a good exercise in creative writing with a first rate prize.

Deadline for entries is Sunday, April 12. (The 1987 Trenton Computer Festival is Saturday and Sunday, April 11 and 12, and entries can be submitted at the Festival up until 4 pm.) Mail entries should be sent to TCF Computer Contest, Trenton State College, Hillwood Lakes CN 4700, Trenton, NJ 08650-4700.

More From PC/BLUE

PC/BLUE Volumes 251 and 252 contain PAYROLL USA, a payroll system for a small business. Volume 253 is an update to Jerry Medlin's PC Accounting Systems (PC/BLUE vol. 231), with an update on the general ledger and accounts receivable program (same payroll module). Volumes 254 and 255 contain version 2.7 of PC-Write (replacing volume 202).

Volume 256 has the fourth release of the PC Magazine benchmark series. DBS-KAT, a super capacity diskette cataloger, is updated to version 2 on Volume 257, replacing volume 197. Volumes 258 and 259 contain version 2.0 of FANSI-Console from Hersey Micro Consulting, Inc. (replacing volume 161).

Volume 263 contains LQ version 2.24. It promises near letter quality from dot matrix printers. KWIKSTAT, a statistical analysis package, is on volumes 264 and 265.

I can see all of Micro C's readers ordering 266 and 267, which contain DROEGE — a name only an engineer could dream up. DROEGE stands for Design Robot for Origination of Exacting Graphic Engineering. It designs printed circuit layouts.

Volume 268 contains a utility to control screen color, some Turbo Pascal utilities, a mailing list system, and a portfolio management system. Volume 269 contains ProComm version 2.42 (replacing volume 200). If you haven't used ProComm on your PC you're missing something. It's really great!

How To Order

SIG/M Volumes are available on 8" SSSD Disks for $6.00 each ($9.00 foreign) directly from SIG/M, Box 97, Iselin, NJ 08830. They're also available in most 5" formats. The charge for 5" disks is $7.00 per volume. However, for SSSD formats, or any format which requires more than one disk, please add another $2.00 per volume. Printed catalogs are $3.00 each ($4.00 foreign).

PC/BLUE Volumes are $7.00 each ($10.00 foreign). The printed catalog is $5.00. Both are available from the New York Amateur Computer Club, Box 100, Church Street Station, New York, NY 10008.

Each group has a disk catalog (Volume 0) available at the price of a standard disk volume. This catalog

(continued next page)

(continued from page 59)

volume is usually more current and more readily available than the printed catalog.

The ACGNJ BBS

All of the recent SIG/M and PC/BLUE releases are also available on the ACGNJ BBS at 201-886-8041. No pre-registration is required to download or upload. SIG/M volumes are stored as SIGxxx.LBR. PC/BLUE volumes are PCxxx.ARC. In addition, any SIG/M or PC/BLUE volume (or any file from any volume) will be put up on request.

Volumes can be downloaded in their entirety or, in most cases, component files can be downloaded separately. The board also has a considerable quantity of software not in the SIG/M or PC/BLUE library, including more than 200 AMIGA files.

One of the fastest ways to make donations to either the SIG/M or the PC/BLUE library is to upload to this board.

The State Of Public Domain

Public domain software began as a way for programmers to help other programmers avoid reinventing the wheel, but sometimes it turns into crass commercialism.

Not that this has happened suddenly. I remember when SIG/M started putting copyright notices on volumes because the commercial company that was then issuing the CPMUG volumes started releasing SIG/M disks under their own label. (Editor's note: That's Lifeboat, I believe.)

They finally shut down CPMUG, but we continued issuing SIG/M volumes with copyright protection. Our primary means of distribution has been via the non-profit clubs in our distribution organization. There were a few commercial organizations which distributed SIG/M software without permission, but we didn't threaten to sue them. We felt that, by participating in the chain of distribution, they were helping SIG/M achieve one of its goals — the wide distribution of non-commercial software at minimum cost to the user.

National Public Domain

What started this off was a press release from the National Public Domain Software Rental Center announcing they were going out of business because of the threat of litigation from PC-SIG. Who is PC-SIG? Well, according to my interpretation of a letter from one of my brother lawyers to Don Johnson of the Public Domain Copying Company, PC-SIG claims to 'own' just about every bit of public domain software and freeware that is out on the market!

We lawyers are kind of inclined to stretch the rights of our clients to the limit. There is no question that Don Johnson has no right to distribute PC/BLUE, SIG/M, or PC-SIG volumes under the respective PC/BLUE, SIG/M or PC-SIG names, because all of us by putting a copyright notice on the anthology preserve our rights. However, of the three, only PC-SIG is a commercial (for profit) venture. SIG/M and PC/BLUE are strictly non-profit membership clubs.

Yet look at the PC-SIG ads. Look at their name. In the computer world, 'SIG' stands for Special Interest Group. They advertise a one year membership for $20. A membership in what? It sure sounds like a membership in a computer club's Personal Computer Special Interest Group. 'Membership' in PC-SIG is valuable because it includes a newsletter and software support, but why not call it a 'subscription' to avoid the confusion?

CompuServe

Equally shocking was a story in InfoWorld that CompuServe claims a copyright to all of the public domain software uploaded to it. This story was so absurd that I called CompuServe's lawyer. He promised a statement on the subject within two days. That statement did not arrive by press time, and a layman might suspect that PC-SIG and CompuServe are going to litigate which of them owns what neither of them owns.

However, after reading the CompuServe rules (GO RULES), I suspect that InfoWorld is wrong. After reading the PC-SIG position, it sounds like another case of a full moon on the West Coast. Let them go out in the daytime and review their position before they push it further.

There is nothing wrong with commercial ventures distributing public domain software. Micro C has a fine library of public domain software available for distribution. So does PC-SIG.

I agree that Don Johnson should not distribute software under the PC-SIG label anymore than he should do so under the Ashton Tate label. However, Don has every right to distribute the public domain software on those disks, unless the contents are truly proprietary to PC-SIG.

Consider this from the author's standpoint. PC-SIG appears to claim ownership of everything on its disks. I noticed that they put out a volume containing PC-Outline (as did PC/BLUE). Brown Bag has essentially the same program out in a commercial package. Following PC-SIG's thesis to its logical conclusion, should they not sue Brown Bag? I have no love for freeware authors, but if I were one of them I'd think twice before I let PC-SIG include my material in their library.

Litigation is an expensive matter as the profession is well-paid for its work. Sometimes it does pay to litigate rather than fold for fear of the costs. Moreover, a copyright suit would properly belong in Federal Court. The way things have been going in the Federal Court system, PC-SIG could very well get hit with the expenses of a litigant defending against their law suit. Maybe that's the way to resolve this issue.

I would prefer, however, that PC-SIG rethink its position and adopt a 'live and let live' attitude for the benefit of both users and authors. By that I mean, it should clarify to the public that it is not a not-for-profit 'user group.' It should also refrain from making claims to what it does not own. It should behave as a responsible member of the computer community and not as a bully.

■ ■ ■

Adding Automatic Capitalization

Adding Features To The FTL Modula-2 Editor

By Mark J. Boyd
Computer Science Department
University of North Carolina at Asheville
Asheville, NC 28804

Here's an addiction that could be deadly. Once you have it you won't want to give it up. Automatic indentation and capitalization may not sound like much, but wait till you have them. Built into the editor, yet.

In the Kaypro column in issue #32 I showed how the FTL Modula-2 Editor could be customized to allow proper use of a Kaypro II keypad. In this article we'll see how to make it handle capitalization and auto indenting as well.

The previous modification was quite Kaypro specific, but this one should work with either version of the FTL Modula-2 Editor, CP/M or MS-DOS.

You could also install this in other editors (if you have source) or make it a standalone program.

I use a stand alone version with TDI Modula-2 on an Atari 1040ST, but I'm also working on adapting the FTL Editor to my Atari system. This is possible because Modula-2 code is quite portable and easy to modify. I'm really excited about having this editor on a system with 1M of RAM and a 50-line screen. You didn't know the Atari Monochrome display supported 50 lines? Well it does, and they are quite readable.

Back To FTL

I love Modula-2, but I hate having to capitalize all of its reserved words. I have enough trouble just spelling them right. When I have to get them into upper case as well, my error rate goes way up. I'm sure many of you have the same problem. On the other hand, I do like having the reserved words capitalized, they really stand out in the source.

Ford and Wiener, in their excellent text *Modula-2, A Software Development Approach* (Wiley, 1985), offer a solution. They have designed a file processor, in Modula-2, which will convert all Modula-2 reserved words to upper case. Neat.

But if the Modula-2 editor and compiler are linked together, it's a pain to leave the editor, do the capitalization, and then return to the editor to start the compiler. So I integrated the capitalization into the editor.

Much To Do

The capitalization routine scans the code as it is entered. With the editor doing capitalization, I can tell immediately when I've misspelled a reserved word or accidentally used one as an identifier.

Auto indentation makes pretty printing almost automatic. You just type in

Figure 1 - Modifications to KEYBOARD.MOD

```
Near the beginning of this module there is a long list of modules
IMPORTed from EditControl, add: Scan, AutoIndent

At the end of this module there is the intialization of the keyboard
dispatch table. Add the following:

        MainTable[5cx]:=Scan;          (* '\' - new scan function *)

and change the LF assignment to:

        MainTable[0ax]:=AutoIndent;    (* LF- new autoindent function *)
```

Figure 2 - Modification to EDITCONTROL Modules

```
(* THESE GO IN AT THE END OF THE DEFINITION MODULE - EDITCONT.DEF*)
PROCEDURE Scan;
PROCEDURE AutoIndent;

(* THESE GO IN THE IMPLEMENTATION MODULE - EDITCONT.MOD *)
(* THIS GOES AT THE BEGINNING  *)
FROM scanner IMPORT scan;

(* THESE GO JUST BEFORE THE INITIALIZATION SECTION, NEAR THE END *)

PROCEDURE Scan;  (* Capitalizes keywords and autoindents next line *)
BEGIN
        scan(CurFilePtr^,TRUE);(* CAP the keywords in the file *)
        RedoLine;              (* update the displayed line    *)
        InsOneChar(CR);        (* put in a CR                  *)
        scan(CurFilePtr^,TRUE);(* put the intro in the file    *)
        RedoLine;              (* update the displayed line    *)
        END Scan;
PROCEDURE AutoIndent; (* Autoindents next line *)
BEGIN
        scan(CurFilePtr^,FALSE);(* Get the intro for the line  *)
        InsOneChar(CR);         (* put in a CR                 *)
        scan(CurFilePtr^,FALSE);(* put the intro in the file   *)
        RedoLine;               (* update the displayed line   *)
        END AutoIndent;
```

a couple of spaces each time you enter a control structure, and delete the same number each time you exit one.

After entering a line of code I can choose capitalization or indentation. If I use a backslash instead of a carriage return, the program capitalizes the current line and indents the next. If I use a linefeed, it just indents the next line.

Since all keyboard interpretation is handled by one module in the FTL Editor, you can pick the keys you want to use.

In fact, the FTL editor's modular structure makes it easy to add new features. The editor modules involved in this mod are Keyboard, EditControl, EditDisplay, and MakeEdits.

EditDisplay and MakeEdits are low level modules used by EditControl to control the display and the file. Keyboard connects procedures in EditControl with user keystrokes.

I added two new procedures to EditControl, Scan and AutoIndent. Then I set up Keyboard to call them when it saw a backslash or line feed. See Figures 1 and 2.

Another new module, scanner, IMPORTs procedures from MakeEdits. These procedures allow scanner to move around in the file, read characters from the file, insert characters in the file, and delete characters from the file. scanner EXPORTs scan, which is IMPORTed by EditControl. It's used by the procedures Scan and AutoIndent. *Notice that Scan and scan are two different procedures. Modula-2 is case sensitive.*

EditControl also IMPORTs screen procedures from EditDisplay. One of these, RedoLine,is used in the Scan and AutoIndent procedures.

Finally, EditControl has a procedure, InsOneChar, which is used by both Scan and AutoIndent to put carriage returns in the file. Figure 3 and Figure 4 contain the DEFINITION and IM-

(continued next page)

Figure 3 - The New Module Definition

```
DEFINITION MODULE Scanner;

(* A routine which alternates between:
        scaning a line of Modula 2, capitalizing reserved words and
        autoindenting the next line with the indentation found in
        the first pass. It is always called twice, but other work
        must be done between the calls, see editcont.mod.
*)

FROM MakeEdits IMPORT EditFile;

PROCEDURE scan(VAR f:EditFile; full:BOOLEAN);
```

Figure 4 - The New Module Implementation

```
IMPLEMENTATION MODULE Scanner;

(* based on the FSM approach given by Gary A. Ford and Richard S. Wiener,
   MODULA-2 A SOFTWARE DEVELOPMENT APPROACH, Wiley, 1985, page 350 *)

FROM MakeEdits IMPORT EditFile, DelChars, InsChars, BackOneChar,
                      ForwardOneChar, GetCurrentChar;

CONST
   bufsize   = 15;   (* Max length for Modula-2 'symbol' *)
   introsize = 40;   (* Half screen width - to allow 40 spaces and/or tabs *)
   LF        = 0ax;  (* Last character in EOL marker in file *)
   SPACE     = 20x;  (* White space characters *)
   TAB       = 09x;
TYPE
   (* These are the states of the Finite State Machine *)
   states   = (start, insym, incom, encom, excom);
   barray   = ARRAY [1 .. bufsize] OF CHAR;
   iarray   = ARRAY [1 .. introsize] OF CHAR;
   CHARSET  = SET OF CHAR;

VAR (* all variables are global in this module *)
   state    : states;  (* state of FSM *)
   ch       : CHAR;    (* current character in buffer *)
   delim    : CHAR;    (* holds string delimiter *)
   buffer   : barray;  (* symbol buffer *)
   intro    : iarray;  (* intro buffer  *)
   buflen   : CARDINAL;(* number of characters in symbol *)
   introlen : CARDINAL;(* number of characters in intro to line *)
   j,k,l    : CARDINAL;(* counters for various loops *)
   symset   : CHARSET; (* the set of characters used in symbols *)
   flag,sec : BOOLEAN; (* flag is gnl purpose BOOLEAN, sec is for pass *)
   sp : ARRAY [2 .. 15] OF CARDINAL;(* Array of indexes into the CHAR array *)
   a2 : ARRAY [0 .. 13] OF CHAR;  (* this is really one big CHAR array *)
   a3 : ARRAY [0 .. 57] OF CHAR;  (* but there is no easy way to do    *)
   a4 : ARRAY [0 .. 75] OF CHAR;  (* initialization on any array of    *)
   a5 : ARRAY [0 .. 49] OF CHAR;  (* 350+ characters - so I do it on   *)
   a6 : ARRAY [0 .. 41] OF CHAR;  (* ten smaller arrays. Obviously this*)
   a7 : ARRAY [0 .. 35] OF CHAR;  (* requires that bounds checking be  *)
   a8 : ARRAY [0 .. 15] OF CHAR;  (* turned off and that arrays be     *)
   a9 : ARRAY [0 .. 17] OF CHAR;  (* be stored contiguously in memory. *)
   a10: ARRAY [0 .. 29] OF CHAR;
   a14: ARRAY [0 .. 15] OF CHAR;

PROCEDURE scan(VAR f:EditFile; full:BOOLEAN);
(*
   This procedure alternates between scanning a line and writing the
   intro from the line just scanned - IF full THEN auto capitalization
   is performed as part of the first pass. ELSE the first pass just
   gets the intro string for use on the second pass. f is the file
   being edited.
*)

PROCEDURE check ():BOOLEAN;
(*
  this procedure checks a symbol to see if it should be capitalized
*)
```

(continued from page 63)

PLEMENTATION MODULEs for scanner.

If you're confused by this discussion, spend some time studying the editor structure. It took me quite a few hours to figure out how to integrate the new functions. It's a nice clean structure, but a text editor is complicated.

scanner

scanner's structure is a bit strange. It exports a single procedure which alternates functions on successive calls. The first time it is called it scans the line of code in the file. I think of this as its first pass.

On this pass it looks for spaces or tabs, and, if a flag (full) is set, it capitalizes the reserved words. The second time it is called it sticks white space into the beginning of the (next) line.

Both Scan and AutoIndent call scan twice. The difference is that AutoIndent doesn't ask scan to do capitalization.

The algorithm in scanner works a bit like a spelling checker. It scans the line of code, checking everything that could be a reserved word against a list of reserved words. Unlike a spelling checker, it must keep track of not only the current word, but some of the context in which the word occurs. In Modula-2 we have strings and comments where a reserved word could not occur. For example, MODULE may be a reserved word in Modula-2, but it is not when it occurs in a string or a comment. Thus:

DEFINITION MODULE test;
(* This is a test module *)

and

WriteString ('This module is done');

A deterministic finite state machine (FSM) is just what we need. If we think of the system as being in one of a finite number of possible states, then the state of the system (for example "in a comment") is all the information we need to decide how to process the current input.

Each state has its own rules. These

Figure 4 - The New Module Implementation (continued)

```
BEGIN
   flag := FALSE;   (* will be set true if match on all characters *)
(* Put a marker after the end of the symbol to stop comparison *)
   buffer[buflen+1] := '!';
(* Then point to array position corresponding to buffer length *)
   k := sp[buflen];
(* Now scan all entries of that length for a match on all characters *)
   LOOP
      l := 0;
      WHILE a2[k+l] = buffer[l+1] DO l := l+1 END;
(* If a match on all characters is found then return TRUE *)
      IF l = buflen THEN
         flag := TRUE;
         EXIT;
      END
      k := k + buflen;
(* Else if all keywords of that length checked then return FALSE *)
      IF k >= sp[buflen+1] THEN EXIT END;
   END
   RETURN flag
END check;

BEGIN (* PROCEDURE scan *)
(*
   The BOOLEAN sec keeps track of which pass we are on.
*)
   sec := NOT sec;
   IF sec THEN          (* second pass *)
      IF introlen > 0 THEN InsChars(f,intro,introlen) END;
   ELSE                 (* first pass  *)
(*
   Place a marker in file - to mark current position and to make sure
   that all lines ending with a symbol return to the start state.
*)
   intro[1]:=20x;
   intro[2]:=00x;
   InsChars(f,intro,2);(* use procedure InsChars from MakeEdits *)
(*
   Move back to beginning of line.
*)
   WHILE BackOneChar(f) AND (GetCurrentChar(f) # LF) DO END;
   introlen := 0;
(*
   IF NOT BEGINNING OF FILE THEN move forward to first character of line.
   IF BOF THEN we are already on the first character of the line.
*)
   ch := GetCurrentChar(f);
   IF ch = LF THEN
      flag := ForwardOneChar(f);
      ch := GetCurrentChar(f)
   END;
(*
 Now we save the intro to the line ( white space )
*)
   WHILE ((ch = SPACE) OR ( ch = TAB)) AND (introlen < introsize) DO
      INC (introlen);
      intro[introlen] := ch;
      flag := ForwardOneChar(f);
      ch := GetCurrentChar(f)
   END; (* WHILE *)
(*
  There are two possibilities on first pass: if not full then
  we just get the intro. If full we also do capitalization.
*)
   IF full THEN
      buflen:=0;          (* Initialize symbol buffer *)
      state := start;     (* and state of FSM *)

      WHILE (ch # 00x) DO (* Now process line until end marker is found *)
(*
 This is basically Ford and Weiner's FSM, but without handling of nested
 comments - their design processes a file, this one just does a line.
*)
```

```
        CASE state OF
        start :
           CASE ch OF
              'a' .. 'z' :
(*
   There are three ways to exit the start state; ecountering a symbol,
    ecountering a comment, or ecountering a string. If a letter is
    encountered,it starts a symbol. We go to the insym(bol) state.
*)
                           state := insym;
(* Symbols are stored in buffer for comparison to the keywords *)
                           INC(buflen);
                           buffer[buflen] := ch;
              |
(*
   An open paren may be the start of a comment, state is en(tering)com(ment).
*)
              '('        : state := encom;
              |
(*
   A single or double quote is the start of a string literal.
   The new state is instr(ing). We must record the delimiter.
*)
              '"',"'"    : state := instr;
                           delim := ch;
           END |
        insym :
(*
   When we are in a symbol, we continue as long as the characters
   encountered are lowercase letters - we do not allow mixed case
   or digits. Each letter is stored in the buffer.
*)
          IF (ch IN symset) AND (buflen < 14 ) THEN
               INC(buflen);
               buffer[buflen] := ch;
(*
   When a character other than a letter is encountered, we check to see if
   the symbol in the buffer is in our table. If it is we CAP the symbol in
   the file. The function BackOneChar and the procedures DelChars and
   InsChars used for this process are from the module MakeEdits.
*)
          ELSIF (buflen > 1) AND (check()) THEN
               FOR j := 1 TO buflen DO         (* move back thru file  *)
                  flag := BackOneChar(f);      (* flag is just a dummy *)
                  buffer[j] := CAP(buffer[j])(* CAP the buffer too    *)
               END;
               DelChars(f,buflen);             (* remove the symbol    *)
               InsChars(f,buffer,buflen);      (* and replace it       *)
               buflen := 0;                    (* reset for next symbol*)
               state := start;                 (* back to start state  *)
(* If the symbol is not in the table then we leave the file alone *)
          ELSE
               buflen := 0;                    (* reset for next symbol*)
               state := start;                 (* back to start state  *)
          END |
        instr :
(* When in a string we just watch for the string delimiter *)
           IF ch = delim THEN
              state := start;
           END |
        encom :
(*
   When en(tering)com(ment) we go to incom(ment) if the next character
   is '*', instr if the next character is a delimiter, and insym if it
   is in symset, else back to start.
*)
          IF ch = '*' THEN
            state := incom;
          ELSIF (ch =  "'") OR (ch = '"') THEN
            state  := instr;
            delim := ch;
          ELSIF (ch IN symset) THEN
            state := insym;
            INC(buflen);
            buffer[buflen] := ch;
```

rules are checked each time the system accepts input. They determine what the system does with an input.

An input may cause a change of state, some action in the current state, or it may be ignored. FSMs are very powerful tools for designing many types of computer programs. Ford and Wiener use one as the basis for their capitalization algorithm.

Six States

1. Start. This is the default state.
2. In-string. This occurs when a ' or " is found, and continues until the same character is repeated. (Entered from start.)
3. Entering-comment. When a "(" is found, there's a chance it's the beginning of a comment. (Entered from start.)
4. In-comment. If the next character after the "(" is "*", we are indeed in a comment. (Entered from entering-comment.)
5. Exiting-comment. When we find another "*", it may be the beginning of the end (of the comment). (Entered from in-comment.)
6. In-symbol. This is entered when a lower case letter is encountered. It keeps track of the characters entered, and continues until something other than a lower case character is found. (Entered from start.)

I used a linear search to compare the symbol with the reserved words of the same length. If there's a match, the symbol is deleted from the file and the uppercase reserved word is inserted. If there is no match, the symbol is discarded. Since the program only checks symbols that are all lowercase, you can use mixed-case symbols with the same spelling as reserved words.

The processing of a line is fairly involved, but still fast. On my 5-MHz Kaypro 8, it appears to be instantaneous.

I type the line and hit the backslash. All the reserved words switch to upper case and the cursor appears under the first non-blank character in the line. If you hit line feed, you just get the auto indent.

I find it easy and natural to use these new features when entering Modula-2 source. They do not inter-

(continued next page)

Figure 4 - The New Module Implementation (continued)

```
          ELSE
            state := start;
          END |
        incom :
(* When in comment we watch for '*', new sate is  ex(iting)com(ment) *)
           IF ch = '*' THEN
              state := excom;

           END |
        excom :
(* When exiting a comment we must find ')' for the new state to be start *)
           IF ch = ')' THEN
              state := start;
           ELSE
              state := incom;

           END
        END (* CASE *)
(* Get the next character to process *)
        flag := ForwardOneChar(f);
        ch := GetCurrentChar(f);
     END (* WHILE *)
   ELSE (* NOT full *)

(* If we are only getting intro we must skip over the rest of the line *)
      WHILE (GetCurrentChar(f) # 00x) AND ForwardOneChar(f) DO END;
   END (* IF *)
(* Finally, we remove the marker we inserted at the end of the line *)
      flag := BackOneChar(f);    (* We are on second character *)
      DelChars(f,2);             (* Remove both characters *)
   END (* IF *)
END scan;


BEGIN   (* initialization of data structures used by scan *)
   symset := CHARSET{'a' .. 'z'}; (* only lower case will be considered *)
   a2 := 'bydoifinoforto';
   a3 := 'absadrandcapchrdecdivendforincmodnewnilnotoddordsetvarval ';
   a4 := 'bytecasecharelseexclexitfromhalthighinclloopprocrealsizethen
          truetypewithword';
   a5 := 'arraybeginconstelsiffalsefloattrunctsizeuntilwhile';
   a6 := 'exportimportmodulerecordrepeatreturnsystem';
   a7 := 'booleandisposeintegerpointerprocess ';
   a8 := 'cardinaltransfer';
   a9 := 'procedurequalified';
   a10 :='definitioniotransfernewprocess';
   a14 :='implementation  ';
   (* These are the positions 'within' a2 where each of the above arrays
      start. This is used to shorten the search by only searching that
      part of the table which contains entries of the same length as the
      symbol. These positions are correct if the arrays are stored adjacent
      to each other in the order they were declared. The arrays all have
      an even number of bytes. This allows the same declaration to work
      on a Z80 ( byte aligned variables ) and a 68000 ( word aligned ).
      I think it should work on other 16 bit processors as well, but I
      haven't tried it. *)
   sp[2] := 0; sp[3] := 14; sp[4] := 72; sp[5] := 148;
   sp[6] := 198; sp[7] := 240; sp[8] := 276; sp[9] := 292;
   sp[10] := 310; sp[11] := 354; sp[12] := 354; sp[13] := 354;
   sp[14] := 340; sp[15] := 354;
   sec := TRUE; (* set it up to start on first pass *)

 END Scanner.
```

(continued from page 65)

fere in any way with the use of the editor for non Modula-2 text. There is only one drawback: you lose about 2K from the maximum file size.

This shouldn't be a problem for the MS-DOS version. My system (61K TPA), can only edit files up to 16k.

If you must have 20K files and these features, or if you have an unusually small TPA, you might consider removing something from the editor. The macro features use over 4K. Just eliminate the Macros related procedures from Keyboard and DoMenu.

This reduces the editor to less than 32K, even with my new functions. The only problem is the input function, ReadAChar(ch), which is used in Keyboard. It can be replaced with

```
REPEAT
  BusyRead(ch);
UNTIL ch <> 0;
```

Before You Begin

Make a copy of your editor disk and compile the unmodified editor. (It's not difficult, but you may have a disk space problem).

Now, modify the Keyboard.mod, EditCont.def and EditCont.mod modules (following the instructions in Figures 1 and 2), then enter (or better yet get copies of) scanner.def and scanner.mod.

Next compile these modules and everything higher up in the editor structure (everything that comes after the modified modules in RECMPED.SUB).

Finally, link the editor (last step in RECMPED.SUB).

You will be rewarded with an editor that really makes Modula-2 source entry a lot easier. And you will have learned enough about the editor to be able to design your own modifications.

■ ■ ■

By Ron Miller
1157 Ellison Dr.
Pensacola, FL 32503

C'ing Into Graphics

Now that you've got graphics, all kinds of graphics, whatcha gonna do about it? Be impressed? You can do better than that.

Of late I've been toying with the creation, alteration, and presentation of screen graphics on IBM PC compatibles. Nothing very ambitious, mind you, but enough to excuse more excursions into C. Screen graphics and C go as well together as — uh — the old Kaypro Tandon drives and BDOS errors. C is fast, provides a rich set of bit operators, and is easy to soup up with assembly language.

All three traits are essential. Since a Hercules graphics display contains 32,768 bytes, and every byte contains 8 bits, we are talking about LOTS of calculations to wander across the screen.

The standard color graphics display involves a piddling 16,384 bytes — but that's a quite few "for" loops, too. A clear and simple representation of bit arithmetic is essential if one needs to alter parts of the bytes in a memory-mapped display.

Assembly language is the greatest pain of all, the fly in my personal ointment, and yet I haven't found a way to avoid it. That's mostly because the DOS graphics utilities, like all the rest of the DOS screen utilities, are so darned slow. Besides, DOS doesn't even acknowledge the existence of such a de facto standard as the Hercules graphics card.

To get full use of one of those delightful $80.00 Hercules clones from Taiwan (I'm gazing at one doing its thing right now), you have to program down to the bare metal.

So I've been experimenting with the ways to manipulate graphics with C, and I mean experimenting. The following discussion is nothing more than progress notes from a rank amateur.

This time I'll offer general methods and observations; next time I'll get practical and show some ways to capture, edit, and redisplay graphics screens from commercial programs such as Lotus. We'll even do some resident-C work.

I know there are faster algorithms and nice commercial and public domain utilities that do much of this for you. But anyone who reads Micro Cornucopia must be a bit of a do-it-yourselfer, and those of us who read or write C columns must be the worst of the lot. (Editor's note: Best of the lot.)

Getting Into Graphics

The graph hacker needs first to know how to get into the proper graphics mode. Though you can of course dedicate different programs for different graphics cards and leave the choice up to the user, I find it easy to use an interrupt 11h call to find out what video card is available.

If we ignore the possibility of an EGA card or some exotic variation like Plantronics — something that's going to get harder and harder to do in the years ahead — we face three possibilities:

1. IBM color graphics card, or equivalent.
2. IBM monochrome card, or clone.
3. Hercules graphics card, or clone.

On returning from an interrupt 11h call, if bits 4 and 5 of the ax register are set to 1, it's a monochrome or a Hercules card, with its screen memory beginning at 0xb0000. Otherwise it's color graphics, beginning at 0xb8000.

My usual trick is to declare a global unsigned "scrseg" that can act both as a flag and as a segment locator for high speed block move statements. My code usually contains something that looks like:

```
interrupt(0x11,&rr);
scrseg=((rr.ax>>4)&3==3)?0xb000:0xb800;
```

Of course, we have those delightful bit operators from C, plus the ternary operator. If nothing else, C is efficient.

If the video card provides color graphics, you can just use interrupt 10h, service 0, to set the mode to 4, 5, or 6, depending on your tastes and your monitor.

Hercules?

If screen memory is at b0000h, things get sticky. The results, however, are much better. Anyone who has had to put up with medium resolution graphics on a composite monitor has a revelation in store when he sees 720x348 graphics. But how is a program to know whether it has a graphics card? Try to send Hercules graphics data to an IBM monochrome (character only) card, and you get either garbage or nothing.

Figure 1 - Check For Hercules Card

```
#define STATUS 0x3ba

herc()        /** courtesy of Hercules Corp. */
{
    char test;
    unsigned i;

    test = inb(STATUS) & 0x80;
    for(i=0;i<0x8000;i++) if( (inb(STATUS) & 0x80) != test )  return 1;
            /* it's graphics!! */
    return 0;    /* no retrace bit change, no herc card */
}
```

Short of risking a lock up, the only way to find out whether you have a Hercules card is to get down to the hardware. The status port of the 6845 video-controller chip on the mono/ Hercules cards is at 0x3ba. It so happens that the Hercules Corporation sets the high byte of that port during video retrace in the text mode. The standard mono card doesn't. All you have to do is test the high byte over a period of time. If it changes, you have a graphics card. I put a "boolean" function in my program that returns TRUE if the card can show graphs — see Figure 1.

If this function returns a FALSE, then a polite "sorry no graphs" signal pops on screen when necessary.

The real problem is getting the Hercules card into the graphics mode. As I said, DOS won't help you. Neither will the documentation on many commercial cards, which simply offer patched BASIC routines. There are some things I will not do, even for beautiful graphics.

Fortunately, my Taiwanese clone-maker published the requisite port calls amid the broken English. All you have to do is send a herd of port calls to the 6845 video chip. See Figure 2.

Yeah, I know that that's sloppy, sloppy code. Unfortunately, the 6845 must be very unstable in transition, because when I set up the code to read an array of numbers with a nice loop, it apparently takes too long and the whole thing locks up. I'm not sure whether any other higher-level language will be fast enough to permit even this. Maybe one would have to go to in-line machine code with Turbo Pascal. I wouldn't even try it in BASIC.

Tinkering With The Bits

Now for some graphics. The first rule for speed and ease of operation is that screen memory is much easier to

(continued next page)

Figure 2 - Getting A Hercules Card In & Out Of Graphics Mode

```
#define INDEX 0x3b4    /* index register for herc adapter */
#define MODE 0x3b8    /* mode control adapter */
#define CONFIG 0x3bf  /* configuration switch */
#define TEXT 0x29     /* text setting for mode control */
  /* This could be 0x28; but 0x29 keeps mono card from self-destructing
       if you accidentally call the wrong video card */
h_tograph()
{
  outb(3,CONFIG);  /* set to full screen buffer=both pages activated */
  outb(CON | 2,MODE); /* set to graphics mode */
  crtout(0,0x35);
  crtout(1,0x2d);
  crtout(2,0x2e);
  crtout(3,0x7);
  crtout(4,0x5b);
  crtout(5,0x2);
  crtout(6,0x57);
  crtout(7,0x57);
  crtout(8,0x2);
  crtout(9,0x3);
  crtout(10,0x0);
  crtout(11,0x0);
  crtout(12,0x0);
  crtout(13,0x0);
  crtout(14,0x0);
  crtout(15,0x0);
  crtout(16,0x0);
  crtout(17,0x0);
}
/************/
h_fromgraph()
{
  outb(CON,MODE);     /* text mode */
  outb(0,CONFIG);     /* cancel paging */
  crtout(0,0x61);
  crtout(1,0x50);
  crtout(2,0x52);
  crtout(3,0xf);
  crtout(4,0x19);
  crtout(5,0x6);
  crtout(6,0x19);
  crtout(7,0x19);
  crtout(8,0x2);
  crtout(9,0xd);
  crtout(10,0xb);
  crtout(11,0xc);
  crtout(12,0x0);
  crtout(13,0x0);
  crtout(14,0x0);
  crtout(15,0x0);
  crtout(16,0x0);
  crtout(17,0x0);
}
/***************/
crtout(reg,val)
  char reg,val;
{
 outw(reg+0x100*val,INDEX);
```

(continued from page 69)

manipulate down in the cozy confines of the data segment of your C program than in the stratosphere.

Even if you own a "long" C that lets you point directly to the screen memory, bit-fiddling will go ever so much faster if work is done in a 64K "short" region of main memory. To go over an entire screen means a multitude of function calls, and long calls (that is to say, calls that change the code segment as well as the offset) take time.

As I've pointed out in previous columns, the high speed block transfer routines of the 8086 family are so fast that moving 16 or 32K of memory back and forth from buffer to high memory takes almost no time. Of course, this presupposes that you have 16 or 32K in the data segment to spare.

If only it were easy to relate buffer addresses to CRT coordinates. The color graphics card uses double interleaving, gaining no doubt some speed. But it complicates writing to the screen. In either medium or high resolution, the bytes representing every other line on the CRT are 0x2000 apart in memory. Not to be outdone, Hercules uses quadruple interleaving, with four "pages," each 0x2000 long.

Graphing Something

Let's draw a one-cycle sine wave (with X Y coordinates) and make it run on both the CGA (color graphics adapter) and Herc card. (See Figure 3.)

That horrid mess of shifts and and's and or's in the pixel() function is just a faster version of the div-ing and mod-ing forced upon us by the interleaving of graphics memory.

Shifting and bit arithmetic are usually faster than adding, subtracting, multiplying, and dividing when we're dealing with powers of two. In this case the speed gain is minimal, since floating point arithmetic is the real boat anchor.

I never claimed this was a tutorial in high-speed graphics. But if you can do your screen manipulations with integer operations, things speed up a lot.

■ ■ ■

Figure 3 - Draw Your Own Sine Wave

```
#include <regs.h> /* for interrupt calling */
#define pi 3.1416
unsigned scrseg;
double x_ratio,y_ratio;
struct regs rr;
char *cptr;
main()
{
 unsigned i,j,*intptr,x_count,y_count,mid_x,mid_y,scrsize;
 char *hold;

 interrupt(0x11,&rr);   /* use int 11h to find text mode */
 scrseg = ( (rr.ax >> 4) & 3 == 3) ? 0xb000 : 0xb800;
 if( (scrseg == 0xb000) && !herc() ) exit(); /* exit if no graphics */
 if(scrseg==0xb000){
   x_count=720;
   y_count=348;
   scrsize=0x8000;
   }
  else{
   x_count=640;
   y_count=200;
   scrsize=0x4000;
   }
  mid_x = x_count/2;       /* mid screen */
  mid_y = y_count/2;
  y_ratio= 2.6667/y_count;
  x_ratio = 2*pi/x_count; /* one cycle of sine */
  hold=malloc(0x8000);   /* allocate and clear buffer */
  memset(hold,0,0x8000);
  for(i=0,cptr=hold;i<x_count;i++){
    pixel(i,mid_y-sineconv(i-mid_x)); /* origin at mid screeen */
    pixel(i,mid_y);          /* draw in y-axis */
    }
  for(i=0;i<y_count;i++) pixel(mid_x,i);   /* draw in x-axis */
  if(scrseg==0xb000)  h_tograph(); else c_tograph(); /* to graph mode*/
  poke(scrseg,0,hold,scrsize);  /* send buffer to screen memory */
  getchar();
  if(scrseg==0xb000) h_fromgraph(); else c_fromgraph();
  clr('s');                 /* clear screen routine */
}
/*****************/
sineconv(x)
 int x;
{
  double sin();
  return  (int) (sin((x)*x_ratio)/y_ratio);
}     /* using a cast to truncate floating point */
/*****************/
pixel(x,y)
 int x,y;
{
 unsigned offset;
  offset=(scrseg==0xb000) ? 0x2000*(y&3) + 90*(y>>2) + (x>>3)
     : 0x2000*(y&1) + 80*(y>>1) + (x>>3);
 cptr[offset] |= 0x80 >> (x&7); /* cptr points to base of buffer */
}  /* or-ing pixel position with the graphics screen byte */
/****************/
c_tograph()
{
 rr.ax=6;       /* go to high resolution mode*/
 interrupt(0x10,&rr);
}
/********************/
c_fromgraph()
{
 rr.ax=3;         /* back to text mode */
 interrupt(0x10,&rr);
}
/********************/
 ....AND INCLUDE THE CODE FROM THE OTHER FIGURES
```

Path Searching
Archiver
8Mb Print Spooler
Aliases
Disk Variables
Screen Paging
Multiple Commands
Pipes
Memory Management

CP/M™ + ConIX™ =

I/O Redirection
Function Keys
Argument Handling
Virtual Disk
Overlay Searching
Memory Files
New SysCalls
Perfected User Areas

If you've tried some of the earlier CP/M-80 add-ons such as Microshell™ and ZCPR™, then you know it's possible to add new features to CP/M. Now you can have much more without all the trade-offs when you use ConIX, ***the latest and greatest upgrade for CP/M!***

"This system has all of its competitors beat."
Computer Language, June 1985.

"A superb product... ConIX is a bargain."
Micro/Systems Journal, May 1986.

ConIX Operating System: List ~~$49.95~~ **Low As $19.95!**

A complete upgrade for 48K+ CP/M 2.2/3.0 and equivalent systems. Provides professional capabilities with blinding speed, as often found on more expensive MS-DOS™ and UNIX™ machines. Installs easily in 1 minute to add over 100 new commands and features. Eliminates many points of user frustration with CP/M. Uses only 1/2K TPA, 0-27K disk minimum.

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 Pull-Down Menu: List ~~$34.95~~ **FREE OFFER!**

A user-friendly interface to ConIX that is fully menu-driven, with helpful prompts and easily understood explanations of all menu functions. Loads with a single keystroke at the prompt level. Source code included*!*

ConIX Programming System: List ~~$49.95~~ **Now Only $29.95!**

A full programming language for ConIX extends CP/M SUBMIT capability. Features conditionals, loops, subroutines, labels, nesting, interrupt processing, error traps, and debugging facilities. Design intricate menu systems. Includes a special "compiler" that provides string and numeric variables, an integer math package, a relocatable assembly code generator, and much more. A *must* for CP/M power-users and developers*!*

ConIX Library I - XCC Utilities: List ~~$49.95~~ **Low As $10.00!**

Over 25 utilities written in the ConIX 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*!* Does not require Programming System.

Special Distribution Versions of the ConIX Operating System

NEW! ConIX Shareware Version:

Introducing a new Shareware version of our best seller, the ConIX Operating System. Included is 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. Users who wish to register their copy pay regular advertised prices and receive the complete software package as well as full-service customer support with upgrade notifications.

NEW! ConIX Disk Manual Version:

To reduce the cost for those who want to purchase only the ConIX Operating System, we are offering the complete software package without the printed manual. Instead, complete documentation is provided on disk with each chapter stored in individual files.

The disk manual does not include the Chapter Summary, Chapter Reference, and Index sections that come standard in our regular typeset manuals. Copies of the printed manual are available separately should you wish to obtain one at a later date.

ConIX™ Liberty Centennial FREE-Order Give-Away, New Price Reductions, Specials, and Free Offers

CHI is celebrating the 100th birthday of one of our nation's most beloved landmarks, the *Statue of Liberty*, with our *Free-Order Give-Away*. We're simply saying thank you for being our customer by giving you a chance to receive your order absolutely *FREE*!

Every 100th order processed by our computer will be shipped with a Credit Certificate for the total purchase price or $100.00, which ever is lower. This credit may be used towards payment of a future order from CHI, or may be redeemed within ninety (90) days of receipt for a check for the full credit amount. Your odds of receiving a free order are an incredible *1 in 100!* To apply, certify eligibility by enclosing the completed and signed form below with your order. (Photocopies accepted.)

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. This offer fully replaces Sweepstakes previously advertised.

FREE! Pull-Down Menu System:

As a special offer, the ConIX Pull-Down Menu System is included *FREE* with all Operating System purchases. That's a savings of $24.95 off the regular list price!

SPECIAL! Buy 'Em All Discount Offer:

As a special incentive, when you buy all three ConIX packages, you get the Library for only $10. That's right, for $69.95 you'll get the ConIX Operating System, Programming System, Library I, and three instructional manuals. In addition, we'll throw in the Pull-Down Menu at no extra cost. Act now and secure your copy of the complete line of ConIX products at our lowest price, ever!

FREE-Order Give-Away

Name: ____________________

Address: ____________________

City: __________ State: ______ Zip: ______

Dear Computer Helper Industries:

Please enter my name in your Give-Away offer. I understand that every 100th qualified order will be shipped with a Credit Certificate for the full price of the order or $100.00, which ever is lower, and I will have up to ninety (90) days to exchange the credit for a check in full. I also understand that I am only eligible to enter if the order is for private, non-commercial, or non-profit institution use.

Signature: ____________________

Void where prohibited by law. Offer may be cancelled without prior notice.

DT: O#: C#: PP$:

Name: ____________________

Company (If any): ____________________

Address: ____________________

City, State, Zip: ____________________

Tel. #, Hours: ____________________

__	**ConIX Operating System:**	$ 29.95	$ ______
__	Disk Manual Version	$ 19.95	$ ______
__	Limited Shareware Version	$ 0.00	$ ______
__	Printed Manual Only	$ 9.95	$ ______
__	**ConIX Pull-Down Menu System**	$ 0.00	$ ______
__	**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: ____________________

Disk Format Information: *

__	8" SSSD Standard	$ 5.00	$ ______
__	5¼" DSDD 48 TPI Soft Sector	$ 5.00	$ ______
__	5¼" SSDD 48 TPI Soft Sector	$ 6.00	$ ______
__	5¼" __S__D __TPI ____ Sector	$ 10.00	$ ______

* Add one format charge for each package ordered (complete system counts as one).

Format Brand: ____________________

Shipping Information:

☐	UPS Ground USA	$ 4.50	$ ______
☐	Air Mail Canada	$ 9.50	$ ______
☐	Air Mail Foreign	$ 12.50	$ ______

Subtotal: $ ______

Tax (N.Y. Residents Only): $ ______

Total (Thank You!): $ ______

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 dollars.

Computer Helper Industries Inc.
Post Office Box 680
Parkchester Station, N.Y. 10462
(212) 652-1786 9AM-5PM M-F

256K Upgrade For 84 Kaypros: A Mod From The Ether

By Charles McHan
3346 Royal Palm Dr
Jacksonville, FL 32250
904-223-2551

Bulletin boards are wonderful collectors of stray radiations from the ether (namely software). After publishing the 256K upgrade for 83 Kaypros we had a bunch of requests for the same project on the 84 models. And what should appear on the board one day? The following article.

The Software

You'll need KAY256.LBR (contains all files listed below and is available on our disk K-47 for 8 bucks or on the Micro C BBS.) If you have Microsoft's M80/L80 assembler/linker package, now's the time to configure RAMDRIVE.MAC for your system. For those without M80, R2.COM (RAMDRIVE for Kaypro 2s) and R4-10.COM (for Kaypro 4s and 10s) work well with both the standard Kaypro ROM and our 884 series ROMs. The ZCPR1 in ROM feature of the 884 MAX ROM does not work properly with RAMDRIVE. (The RAMDRIVE works fine if you boot your MAX system with a non-ZCPR disk.)

The Hardware

Follow Dr. Liddle's procedures for removing the old RAM chips (see the Kaypro column in issue #30). Install sockets, if necessary, and wire together pin 1 of each socket. With the old RAM chips reinstalled, check the machine thoroughly using MT256.COM. It tests each location for every possible read/write value. An error will appear if any location tests bad.

Build the daughter board as shown in Figure 1. Please note the decoupling capacitors added between +5 V and

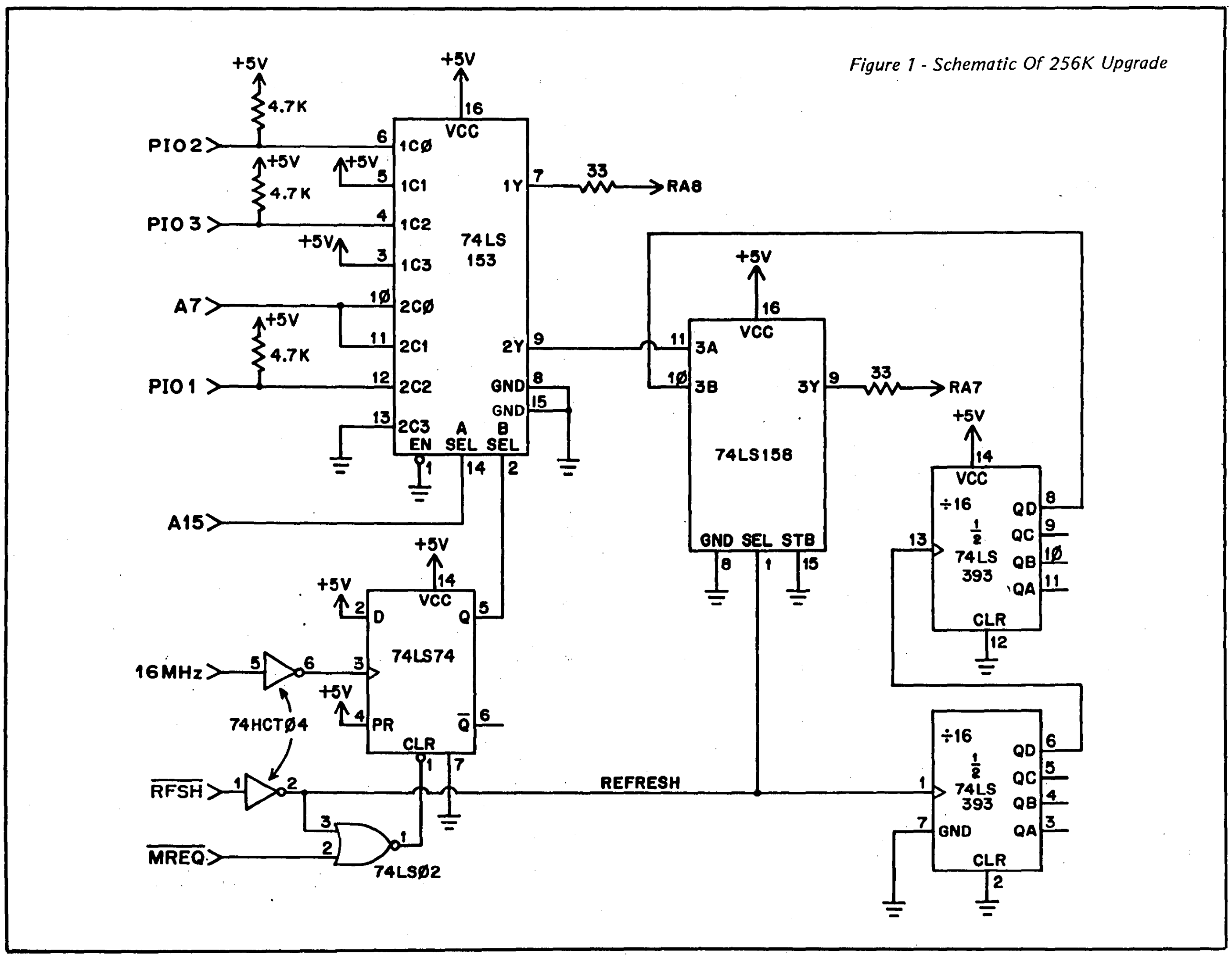


Figure 1 - Schematic Of 256K Upgrade

ground on each chip. Mount the daughter behind the monitor ROM keeping all wires as short as possible. Don't forget to bend out pin 24 of U29 (just enough so that it doesn't go back into the socket).

Spare Socket Method

If you have a 4-84 or 10-84 with the internal modem and real time clock, skip to the piggyback mod. Otherwise install a new Z80 PIO in the empty socket at position U35.

Tie PIO1, PIO2, and PIO3 from the daughter board to pins 27, 28, and 29 of U35, respectively. If U27 (74LS138) is missing, install it also. Your data and control ports for RAMDISK.MAC will be 021H and 023H. Set these equates near the end of RAMDRIVE.MAC. Or use the already configured R4-10.COM.

Piggyback Method

For 4-84s and 10-84s with a PIO already installed in U35, you must piggyback a new PIO carefully on top of the existing one as follows:

Solder the new PIO's pin 1 to the lower PIO's pin 1. Then, new PIO's pin 2 to old PIO's pin 2 and so on for pins 3, 5, 6, 11, 19, 20, 22 through 26, and 34 through 40. (Work carefully!).

On the new PIO, bend out the remaining pins slightly so they don't touch anything. Now you have two PIOs stacked in orbiter/747 fashion.

Attach new PIO pin 4 to U27 pin 13. This designates the new PIO as ports 029H and 02BH for data and control. These are the values you will use for the PIO equates in RAMDRIVE.MAC. R2.COM is the RAMDRIVE version configured for you piggybackers.

Tie PIO1, PIO2, and PIO3 from the overlay to pins 27, 28, and 29 of the new PIO, respectively. As an added bonus, you end up with a spare side for another parallel port if you like. (Z80 PIOs have two sides and we've only used one.) (Its best side.)

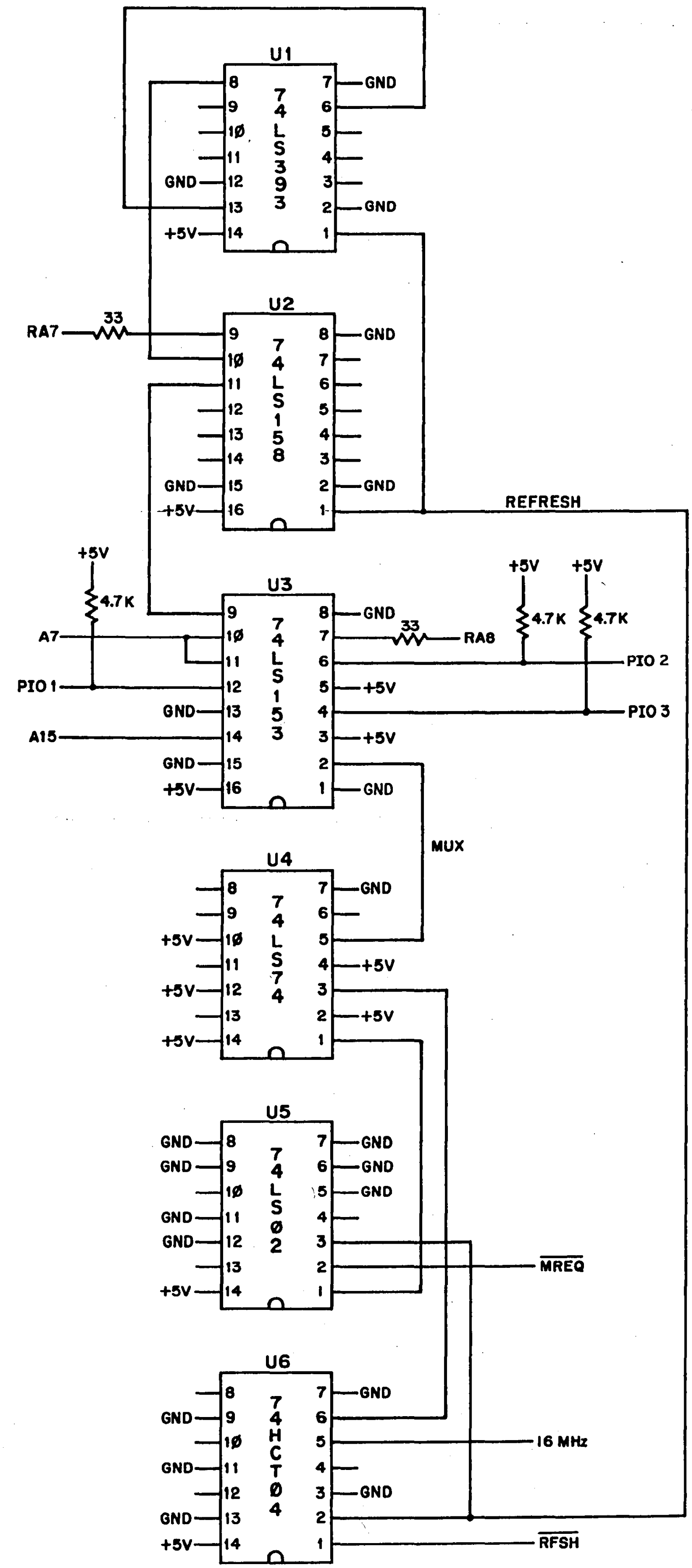
Install The RAM

Now install new 256K chips in the RAM sockets. Finish wiring the overlay, keeping all wires, ESPECIALLY 16 MHZ, as short as possible. Try to keep them separated too.

Editor's Note: "Finish wiring the over-

(continued next page)

Figure 2 - Wiring Diagram For 256K Upgrade



KAYPRO

(continued from page 75)

lay..." is an oversimplification. This is not a trivial mod. We used a small prototype board from Radio Shack (part number 276-162) as the base for the daughter board.

There's a fair amount of wire-wrap wire under the board. By playing with the length and position of these wires we were able to reduce crosstalk and other noise enough to make the RAM work. If you don't have access to a decent scope (50 MHz+), you should think three or four times before attempting this mod.

This is significantly more involved than the Kaypro speedups or 2 - 4 mods.

Gentlemen, Start Your Kaypros

Turn on your machine. It should boot and perform properly. Test disk I/O, printer function, and anything else you can think of. Check the memory again with MT256. When it looks like everything is behaving itself, run RAMDRIVE. You should now be able to log onto the RAMDRIVE. If you use R2 or R4-10 the ramdisk will be E:.

What's Going On

The theory is simple: We need the MUX and RFSH signals that the 84s lack. U6 buffers the 16 MHz clock signal and inverts 1/RFSH from the Z80. U5 provides a pulse from 1/MREQ and RFSH that allows U4 to generate the missing MUX signal. I originally used side 2 of U4 to generate a 1/CAS signal to replace the one from U29, but that wasn't necessary.

In Conclusion

I'm running the RAMDRIVE on a Kaypro 4-84 that's fully loaded. My BBS uses both an external Hayes and the internal modem in a multi-user configuration, a Kaypro clock, and a Shugart 712 (stretched to 13 meg) hard disk with HDO controller. It's been online for about 12 hours as I write this, with no flaws evident. I managed to cut HD access by 90% by placing my menus on RAMDISK!

Any comments or suggestions would be appreciated. Leave them on GEnie (CWMCHAN) or call AMY < > BBS at 904-725-7461 (300/1200 baud) or 904-725-1226 (300 baud). This is kinda rushed, but I hope it helps!

■ ■ ■

LETTERS

(continued from page 5)

it proved impossible. No representative would take my calls, or return them. After more than a week of this, I wrote Express a certified letter demanding action on the returned unit and contacted the local Better Business Bureau.

As a result of one or both of these actions, Express returned the original unit — in its original failed condition — without any explanation or justification. The BBB, on the other hand, has received no response to their inquiries, and has closed the complaint accordingly.

Bob G. Roberts
11860 E. Fair Oak Ave.
Baton Rouge, LA 70815

Tech Help

I wrote to PROFILES, but know it will take forever to get an answer to a simple problem you guys must know off the tops of your heads.

I've graduated from canned programs to attempting my own in assembly language. Carefully studying the obfuscating CP/M operating system manual which came with my Kaypro IV '84, nowhere can I find the most fundamental and essential information — the port addresses! Could you please tell me where one may find the port addresses for keyboard, CRT, printer, and status port (with which bits are assigned to which device)?

Are you aware of any useful Z80 assembler text for beginners or intermediate users?

Edmund B. Lewis
1229 E. 32 St.
Brooklyn, NY 11210

Editor's note:

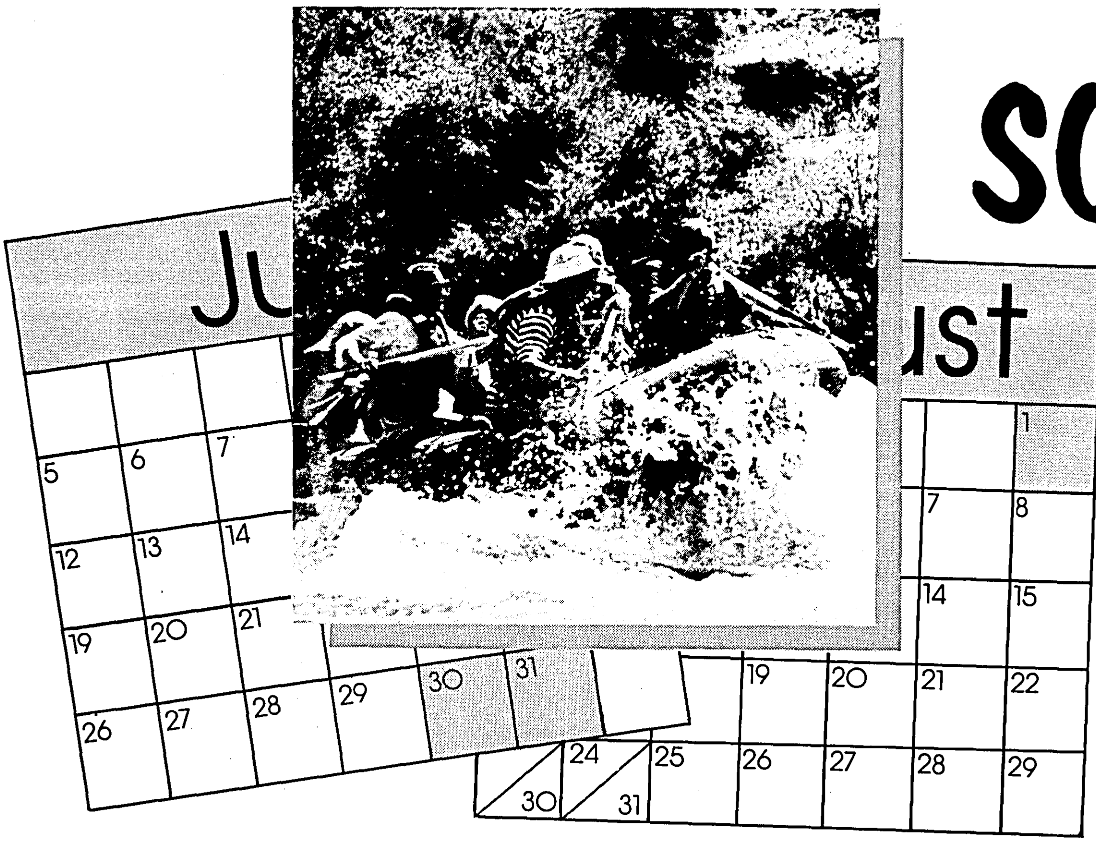
All the port information (and more) is in our schematic and theory of operation package for the 84s ($20). The keyboard port is 07h for control bytes and 05h for data. Parallel printer port is 14h (bit 3 strobe, bit 6 ready) for control and 18h for data. Serial printer port is 0Eh for control and 0Ch for data.

Programming the 6845 video controller is not trivial (there's sample code in the theory of op). Anyway, the ports are: video control 1Ch, video attributes 1Dh, and video memory 1Fh.

If you want to program the SIO or PIO, get a data book from Zilog or check out the Z80 books at B Dalton's (they should still have a few).

Inside CP/M *and* Soul of CP/M *are good books for futzing around inside CP/M systems. We have* Inside CP/M. *B Dalton's should have* Soul of CP/M.

■ ■ ■



SOG VI is coming!

Mark Your Calendar For The World's Wettest Conference

Recharge your technical batteries and your recreational batteries all at once. SOG VI (Semi-Official Get-together VI) will be a technical event like no other. You'll have a chance to practice technical white water rafting, technical hiking, technical horse back riding, and technical shoulder rubbing.

You'll have a chance to rub shoulders with the best technical people in the industry. (They're coming to rub shoulders with you.)

Technically speaking, there'll be technical forums, technical lectures, technical sightseeing (both terrestrial and extra-terrestrial), and just plain technical companionship.

It's even cheap, technically. The dorm is not very fancy, but it's very inexpensive. We don't have the exact prices this year but it's probably not going to be over $90 for a two-person room for four nights (Wednesday night through Saturday night, checkout Sunday p.m.).

Or, if you're even cheaper: Bring your motor home, trailer, or van, and sleep in the parking lot (a quiet spot nestled right up against the dorm.).

Or, you can stay off campus. We have motels (lots and lots of motels) as well as campgrounds and trailer parks.

Details on this and more in this year's SOG packet. Write or call for yours tomorrow (or better yet, today). It'll have particulars on the dorms, motels, campgrounds, and trailer parks. (The dorm filled on May 1st last year, so if you've got your mouth set on a room, get your reservation and money in early).

Note that PSA is now providing jet service between San Francisco and Redmond. Redmond is just 15 miles away.

Price: You pay for your own food, lodging, and the guide's fee for rafting, etc. (see the SOG packet for details), but the conference is free. (This year you can make a $25 donation to help pay for use of the college, but it's optional.)

Thinking of speaking? Contact Larry Fogg or Dave Thompson here at Micro C (503-382-5060) or via the bulletin board (503-382-7643). As a speaker, you get a free 2½ hour raft trip, kickoff dinner, and a special speaker's T-shirt. (Fame and a good time, how can you beat that combination?)

Micro Cornucopia | P.O. Box 223
Bend, OR 97709

By John P. Jones
6245 Columbia Ave.
St. Louis, MO 63139
(314)645-1596

The Tale Of Two Modulas

John takes a good look at Pascal's offspring, Modula-2. In this column he also compares two versions of the language — Logitech's Modula-2/86 for MS-DOS and FTL's Modula-2 for both MS-DOS and CP/M.

Although Pascal in its many implementations has evolved into a capable application development language, there are situations where it isn't the best choice. Niklaus Wirth, the inventor of Pascal, has combined the best features of Pascal with those of Modula (a language he developed for multiprogrammer environments) to develop a new language, Modula-2. Since some of its roots are in Pascal, Modula-2 has many similarities to that language. Where there were deficiencies and inconsistencies in Pascal, they have been corrected. The differences between the two languages contribute both to the greater flexibility of Modula-2 and to Modula-2's superiority for development of larger applications.

Modula-2 was created as a single programming language suitable for programs ranging from operating systems down to low level device drivers and up to applications for business or engineering. At the same time, Niklaus retained portability and machine independence. All this flexibility is made possible by the language's use of modules.

Modula-2 is a 'sparse' language; the language itself has NO input or output statements, math functions, string manipulation functions, or the like. If these are needed they must be IMPORTed from the appropriate library module. See Figure 1 for a list of the reserved words and pre-defined identifiers in Modula-2.

By confining machine specific functions to a few modules, Niklaus has made it relative easy to move the language (or a program) onto another computer. One such module, SYSTEM, is required by the language definition for some low level identifiers.

Compare And Contrast...

Modula-2, like Pascal, is a block structured language but its block boundaries are more precisely defined. All blocks begin and end with a keyword. In addition, Modula-2 is more strongly typed than Pascal.

Modules

A Modula-2 program is a MODULE. To provide for separate compilation of libraries, there are two other types of module: DEFINITION and IMPLEMENTATION. The definition module serves as a prefix to the implementation module, and also defines what will be visible outside the module. In the third edition of N. Wirth's book (*Programming in Modula-2*, Springer-Verlag) the requirement for the explicit EXPORT of identifiers from a definition module was dropped. If the identifier needs to be hidden, it's defined in the implementation module.

The implementation module contains the code for the identifiers in the definition module, as well as any definitions that are to be hidden. Three important points need to be made about this separation of definition and implementation.

First, it is only if the definition module is recompiled that any modules which import it need to be recompiled; the implementation can be changed and its importers need only be relinked.

Second, more than one implementation module can exist for a single definition. (A good example of this is Logitech's Modula-2/86 which has separate implementations for 8087 emula-

Figure 1 - Reserved Words & Pre-defined Identifiers In Modula-2

```
A) Reserved words

   AND        ARRAY           BEGIN       BY
   CASE       CONST           DEFINITION  DIV
   DO         ELSE            ELSIF       END
   EXIT       EXPORT          FOR         FROM
   IF         IMPLEMENTATION  IMPORT      IN
   LOOP       MOD             MODULE      NOT
   OF         OR              POINTER     PROCEDURE
   QUALIFIED  RECORD          REPEAT      RETURN
   SET        THEN            TO          TYPE
   UNTIL      VAR             WHILE       WITH

B) Pre-Defined Identifiers

   BITSET   BOOLEAN   CARDINAL  CHAR  INTEGER
   PROC     REAL      NIL       TRUE  FALSE

C) Pre-Defined PROCEDUREs - built in to compiler

   ABS   CAP   CHR   DEC    DISPOSE  EXCL  FLOAT
   HALT  HIGH  INC   INCL   MAX      MIN   NEW
   ODD   ORD   SIZE  TRUNC  VAL

   SIZE, NEW and DISPOSE may require IMPORTation of
    other identifiers or procedures.
   MIN and MAX new with third edition
```

tion and native coprocessor code.)

Finally, large projects can be split among several programmers. If all have access to the definition modules, the implementation details (assuming they work) are not important.

A module IMPORTs identifiers from other modules in order to use them. Either entire modules or specific identifiers can be imported:

FROM Terminal IMPORT WriteLn, WriteString;
IMPORT TkernelIO;

Identifiers

Identifiers are constructed as in Pascal, but with some important differences. Modula-2 identifiers are case sensitive. Variable, variaBle, and variablE are different. For me, this is one of the more difficult things to get used to; I'm constantly misspelling standard and library module identifiers. Only letters and digits are valid in identifiers; the convention is to capitalize the first letter of each word in an identifier. (FirstPointerVariable, for example.) Reserved words and predefined identifiers (Figure 1) are always all caps.

Data Types

In Figure 1 you will see that all the familiar Pascal data types are available, with a few additions. A CARDINAL is not a bird, but an unsigned integer, and can take values in the range 0..65535. Operations on cardinals are a bit faster than on integers, so they are preferred when negative values are not possible. Cardinals are assignment compatible with integers, but not expression compatible. The type BITSET is the set of values which can be represented by the bits in a single machine word, and is somewhat machine dependent. For a 16 bit word, it is the set [0..15]. Operations on bitsets are fast.

The type PROC is the predefined type which is compatible with a parameterless procedure. This indicates that you can declare PROCEDURE types. If a procedure can have a type, it can be used as a parameter for another procedure. There are some restrictions on this; only procedures declared at the outermost level of a module can be used as parameters, and pre-defined procedures cannot be used as parameters. Procedure types are compatible if their parameter lists are compatible. Using procedures as parameters is a very powerful tool, but both incomprehensible and unworkable code can result.

Like Pascal, Modula-2 allows various user defined data types, including RECORDs (which can have multiple variant parts), ARRAYs, SETs (restricted in the base language), and subranges. It also supports user defined enumerated types.

Unique to Modula-2 are imported types. A data type imported from another module can be either transparent (structure known) or opaque (structure unknown). For most compilers, an opaque type must be compatible with a POINTER. Since only assignment and allocation of space are allowed with an opaque type, the exporting module must also export all procedures needed to manipulate that type.

Modula-2 allows type breaking to circumvent its built in strong typing. The result is the responsibility of the programmer since no data conversion is done. The data is just acted on as if it were a different type.

PROCEDUREs

PROCEDUREs in Modula-2 are declared as in Pascal. Modula-2 has no function declaration, instead a procedure can RETURN a value. Significant differences from Pascal: a procedure declaration must end with its identifier appended to the final END statement, a function procedure must exit with a RETURN statement, and a parameterless function procedure must be both declared and called with an empty parameter list.

Formal procedure parameters are similar to Pascal, with both VAR and value parameters supported. Modula-2 also allows open array parameters. An ARRAY parameter declaration without an index subrange is used for this. The actual parameter will be mapped to an index range from 0 to one less than the number of elements in the array. The built in function HIGH returns the actual array's upper bound. (See Figure 2.)

Concurrent processing

Modula-2 provides facilities for concurrent processing. This allows activation/inactivation of procedures based on the status of other procedures or events. I'll defer this topic to another time.

Standard Modula-2?

As with any new language, Modula-2 is still evolving. Unfortunately, this leads to a number of different 'stan-

(continued next page)

Figure 2 - The Use of Open Array Parameters

```
PROCEDURE NotOpenArray ( x: ARRAY [0..11] OF MonthLength);
PROCEDURE OpenArray ( x: ARRAY OF MonthLength);
.
.
  FOR i := 0 TO HIGH(x) DO
.
.
```

(continued from page 79)

dards' for the language.

Although you can expect certain functions in the standard library, neither their names nor their contents have been standardized. Keep this in mind as I look briefly at two Modula-2 development systems for MS-DOS.

Logitech's Modula-2/86

Logitech Modula-2/86 conforms to the language definition in the second edition of Wirth's book (1982). The library modules have been derived from those for the Lilith (a minicomputer designed specifically to run Modula-2).

I bought the base language system, which includes the editor, compiler, linker, a set of standard and utility modules, and a utility for conversion of linker output files to standard .EXE format. Modula-2/86 uses a large memory model: programs have access to the 640K of user memory which MS-DOS supports.

The syntax assisted editor is loosely coupled to the compiler. Although you can compile (and link) from within the editor, the compiler MUST read the source from disk, and always writes its error listing (for the editor) to disk.

The loose coupling means that the compiler can also be used in batch mode, independent of the editor. The editor supports windows with cut and paste, keyboard macros (somewhat inconveniently), and can be customized. It is a bit slower than I'm used to, but performs well when creating Modula-2 source code.

It's a four-pass compiler, each pass in the base language system is loaded from disk. It's a lot slower than the one-pass compilers I'm used to; a 500-line program (with imports from 5 library modules) takes 1 min. 15 sec. to compile (8 MHz V20 Clone with 20 Meg winchester).

The linker is quick, the same module took about 16 sec. to link. The final step, conversion to an .EXE file (41K), took 13.5 sec.

The libraries which are supplied support file and terminal I/O, string manipulation, processes, interrupts, and the like. Given the philosophy of the language, if you don't like a library module, re-write it! Printed source for the library definition modules is in the manual.

I'd also like to have the sources on disk; it's very handy to open an editor window on a definition module while you're editing the module that imports it.

The manual is large, even impressive in size, until you realize that more than a third of it is either definition module source or devoted to utilities not provided with the base language system. If you're comfortable with Pascal, the tutorial on Modula-2 for Pascal programmers may be as much as you need to get started.

There are examples of Modula-2 code in the manual, but more would be better. The text index is much too brief, but the index of procedures in the library modules is helpful. You'll probably want another book to supplement the manual.

Logitech offers several options and additions to the base language system. I haven't used all of these, only the 8087 support and Turbo to Modula-2 source translator.

1. 8087 Coprocessor support ($40 additional) — Recommended for floating point intensive applications.
2. Fully linked compiler ($100 additional) — Reduces compile time. (Requires 512K.)
3. Turbo Pascal to Modula-2/86 source code translator ($49) — This gem is not perfect, but does a remarkable job. The manual is excellent.
4. Run time debugger ($69) — Source level trace, breakpoints, etc.
5. Utilities ($49) — Post mortem debug, disassembler, cross reference.
6. Library module source ($99) — Could probably learn a lot by getting this one.
7. Window package ($49) — Teach your application to do windows.
8. Make utility ($29) — Selects modules to recompile when definition modules are changed.
9. ROM package ($199)

FTL Modula-2

FTL Modula-2 is largely the work of Dave Moore of Cerenkof Computing, Brisbane, Australia, and is available in this country from Workman & Associates. It is available for both CP/M-80 and MS-DOS. The FTL base system includes a source editor, compiler, linker, assembler, and library and utility modules. The system conforms to the language definition in the third edition of Wirth's book, so is more current than Logitech's system. FTL Modula-2 uses a small memory model: you are limited to 64K code and 64K data. (There is a library module, LongHeap, which allows use of the remainder of RAM for indirect data storage.)

The editor (which uses WordStar commands) supports windows, but is not syntax assisted. Compiles and links can be done either from within the editor, or from the command line.

One unique feature — when compiling from the editor, the compiler and imported symbols are kept in memory and do not have to be reloaded from disk for subsequent compiles.

The one-pass compiler is fast the first time; darned fast on subsequent times. Keyboard macros can be defined on the fly, while editing. Plus, you can examine the disk directory from the editor: more than once I've had to exit Logitech's editor because I'd forgotten a file name.

Although you have the option of continuing a compile after an error, I've found that it's usually best to fix the errors as encountered. Because it's a one-pass compiler, it can get very confused after an error. My 500-line module (tested earlier with Modula-2/86) compiled in 8.6 seconds.

The linker links directly to a .COM file, no additional conversion is needed. The test file linked in 22 seconds. The .COM files are consistently about 1/3 the size of Modula-2/86's .EXE files.

The standard libraries with FTL Modula-2 were taken from the CP/M-80 version, which in turn were taken from Modula-2 for the PDP-11. In many cases, they include both CP/M equivalent and improved MS-DOS modules. The CP/M style modules give you a quick migration route between the two operating systems. I've found the library source modules very instructive.

It's especially easy to incorporate assembly language modules. The definition module is written in Modula-2, the implementation module in assembler. The linker takes care of the rest.

FTL Utilities

Precedence generates a list of interdependencies. This list is used by Buildsub to create a batch file for recompiling modules affected by a definition module change.

Also included: a simple run-time debugger, a sort module, text and binary file comparison modules, and a source file lister.

Manual

The FTL manual set includes a general manual (originally for the Z80 version) and an MS-DOS supplement. There's a very good index, and it's full of tips and suggestions. I enjoyed Dave Moore's writing style. The manuals are designed to be used along with the library source. It may be significant that whenever I've had a general question on Modula-2, I go to the FTL manuals first, then the Logitech.

The only option available for FTL Modula-2 is the editor source, for an additional $30.

Comparisons

I've run a few benchmarks on both compilers — integer math, floating point math, file I/O, and text processing. For all but the integer math, Logitech's code has been consistently faster. FTL had the edge for integer math. FTL was especially poor at floating point division; I don't understand why it takes almost four times as long to divide as to multiply.

I don't really want to recommend one product over the other, but if you're planning a LARGE project, you'll need Logitech's large memory capability. On the other hand, for a ROM application, the tighter (but slower) code and easy assembler interface of FTL would probably be best. I've had both systems crash on me occasionally but that's not unusual.

I'm distressed at the marked differences between the libraries. I hope that some sort of standard emerges so programs can be portable.

I realize I haven't done justice to either Modula-2 as a language or to the two compilers in this brief space. It's been some time since I've talked about the basics of Pascal, and I'd like to go back over them in future issues. At the same time, I'll be covering Modula-2, and how to write the same basics in this new language.

Sources

Logitech Inc.
805 Veterans Blvd.
Redwood, CA 94063

Workman & Associates
1925 East Mountain St.
Pasadena, CA 91104

■ ■ ■

By Dave Hardy
736 Notre Dame
Grosse Pointe, MI 48230

Cold Storage

Dave puts his S-100 system on a diet and cuts its weight. Join us as he installs a new, low-power winnie.

One of the great things about most S-100 hard disk subsystems is the money you save on heat for your computer room. Unfortunately, the same savings don't apply during the summer, and the electric bills from running many of the old 8 and 14-inch winchesters can be astronomically high, especially if your machine runs all day.

I have two 20 meg Corvus Constellations and a 26 meg Morrow Designs (14" winchester!) running continuously. After paying nearly $100 per month just to run the drives for the last few years, I finally decided that there had to be a better way.

New Drives

New (that is, since about 1980) winchesters are much smaller, I thought, and must be cheaper to run than the iron monsters of the '70s. And because they're so common in IBMs and clones, they must be cheap and easy to interface.

So, I ripped a 20 meg drive out of one of the PCs at the office (the only thing they really used it for was to play SNIPES anyway) and set about connecting it to an S-100.

Like most PC hard drives, this one (a Rodime half-height) was ST-506 compatible, which would make interfacing it to an S-100 machine a simple task, I thought. It wasn't. After playing with several different circuits on paper, I finally decided that a drive controller board would be too complex to wire-wrap in just a few hours. Plus, either the hardware or the software would have to be so complex that my iron monsters began to look good to me again.

To make a long story short, after looking around the S-100 hard drive market for a while, I found exactly what I needed — the Western Digital 1002 winchester controller. A single circuit board, about 6" by 8", and packed with WD's 1000 series winchester controller series ICs, the 1002 would handle all of the drive interfacing chores with a simple command set, and talk to the S-100 bus via a few simple parallel I/O ports. In addition, the WD 1002 will run up to three ST-506-compatible drives at the same time, more than enough for most applications.

Back to the drawing board, and in about an hour I had put together a simple (but complete) interface for the 1002 that should work in just about any S-100, even non-IEEE-696 Imsais, like mine. Twenty feet of wire-wrap wire later, I had the interface board plugged into the S-100 frame, and was talking to the 1002 and, a few minutes later, the 20 meg winchester. (Writing the BIOS routines to interface the new hard drive to my CP/M took slightly longer, but this is a hardware column....)

Now for the good news: The whole thing cost less than $500 to do. In fact, if I'd used a "bargain" 10 meg drive (about $100), it would have cost only about $300, which WAS about the cost of three months' electric service for the old winchesters. The new drive consumes only about 100 watts, which is about $8 per month.

The Circuit

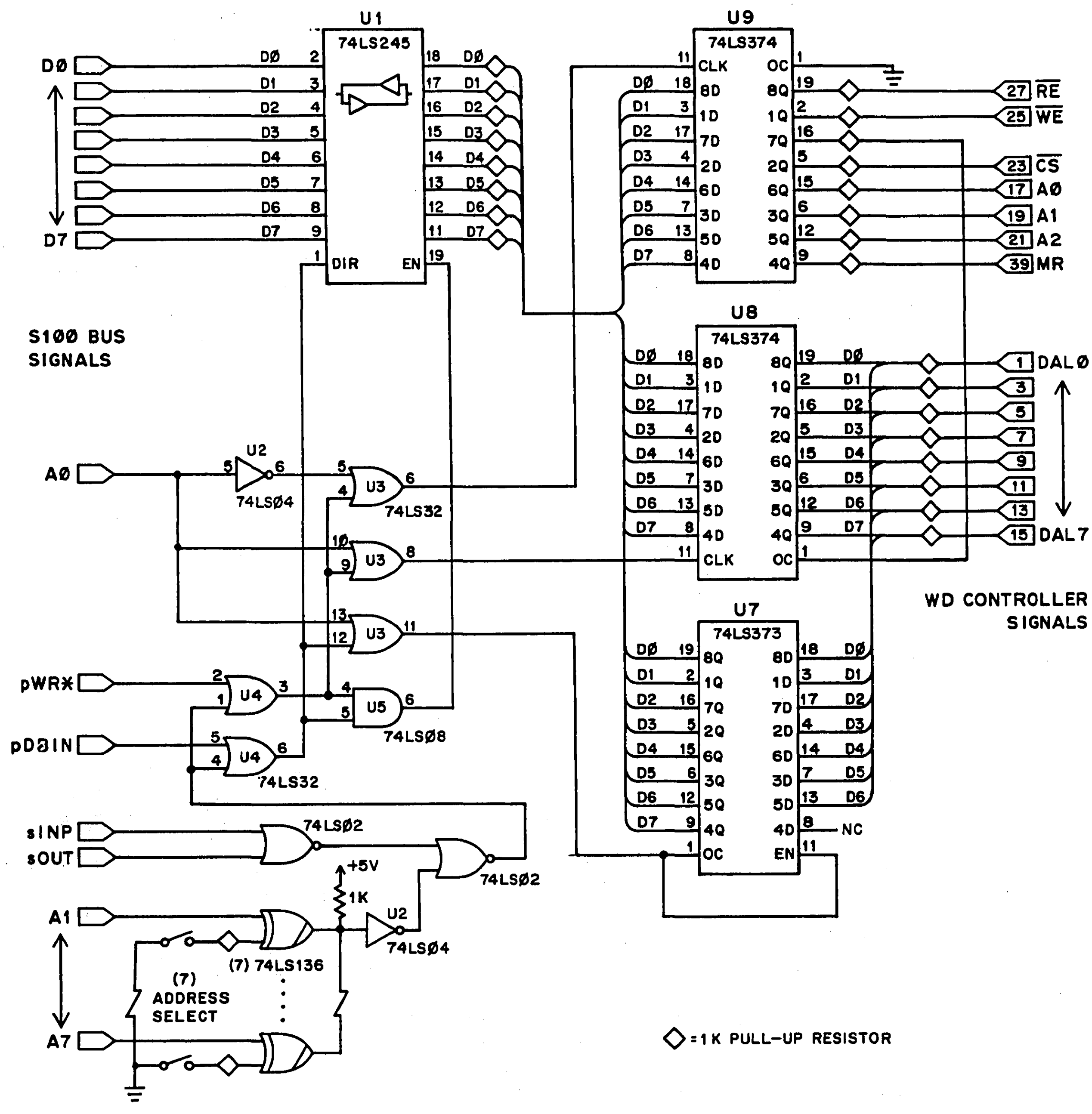
Figure 1 is the WD 1002 interface circuit. It uses only two port addresses, which can be set with the switches shown. Both commands and data are transferred via the data lines, and data transfers are handled a byte at a time, which makes it easy (although slower than DMA or TMA) to use in virtually ANY S-100 environment.

The WD card is easy to program, and the WD manuals give enough examples that you can write both a FORMAT program and functional BIOS routines in just a few hours. As you can see from the drawing, the WD 1002 uses only a few simple control lines — including select, read and write, error, reset, and acknowledge. Software drivers for CP/M systems are already available for this board on many RCP/M systems (including mine), although they'll usually require a bit of modification.

One thing you might want to add to this circuit is the ability for it to clamp the reset line during power down (many manufacturers do), although I haven't had any problems with this circuit as-is, possibly because I always park the heads of the drive before I power the system down. As always, of course, you should park the drive heads before moving the drive.

■ ■ ■

Figure 1 - WD 1002 Interface Circuit



(continued from page 2)

And chances are pretty good that the copies he has weren't copy protected. (If they were protected, then the protection's been broken.) So he's already comfortable with one or more. And, he's going to have to support whatever he recommends. What gets chosen?

Of course, his company could just "borrow" his copy and there'd be no sale. I'm sure that happens. However, all the companies I know who've borrowed programs have purchased copies of the ones they've decided to use.

Copy Protector

One copy protection company (they sell a serial port add-on called the BLOCK) states in its latest ad:

"In reality, the only people who could object (to our copy protection scheme) are those who would like the option of stealing your company's product."

On the other hand, a little temporary theft is often the best advertising. (Meanwhile numerous companies are advertising that they've removed block protection.)

Software Publisher

I got a call last year from a software publisher asking what I thought of copy protection. It turned out that his contacts with customers had become strained since he'd begun the protection. Little problems were being blown out of proportion and he was no longer getting as many new customers by word of mouth.

"Sure, continue the protection," I told him. "That's exactly the kind of situation that encourages new competitors."

Two months later he dropped the protection.

The Hackers' Home Companion

The other day saw several of us just sitting around, our faces glazed from the morning's tech calls. In the midst of the small talk (short words) the subject of identity came up.

"Who are we?"

"Well, uh."

"Hackers."

"You can't say that."

"Hackers."

"I mean, you can't say it out loud. In the magazine."

"Hackers!!"

"They won't understand — with all those teenagers breaking into the Pentagon's main computer. Micro C would be rated PG-13."

"Sounds exciting. I wonder if the Pentagon still keeps secrets in its computer."

"Hey, don't even breath that. You're really a ... uh ..."

"Mild mannered hacker."

"And a futzer, a disassembler, a modifier, an expert system creator, a leader of mortals and computer freaks, a wearer of the red cape."

"Yep, a hacker."

Bad Rap

The word has gotten a bad rap from the decimal majority.

A hex on them. We're taking the word back. It's part of our heritage, just like Dr Dobbs, Motorola, and George Morrow.

"Hacker" is a rich word, bringing back memories of long nights before the terminal and the desire to cast aside the petty demands of everyday life (eating, sleeping, dressing...). A fun, deeply involving addiction.

It's the word that best describes Steven Jobs, Lee Felsenstein, Philippe Kahn, Dave Rand, George Scolaro, and just about every other computer freak of note. In fact, it describes just about everyone who writes or reads Micro C.

Hackers are people who love computers because.... Well, just because.

I can't explain it any more than dog lovers can.

"Hello Mum, I see you're a dog lover. Can you tell me why hackers love computers?"

If hackers had been scheduled to arrive fifty years earlier, they'd have taken a rain check. I did.

Why The Soapbox?

I just watched Wonder Works on public TV. The story was about a computer program that developed a mind of its own. Great story line.

They butchered it. The program (both TV and software) ran amuck: killing people, moving large sums into bank accounts, changing school grades, and making a nuclear plant go critical (because the program needed more memory). It was an absurd cliche.

And, to top it off, the main character was a skinny high school kid with glasses. Need I tell you his name? (Hint: it's that nasty 'H' word.)

Type Caste

I judged a programming competition two days ago. What I saw was eye-opening.

First, every entrant was handicapped. Of the 10 individuals and 3 teams, I did not see a single person touch-typing.

It reminded me a lot of Tektronix. Tek made us raise our right hands and promise not to use more than two fingers. (I crossed mine.)

Hunt and peck is a real disability. Remember the time when the only good copy of the source lay in an inch-thick listing? Remember the blood?

But reentering code isn't the only time that peek and poke slows things down. After all, someone entered those 107 pages in the first place — one character at a time.

Sure, when you write new code you're taking your time, but the mechanics really distract from the concentration. In fact, many programmers hand-write their first pass because the pencil is less distracting.

Hey, wake up! This is 1987! (Soon to be 1988 by the looks of that listing.) Get the lead out! (Not the pencil lead, dummy!) Think how your programming style would improve if your thoughts flowed effortlessly and accurately off your fingertips. Imagine comments magically appearing with the code. You're thinking of them as you write. Right?

Two Tricks

1. Start with your fingers on the home row (index fingers on the F and J, little fingers on the A and ;). Fingers reach when they have to (the keys slant, so the little finger on the left hand gets Q, A, and Z), other fingers follow suit. ALWAYS use the same finger on the same key. (That's the key.)

2. Wean your eyes away from the keyboard as soon as possible. Watch the screen. Put little bumps on the F and J keys if they aren't there already. (Anything you can feel with your index fingers.) Learn how to delete a mistake without peeking.

It's hard. It's frustrating. But, in a week you'll be typing faster than ever. In a month you'll be flying.

The President Taught Me To Type

During my last two years of college I worked on the city desk of a large daily newspaper. As low man on the desk I got the exciting jobs — obituaries, riots, and phone feeds from stringers.

At three p.m. EST every day, our Washington correspondent would find a phone booth. Then, rather than doing something worthwhile like changing into superman, he'd call our desk. It was (always) a long story. Usually a front page story.

My first day on the job I got a tap on the shoulder.

"There's a call for you from Washington."

For me?

I picked up the phone and in my deepest voice said. "City Desk."

By the time I'd put paper in my typewriter (we used long rolls) the deeper voice on the other end was three paragraphs into some kind of White House intrigue.

"Whoa," I said.

When we'd finally gotten together on that story (my version reasonably resembling his), he was ready to break all ties with our paper. (Despite high school typing class, my speed had dwindled to nearly nothing.)

A month later, he was rattling off stories at normal talking speed, stopping only occasionally to be amazed that I was still with him.

It was great training. After that first month, everything I did was at a keyboard. (Manual Underwoods were my favorite word processors.) But I didn't realize how well I was doing until much later.

A Few Years Later:

A fellow from England called up and said he was coming over to show me a new program he'd written. (England? I quickly got out my U.S. highway map.) His program turned out to be a typing tutor that was supposed to be able to help anyone type better.

"It's got 10 levels of difficulty and it measures speeds up to 120 words per minute — that's the limit on a 4 MHz Z80. With the most difficult text (level 10) no one has gotten over 60 words per minute without errors. This will challenge everyone."

"Here. Try it," he said.

"Level 10," I said.

He entered level 1. The program reported 120 wpm, no errors.

"Level 10," I said.

He entered level 3. 120 wpm, no errors.

One the next run, I selected 10. 120 wpm, no errors.

He went back to England (it's near London, by the way) muttering something about fast-fingered Americans.

Back At The Competition

There were 10 individuals and 3 teams. Each individual or team had 3 hours to do (up to) five problems.

One of the teams had three members: a UNIX/C guru from the Bay area, a C/Pascal instructor at the local college, and an engineer from Tektronix. What a group! They even had a fat Mac, and an equally fat printer.

Throughout the competition they carried on truly fascinating discussions, wrote beautiful code, and carefully thought out their algorithms. (At least that's what I gathered from scraps of conversation.)

Unfortunately, they didn't finish a single problem.

Meanwhile, a skinny, pale-skinned 10th grader with thick glasses and a Commodore 64 completed three problems. Not elegantly, perhaps (Commodore BASIC defies elegance), but his programs ran, his algorithms were intelligible, his comments cogent, and he got the right answers.

Speaking of competitions, I wonder how long it would take for someone comfortable with Prolog to complete the same problems. Probably not long. Not long at all.

Desktop Publishing

Sandy and I went to Comdex again. This year the herd was trooping along after the desktop publishing piper. In one year they've progressed from half a dozen Mac products to hundreds on the PC. Page Maker is the desk top equivalent of WordStar. And Ventura Publishing is the equivalent of WordPerfect (faster, zingier, newer).

There were vertical monitors displaying a full page, all 8 1/2 by 11, readable down to 6 point type. And there were the horizontal screens (19") that displayed two pages at a time, all readable, complete with graphics ($2,000 for the graphics board and monochrome monitor).

There were document scanners and mouses (mices?) and books of art (on disk). And, of course, I can't forget all the laser printers with their eternal 300 dots per inch.

It appears we'll see some breakthroughs shortly. Already some folks are showing liquid crystal printers at 300 dpi with 600 dpi due out any time. I couldn't distinguish their output from that of laser printers. Folks say they are cheaper to make and more dependable because they use fewer moving parts (the printers, not the folks).

(continued next page)

(continued from page 85)

Either way, Compugraphic typesetting and its kin are in deep trouble. Deep, deep trouble.

These are the people who produce those $50,000 plus typesetters with wheels and mirrors and photographic paper in light-tight holders. Compugraphic systems are very impressive. They have motors, power supplies manufactured by welding companies, and gigantic enameled cabinets. To that they add fancy paper processors with vats of smelly chemicals, and $1,000 per month service contracts.

They don't have graphics. Just type. (I don't know if they're really my type any longer.)

I told Compugraphic's service rep that his $100 per hour job might not survive three more years. He just smiled.

We'll see if he's smiling three years from now.

A Hot CAD

Almost hidden amongst the remains of aging CAD packages (upgrades from Comdex past) there was a genuine gem. Visionics was showing off its CAD/CAE package.

This package should sell many, many clones because it does everything. In fact, if I were still helping Micro-Sphere design circuit boards, I'd have bought a copy on the spot.

It draws schematics, lays out multi-layer circuit boards (to 26 layers), and it also does logic tests on TTL circuits. Though the package is only $975, it's incredibly complete.

The schematic portion comes with a symbol library (which you can add to); supports mouse, tablet, and keyboard inputs; and lets you rotate symbols (to name just a few of its features).

There's a netlist which extracts data from the schematic for use by the circuit simulation module, the report module, and the printed circuit module.

The printed circuit module is really fun. It lays down the ICs and makes all the connections in rat's-nest fashion. You can then route things any way you wish. (You pick up a line and stretch it like a rubber band around and through the board.) You can rotate or shift any IC without breaking its connections.

After you've finished the layout (you work with two layers at a time, with each layer displayed in a different color) it will print: the circuit plots, the solder mask, and the silkscreen mask. It will even generate a paper tape for controlling N/C drills (including bit changes).

They also sell an autorouter (costs $975). The autorouter worked fine except that it made nearly every pin connection right between the pin and the adjacent pin. I've since talked to one of their engineers about the problem and he told me they've just fixed it.

Support is via a toll-free 800 line for the first 30 days. After 30 days you get free support via a regular phone line. They also have an RBBS for technical help, tips, and announcements. Bug fixes are free. Upgrades to new versions cost only the price difference.

Visionics
1284 Geneva Dr.
Sunnyvale, CA 94089
408-745-1551

P.S. They're just coming out with version II. It adds support for fab layouts, auto-dimensioning of schematics, displays up to 26 layers at a time (not just two of the 26 at a time), will do 5-mill surface mount, supports user definable trace width, and more. It will cost $1875. Meanwhile they are continuing to update, support, and sell version I.

Other Comdex Developments

Apple didn't bother to come, and Atari's was the busiest booth of all, but it was the only place I saw any Atari related products. The rest of the show was just one big blue advertisement.

Commodore wasn't there at all.

Shows have come a long way from the original West Coast Computer Faire. That was a show — "The Jim Warren on roller skates show." User groups got free space, small companies lined the walls, and the uniform of the day was beards, jeans, and overalls. Hackers were revered.

The first speaker at this year's CD ROM forum started his talk with:

"Fortunately, none of us are hackers so we don't wear dirty shorts, we don't sit on the floor, and we don't die young."

I was stunned. What that statement had to do with CD ROMs or Comdex or the state of Nevada or whatever, I hadn't the faintest. But I guess I shouldn't have been too surprised in a crowd where the clothing come in two flavors, two-piece and three-piece.

SOG VI

We've got the dates for the Semi-Official Get-together #6. SOG VI will open with the usual white water rafting and cookout. (What am I saying? The "usual white water rafting"?) Hey, this is a splash.

Anyway, it'll be July 30 (Thursday) through August 2 (Sunday). We have the dorm again — I mean if we couldn't get the dorm it wouldn't be a SOG. There'd be no place to hold the technical marathons.

See the SOG information page in this issue and mark your calendar. It'll be better than ever (if that's possible), especially if you're there.

And be sure to bring the whole family. Last year we had some computer co-pilot sessions and some special trips (horse back riding at Suttle Lake and lift rides to the top of Mt. Bachelor) that kept the spouses and kidses interested (a few SOGgers slipped out for the trips too). We plan to continue those fun tours.

Plus, we took evening trips out to the Pine Mountain Observatory (the view's better) for a guided look at large balls of silicon...

Thanks, I Think

Actually a few of you were kind enough to lambast me with "Harass An Editor Letters." Several of you noted that you'd taken the opportunity to slice up other editors. It worked: in a recent phone survey I didn't find a single sleeping editor (or was it a single editor sleeping?).

Interestingly enough, none of them were taking calls. (Hey, you guys were supposed to write, not call.)

CP/M Turbo Modula-2

Remember Turbo Modula-2, the CP/M version Borland wasn't going to release until the MS-DOS version was finished? Well, MS-DOS still isn't finished, but you can now purchase the CP/M package. Don't call Borland, however, call Echelon, the ZCPR folks. They are handling the marketing so MS-DOS folks won't get confused.

Turbo Modula-2 definitely mimics the Turbo Pascal environment and includes: the editor, compiler, and profiler. It's a two stage compiler, the first stage generates m code. You can run the m code under Turbo. The 2nd stage generates object code which is 3 times as large but 10 times as fast as the m code. You can have unlimited overlays, so data size is the main restriction on program size.

Unfortunately, there is no translator for Turbo Pascal files, and there is no debugger. However, the linker can link in Microsoft .REL files. The manual is 552 pages.

Turbo Modula-2
Echelon Inc. 415-948-3820
885 N. San Antonio Rd. Price $69.95
Los Altos, CA 94022

In This Issue

This issue is full of Modula-2 information; even the Pascal Column has been modularized (modulated?). If you're doing work with procedural languages (versus Prolog...) Modula-2 is probably the best.

Bruce Eckel has definitely upped his technical level this time with his oscilloscope project. However, if you'll build the unit and take the time to understand what he's up to, you'll learn a lot. A whole lot. Meanwhile, you hardware neophytes shouldn't resist the Ohm's Law discussion in the "Quieting A Noisy Fan" article.

And, if you're looking for a little extra income, check out the article on Debugging a Directory or find out what it takes to run a successful computer store in "On Your Own."

Survey

We've gotten back 50 of the 1500 survey forms we put in magazines. The survey is already generating a lot of information, but we'd like to hear from all of you who received forms.

So far: you are professionals, most between 35-45 years old, you have attended college (nearly half have graduate degrees), and you are really into structured languages. Many of you have built or modified a computer.

Your comments on what we're doing and where we're going are really helpful.

So, if you have one of those yellow questionnaires bound in your copy of issue #33, sit down and fill it out. Then send it back in the prepaid order envelope. We'll really, really appreciate it.

Mouses

Logitech is famous around the Micro C office for its serious implementation of Modula-2, so it was with surprise that I read the release on their latest product. I pass it along in a somewhat abbreviated form.

"We offer two-button mice, but we believe that the three-button design is better." (Mouse suits?)

"We believe we can be strong in the retail market because of our technological edge and high quality. We've had years of experience designing and producing mice, and throughout that time we've maintained a strategy of controlling every aspect of mouse design and production. Now we're applying this background to our retail mouse models."

Rats, just when I was working up something for the culture corner, someone had to manufacture a mouse tale of no small proportions. Talk about leaving me in a hole.

David Thompson
Editor & Keeper of the Mouse ■ ■ ■

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Turbo And The Sane Editor

If Tidbits sounds a little like an advertisement for Borland International this issue, I apologize in advance, but it's for a good reason — the Turbo family is coming in really handy this season.

Borland has developed and refined its reputation by building on previous products. For example, Turbo Pascal led to Turbo Toolbox, Turbo Graphix, Turbo Editor Toolbox, and Turbo Games Works, enabling Turbo Pascal programmers to create powerful programming environments and applications, by mixing and matching. But sometimes, it turns out, it's easier to match and mix than others, and therein lies my story.

Turbo Lightning

Because I'm a stickler for trying to get things right (which unfortunately doesn't mean I always get them right), I often reach for dictionaries and other references when I'm working. Some of those references are just out of reach or across the room, but one I'm using a lot lately isn't; it's under my thumbs and fingertips: Turbo Lightning — a RAM resident spelling checker and thesaurus — which has saved me a lot of long reaches.

It's easy-to-use and powerful, and I recommend it highly if you, like me, like to stickle when you're writing.

Recently (continuing the Turbo Pascal tradition), Borland has taken Turbo Lightning a step further, releasing a Lightning Toolbox, "Word Wizard," which permits Turbo Pascal programmers to call the Turbo Lightning engine from within their own programs.

Word Wizard

Word Wizard is a couple of things — a set of word games (wasn't it George Morrow who said, "we need to put fun in software"?) and a set of procedures to call the Lightning engine from Turbo Pascal.

The games ("Code Cracker," "CrossSolver," and 5 more) are fun and include Turbo Pascal source code. So you can play first, and if you decide to write your own game (or other application), use the Lightning engine to call a dictionary to extend its vocabulary.

For example, I wrote a little database program to keep track of the references Alison (my wife) and I use when researching and writing. In our system, a typical record contains fields for —

AUTHOR, YEAR, ARTICLE TITLE, JOURNAL, VOLUME, PAGE#, COMMENTS

One of the features I built into the database engine (which I call DB-LITE) is a subject search, which searches the database for keywords in a title and lists the records containing the keywords (or subjects).

It's simple, but effective, with only a drawback or two. For example, let's say I'm working on an article about games, and I want to know if my database contains any references to "games." Entering "game" would get me some of them, but probably not all, since at least 20 other words, "scheme," "play," "competition," etc., could lead me to articles that might be game-related.

I could enter each synonym for "game" as I think of it, but I'd be gambling on a memory which isn't reliable. I simply won't think of every possibility. And besides, I prefer lazy (slow and easy) approaches.

I could look up the synonyms in a thesaurus, but there's a lazier approach — I can have the Lightning engine look them up for me.

So, I came up with this system: each time I search for a subject, DB-LITE gives me the option of extending my search with synonyms extracted from a thesaurus by the Lightning engine.

If I ask for alternatives, DB-LITE calls the Lightning engine, which reports the number of synonyms and writes them to a buffer. DB-LITE then lists them, and I can step through one by one, deciding whether I want to search the database for references containing a particular synonym.

The database treats each synonym as though it were a search keyword entered at the keyboard. And I can abandon the search at any point by pressing <ESCAPE>.

So, imagine a reference system comprised of 1) a database (data, database engine, and user interface), and 2) an interface to the Turbo Lightning engine.

Word Wizard is the key to the Lightning interface, allowing me to call its engine and do things (like find synonyms) with the dictionaries and thesaurus.

The procedures to call Lightning are provided by the Word Wizard Toolbox, in Turbo Pascal source, so all you need do, for example, is interface your Turbo Pascal Toolbox database to the Word Wizard Toolbox and you're on your way.

Turbo Pascal & Turbo PROLOG

You are, but I'm not. At least, not right away. I wrote my reference database in Turbo PROLOG, not Turbo Pascal. No problem, you say, just link those Turbo Pascal engine procedures right in. (Oh yeah, almost forgot, it's version 4.0 of Turbo Pascal that's going to allow linkable object files.)

So, what are my options?

1. Translate those Turbo Pascal procedures to Microsoft Pascal (MPascal links to Turbo PROLOG)?

I don't have Microsoft Pascal.

2. Translate those Turbo Pascal procedures to Turbo PROLOG?

Sounded strange, but I couldn't think of a better option, and it might be interesting, so I jumped into it.

Details

The Turbo Pascal procedures to call the Turbo Lightning engine use interrupt 16H to call the ROM BIOS. In Turbo Pascal, you'd call the engine with —

Intr(Interrupt#, Registers)
where Registers = a record of registers.

You assign registers values corresponding to the services you want the Lightning engine to perform.

For example, if —

BL = 00h(function code)
AH = EDh(Lightning signature)
BH = EDh(Lightning signature)
DS:SI = address of string

and you generate an interrupt 16h, the engine will check whether Turbo Lightning is loaded (the first step in using the engine).

You can generate an interrupt in Turbo PROLOG with its BIOS predicate

bios(Interrupt #, RegistersIn,
RegistersOut)

So, to call the Lightning engine, I set the registers, called the BIOS, and voila, it worked. Turbo Lightning was loaded.

I began to feel better about my programming skills. I tried a few more engine calls, found the addresses of some important tables (there's a lot of pointing around in the engine), and then decided to get down to business — find the synonyms for the search word in my database.

Splat! Pie in the face. A key interrupt returned a zero in the AX register — i.e. —- no synonyms found. Wrong, wrong, wrong. I know there are synonyms for "game"; I've checked already.

So I went back to the drawing board, checked my figures and angles, called the BIOS predicate another 120 times or so, trying every alternative I could think of (just in case I was doing something stupid), but no luck. (I probably was doing something stupid but I didn't know what it was!)

But I had started thinking, and a little effort eventually got me to the problem — Turbo Pascal doesn't handle strings the way Turbo PROLOG does. And I was passing strings around like a kite vendor.

When Strings Aren't Strings

A Turbo Pascal string begins with a length-of-string byte, and a Turbo PROLOG string has no length byte but ends with a null (0) byte. (See Figure 1.)

What's garbage to one language is meat to the other.

I needed a Pascal string, and the quickest way I could figure how to get one was to write a Turbo PROLOG to Turbo Pascal string converter.

To do it, I —

1. passed the PROLOG predicate (procedure) a string;
2. determined the string's segment and offset (using the built-in Turbo PROLOG predicate, "ptrd_word," which returns the segment and offset of the string passed to it;
3. determined the string's length (using "str_len," another built-in predicate);
4. converted the string to characters (using "str-char," my predicate);
5. and then wrote the string back to the same address in Turbo Pascal form —

LENGTH BYTE + CHAR + CHAR

See Figure 2.

The result is a more intelligent (if not ARTIFICIAL) database, with a range of possibilities (for example, adding an intelligent natural language interface).

If you want to know more about DB-LITE, write me in Davis. For more info about Turbo Lightning, Word Wizard, Turbo Pascal, or Turbo PROLOG, contact —

Borland International
4585 Scotts Valley Dr
Scotts Valley, CA 95066

■ ■ ■

Figure 1 - String Representation

```
                "GAME" In Turbo Pascal
 ___________________________________________________
|   |   |   |   |   |   |   |   |   |   |   |
| 4 | G | A | M | E | * | % | @ | ^ | ~ | } |
|___|___|___|___|___|___|___|___|___|___|___|

                "GAME" In Turbo PROLOG
 ___________________________________________________
|   |   |   |   |   |   |   |   |   |   |   |
| G | A | M | E | 0 | ~ | & | # | ^ | @ | [ |
|___|___|___|___|___|___|___|___|___|___|___|
```

Figure 2 - String Converter

```
PROLOG_To_Pascal(Str):-
    Str = First_String,
    ptr_dword(Str,DS,SI), /* find segment & offset
                             of Str */
    str_len(Str,L),       /* find the length of Str */
    String_To_Char(Str,Charlist), /* Convert the string
                                  to chars */
    write_char(DS,SI,Charlist), /* write the char list
                                as a Turbo Pascal string */
    membyte(DS,SI,L).           /* write length byte */

String_To_Char("",[]).        /* Convert a string to a list
                                of chars */

String_To_Char(Str,[H|T]):-
    frontchar(Str,H,S1),
    String_To_Char(S1,T).

write_char(DS,SI,[]).        /* Pass the string's segment
                                and offset. */

write_char(DS,SI,[H|T]):-
    char_int(H, H_ascii),    /* Char to ASCII */
    S = SI + 1,              /* Add 1 to begin the string
                                after the length byte */
    membyte(DS,S,H_ascii),   /* Write the char */
    write_char(DS,S,T).      /* Continue while we still
                                have chars. */
```

Disk Problems With MS-DOS 3.20

In issue #33 you indicated some problems occurred in using MS-DOS 3.20 on a micro with only two 360K drives. I ran into the same difficulty in attempting to use PC-DOS 3.20 on a SPERRY PC with two 360K drives.

The problem occurs not only in DISKCOPY, but also in DISKCOMP and FORMAT. It appears that all three programs expect to find a 1.2M drive in physical drive 0. Luckily, COMMAND does not.

The work-around is to create new logical drives for existing physical 360K drives. The three problem programs will work properly if they access the new logical drives.

I do agree that the need for new logical drives should not have arisen, especially since COMMAND hasn't had any difficulty in identifying the physical drive types.

Another difficulty which IBM has created for purchasers of PC-DOS 3.20 is that DOS now expects to find the IBM label in the diskette. This means the user had better have formatted the backup diskettes on PC-DOS 2.10 or higher. Otherwise he's in for a lot of diskette editing.

K. Kolovrat
378 Morris Dr.
Fairborn, OH 45324

Editor's note:

For these very reasons, many folks are getting MS-DOS or PC-DOS 3.10 instead of 3.20. Also, there is a difference between MS-DOS and PC-DOS. According to Microsoft, IBM wrote the code for the customizable parts of PC-DOS. (And parts of the utilities.) With MS-DOS, Microsoft wrote them. Anyway, there may be some differences in bugs between the MS and PC versions of 3.20.

Epson Graphics To Okidata

Recently I ordered your Kaypro disk K5 (Epson MX-80 Graphics). I have a new Okidata Microline 193, but I know most of the graphics commands are similar, with the possible exception of the codes for the variable line spacing. Since the source was included on the disk, and I have a Mix C compiler, I felt confident that, if necessary, I could modify and recompile the code.

Some changes are necessary in the source code for this configuration, so for others in a similar situation, here are my notes:

1. Concatenate Mix C's header files STDIO and STDLIB.H into a single file LIBC.H which is referenced in some of the graphics routines.
2. Change all occurrences of "return()" to "return".
3. Change "exit()" in GRAF.C to "exit(0)".
4. There are some differences in the file handling systems between Mix C and the author's development compiler, Manx Aztec C II. MS-DOS versions of Mix C require no changes. For CP/M versions of Mix C, the runtime must distinguish between text and binary files because of the filtering during text file I/O.

Use "rb" instead of "r" for the access mode in calls to fopen() where binary files are used (DEMO.C and GRAFFILE.C). I would also recommend using "wb" instead of "w" based on consistency, even though I haven't tested whether there is a difference in output operations.

5. Mix C's newline character is NULL. In DEMO.C, the function pr() tests for the null character when outputting to the printer, but encounters the newline and does not execute a carriage return/line feed since it is null. Therefore, add the line

```
crlf(1);
```

to the end of function pr().

6. If you have an IBM-compatible graphics printer, you will need to modify the setline() function in the DEMO.C and GRAFUTIL.C files. Add these lines:

```
bdos(LIST,ESC);
bdos(LIST,'2');
```

just before the end of the function definition, to activate the variable line spacing instruction which was just executed.

Mike Rovak
1406 S. 107 E. Ave., #G
Tulsa, OK 74128

Help For Fixing Kaypros

1. Friendly Modeling Compound can be used to form a handle on a cut down allen wrench. This helps to get at those hidden little socket screws that hold the Kaypro disk drives in place. This granular material melts in hot water and then hardens at room temperature. While it is molten, you can easily shape it into any shape. The material is available from:

The Friendly Plastic Company, Ltd.
2888 Bluff Street #233
Boulder, CO 80301
Phone: 303-530-5115

It costs about $25 for 1 3/4 pounds. It's a substantial price but it's a LOT of material. Share it with a friend, a wife, cat, hamster... Anybody.

2. A new book on repairing Kaypros (including the Kaypro 16) that the novice can use is:

Chilton's Guide To Kaypro Repair And Maintenance
By Gene B. Williams
Chilton Book Company
Radnor, PA 19089
ISBN 0-8019-7626-X
1985, $12.95

3. An excellent source of reconditioned, tested, reasonably priced (about $40 for SSDD and $55 for DSDD) Tandon disk drives is:

Janick Data
1869 River Birch Drive
Sumter, SC 29150
Phone: 803-481-9205

4. Inexpensive binocular magnifiers that flip-up. They're called Peer Binocular Magnifiers #22-845. They have a working distance of 10", 2 1/4 X power and cost $12.95. They are available from:

Maxon
P.O. Box 243
Carlstadt, NJ 07072

Frank and Lois Gadek
RD #1, Box 221-I
East Mill Hill Road
East Greenville, PA 18041

FIND 51 Hint

We've found a way to get the FIND 51 program on your Kaypro disk #9 to work in all WordStar documents (you suggest in your notes that it only works with upper case text.) Merely leave off the last letter of the word you're searching for. Using this strategy the program is case insensitive.

This utility is very fast and easy to learn and use. For those who do a great deal of writing it is an invaluable tool because it allows the writer to find references in a text quickly and easily.

Richard Zakin (art department)
David Sargent (psychology department)
Art Department, Tyler Hall
SUNY Oswego
Oswego, NY 13126

Striking Ribbons

I've recently discovered that the Okidata Microline 83A uses only the top half of the the ribbon. Thus, it is possible to reverse the ribbon and use the bottom half to get even more mileage out of it. This trick may also work with other printers that use half inch ribbons.

Another item I was not aware of until a recent trip to our local stationers, is the existence of Multi Strike Black Film ribbons. I bought one just to experiment with, and it appears to give a nice, sharp, black impression — not as smudgy as a new nylon ribbon. I don't know yet how long it will stay black. This depends on the quality of the film, and the coating, which appear to be quite substantial.

Anyone who would like to try these can write to:

Goldsmith Bros.
257 Jericho Turnpike
Mineola, NY 11501

Order Okidata Multi Strike Black Film ribbons, catalog #RIPL473. One ribbon costs $3.75, and quantity discounts are available — 10% for six ribbons, and 20% for twelve ribbons. When threading these ribbons into your printer, be sure that the shiny side faces the print head, and the dull side (with the carbon coating) faces the paper.

Herbert L. Polak, P.E.
39 Violet Ave.
Mineola, NY 11501

Editor's note again:

The Okidata Microline 82B uses the other half of the ribbon. The 82B is an accessory (after the fact) for the 82A. If your dealer doesn't believe you, show him this.

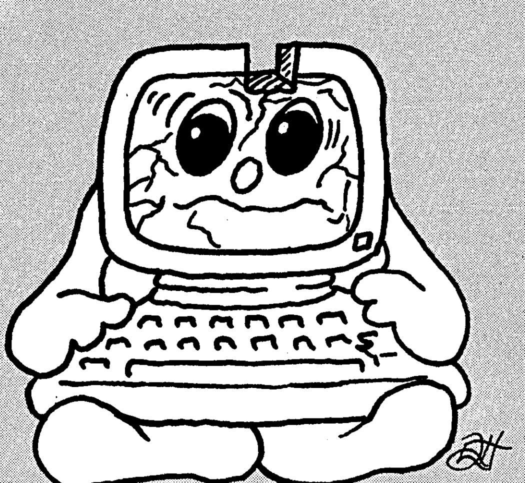
As for multi-strike ribbons, there's something even better. You can now purchase first-strike ribbons. They're not cheap but they're powerful. You can get them from any military contractor.

■■■

WANT ADS

The following folks are reaching you for only 30 cents per word. If you would like to reach the same audience, send your words and 30 cents for each to Micro Cornucopia.

■■■



Is There A Gap In Your Info?

Fill in your back issues of Micro C today

ISSUE #1 (8/81)
Power Supply
RAM Portection
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)
Drive Maintenance
Interfacing Drives
Installing A New BIOS
Flippy Floppies
C'ing Clearly Begins
Xerox 820 Begins
28 pages

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
Saving A Flakey Disk
Hooking Wi... BII
The Dis... ...ctor
JRT ...
S... Keyboard Interface
Pascal Procedures Begins
36 pages

ISSUE #11 (4/83) SOLD OUT
BBI Expansions
BBII Details
Dyna, R... ...sk Review
Eas... ...erse Video Cursor
Pla...erCalc Review
Kaypro Column Begins
36 pages

ISSUE #12 (6/83)
256K for BBI
Bringing Up BBII
dBase II
Look at Wordstar
Double Sided Drives for BBI
Packer RAdio
5MHz Mod for Kaypro
40 pages

ISSUE #13 (8/83)
CP/M Disk Directory
More 256K for BBI
Mini Front Panel
Cheap Fast Modem
Nevada COBAL 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)
Build An $800 Clone
Fixing Your Own System
NEC V20 Processor
Selling Your Own Software
Inside Small C Compiler
104 pages

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 The Real World,
Bruce Eckel's Step 2.
Introduction To Fractals.
From Boots To Device Drivers:
Secrets Of MS-DOS
Professionalizing With Turbo Pascal
96 pages

BACK ISSUES OF MICRO C.

Only $1.50 each when you order 6 or more
All Foreign $2.00 each for 6 or more
(includes surface postage)

U.S. Regular Price .... $3.00 each
6 or more 1/2 price .... $1.50 each

Regular Foreign Price (air mail) .... $5.00 each
6 or more (including Canada) .... $2.00 each
(surface postage)

Advertisers Index
Issue 34

A Compact Introduction To CD ROM

By Gary Entsminger
1912 Haussler Dr
Davis, CA 95616

In a few quick years, the mounds of data we've been able to store and access with a micro have grown dramatically — from too-few K floppies, to and through 20 and 30 meg winnies, into online databases, and right up to many-meg CD ROMs — the new toys in town with a lot to byte about.

Like the compact audio disk player (or CD), the impressively successful new consumer format for the distribution of music, CD ROM is an optical storage device which can store vast quantities (550-600M) of digital information.

CD players were big hits in 1986, with prices dropping significantly — from highs in the $1000+ range in early 1984 (after being introduced in Japan in late 1983) to lows of $150 by the end of 1986. The CD's success will no doubt inspire and motivate the CD ROM market.

Both CD and CD ROM use the same disk and laser scanning technology and the same mastering and replication methods. They're virtually identical — except for the provision in CD ROM for more powerful error correction and more precise addressing of data blocks.

A CD or CD ROM disk is 120 mm in diameter, 1.2 mm thick, and has a hole 15 mm in diameter in the center. Information is stored on the disk in pits, molded onto a surface, which is coated with a reflective metal layer which is itself coated with a protective lacquer.

The pits are 0.12 um deep and 0.6 um wide, a combination which allows information to be densely packed (about 16,000 tracks per inch, compared with 96 tpi for floppies and several hundred tpi for winnies). Plus they are reliable. See Figure 1.

All optical storage devices (CD players, CD ROM drives, LV players, etc.) use a laser beam (usually a gallium arsenide semiconductor laser) focused to a tiny spot by a lens.

To read a CD, a laser beam is focused on a spiral track of pits, and the amount of light reflected back into the objective lens is measured.

Light striking a pit is diffracted (scattered), so very little gets back to the lens. But most of the light striking a flatland between the pits is reflected. So, we get ones and zeros. The reflected light, in both cases, goes to a photo detector, which produces a current proportional to the light intensity.

CD Plusses

1. For starters, 550 meg of digital data — the equivalent of about 150,000 pages of text, or 1200 standard PC floppies.

2. It's relatively cheap: a CD ROM drive will cost about $500 in early 1987, and (once mastered) disks should retail for about $10 each. (Cost of reproduction is between $1.00 and $2.00 each, but that is expected to drop by a factor of 5 by the end of 1987.) The equivalent amount of microfiche costs around $150; books, about $1000.

3. The CD ROM is virtually error free and crash proof, since the information is well-protected. The physical clearance between a CD and the nearest point of contact is 2000 times that of a winnie head or a hard disk head (which is less than 0.5 um).

The laser beam goes through the disk's coating, and because it's almost 1 mm in diameter when it hits the surface it is unfazed by dirt and scratches. Because the information is digital, there is no background "noise."

4. The CD ROM supports any combination of acoustic, video, and computer data — they are all recorded as digital information on the disk. The multimedia possibilities are limited mainly by imagination.

CD Minuses

1. CD ROM technology is still expensive. It costs between $10,000 and $30,000 to produce a disk master, and the turnaround time for mastering a disk is at best about one month. If a database evolves slowly enough (an encyclopedia, for example), one month shouldn't be a problem. But this turnaround would likely be unacceptable in a business system.

2. And since it's basically a read only medium, if you need to "personalize" data contained on a CD ROM, you'd need to first transfer it to other storage (RAM or a winnie, for example), which might be a problem if you're working with a lot of data.

3. Access and transfer aren't the best: CD ROMs are faster than floppies, but slower than winnies —

		Access	Transfer
CD ROM	--	500 ms	1.3 Mbit/sec
Winnie	--	38-65 ms	5.0 Mbit/sec
Floppy	--	...	0.25Mbit/sec
Modem	--	...	0.01Mbit/sec

4. Accessing 550 meg of data isn't trivial. To efficiently utilize that much data will require better (and better) directory methods.

5. And the Phillips/Sony Standard hasn't been accepted by everyone.

The Last Details

Some of the more affluent (and creative?) players in the microcomputer industry are already utilizing CD power.

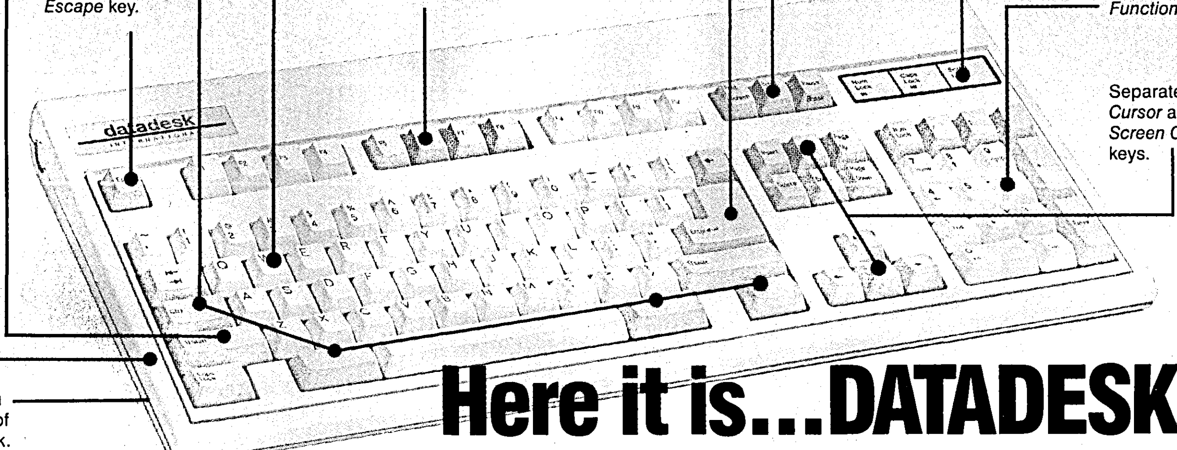
What this means, I think, is a very interesting next phase in computing. Cheers.

■ ■ ■

Figure 1 - Pits And Lands

```
_______________________________________      FRONT SIDE OF DISK
       ____________     __      __          < REFLECTIVE ALUMINUM COATING
_____| /  ^  \    |__|  |_____|  |_______
      / land  \          ^
     /         \        pit
____/___________\________________________    BACK SIDE OF DISK
   /             \
  /  LASER BEAM   \
```

Borland's Turbo Lightning FREE



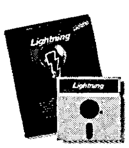
Here it is...DATADESK'S
TURBO-101 ENHANCED KEYBOARD
for the 10 million PC users IBM just ignored!

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

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

Get Borland's Turbo Lightning ™ For FREE!

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



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

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

SPECIAL OFFER!
ONLY $149.95* *FOR BOTH KEYBOARD & SOFTWARE*

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

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

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.: ______

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

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.

New Turbo Prolog and Turbo Pascal Toolboxes add more Power

New! Turbo Prolog Toolbox

Our new Turbo Prolog Toolbox™ enhances Turbo Prolog—with more than 80 tools and over 8,000 lines of source code that can easily be incorporated into your programs. It includes about 40 example programs that show you how to use your new tools.

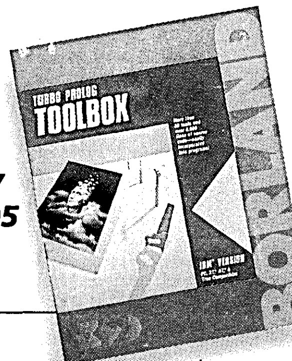
New Turbo Prolog Toolbox features include:

- Business graphic generation
- Complete communications package
- File transfers from Reflex, dBASE III, 1-2-3, Symphony
- A unique parser generator
- Sophisticated user-interface design tools

System requirements
Turbo Prolog: IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. 384K. Turbo Prolog Toolbox requires Turbo Prolog 1.10 or higher. Dual-floppy disk drive or hard disk. 512K.

It's the complete developer's toolbox and a major addition to Turbo Prolog. You get a wide variety of menus—pull-down, pop-up, line, tree and box—so you can choose the one that suits your application best. You'll quickly and easily learn how to produce graphics; set up communications with remote devices; read information from Reflex,® dBASE III,® Lotus 1-2-3® and Symphony® files; generate parsers and design user interfaces. All of this for only $99.95.

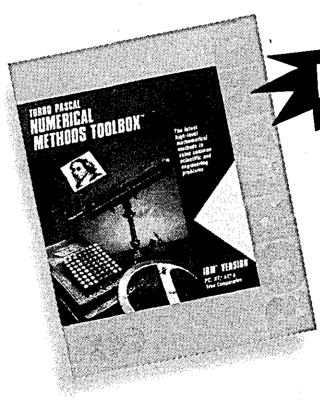
NEW



Only $99.95

The power and high performance of Turbo Pascal® is already in the hands of more than half-a-million people. The technically superior Turbo Pascal is the *de facto* worldwide standard and the clear leader.

NEW



Only $99.95

New! Turbo Pascal Numerical Methods Toolbox

What our new Numerical Methods Toolbox™ will do for you now:

- Find solutions to equations
- Interpolations
- Calculus: numerical derivatives and integrals
- Fourier transforms
- Matrix operations: inversions, determinants and eigenvalues
- Least squares approximations
- Differential equations

As well as a free demo FFT program, you also get Least Squares Fit in 5 different forms.

1. Power
2. Exponential
3. Logarithm
4. 5-term Fourier
5. 5-term Polynomial

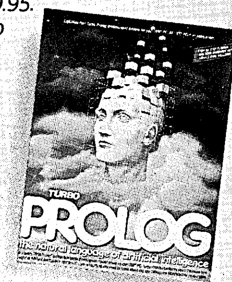
They're all ready to compile and run.

All this for only $99.95

System requirements
IBM PC, XT, AT or true compatibles. PC-DOS (MS-DOS) 2.0 or later. Turbo Pascal 2.0 or later. Graphics module requires graphics monitor with IBM CGA, IBM EGA, or Hercules compatible adapter card, and requires Turbo Graphix Toolbox. 8087 or 80287 numeric co-processor not required, but recommended for optimal performance. 256K.
Turbo Pascal 3.0:
Includes 8087 & BCD features for 16-bit MS-DOS and CP/M-86 systems. CP/M-80 version minimum memory: 48K; 8087 and BCD features not available. 128K.

Turbo Prolog, the natural language of Artificial Intelligence

Turbo Prolog™ is the most popular AI package in the world with more than 100,000 users. It's the 5th-generation computer programming language that brings supercomputer power to your IBM® PC and compatibles. You can join the AI revolution with Turbo Prolog for only $99.95. Step-by-step tutorials, demo programs and source code included.



Only $99.95!

"If you're at all interested in artificial intelligence, databases, expert systems, or new ways of thinking about programming, by all means plunk down your $100 and buy a copy of Turbo Prolog.
Bruce Webster, BYTE"

For the dealer nearest you, or to order by phone
call (800) 255-8008
CA (800) 742-1133
Canada (800) 237-1136

BORLAND
INTERNATIONAL

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373