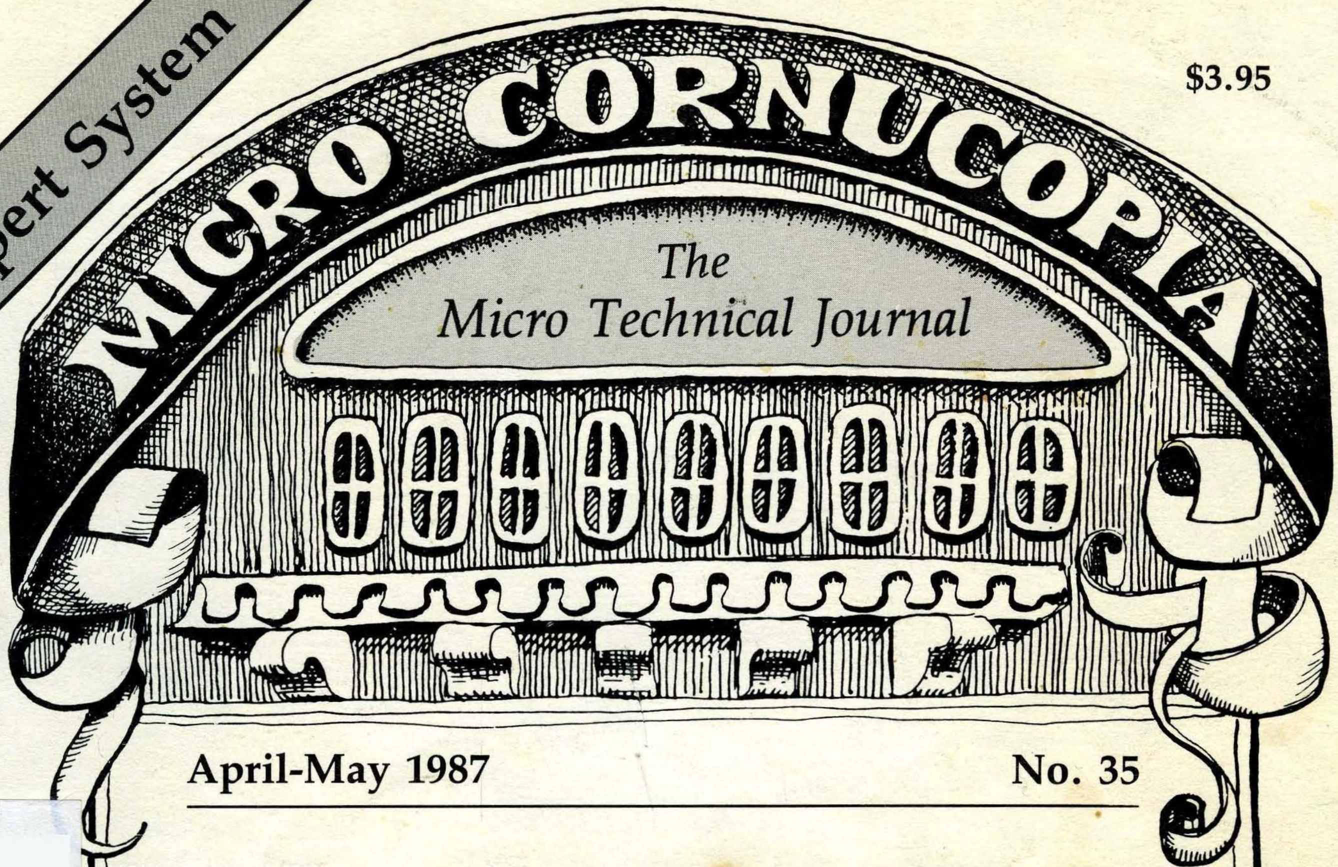
Expert System

$3.95



April-May 1987 No. 35

Software Applications

Hardware Applications

Expert System

$3.95

MICRO CORNUCOPIA

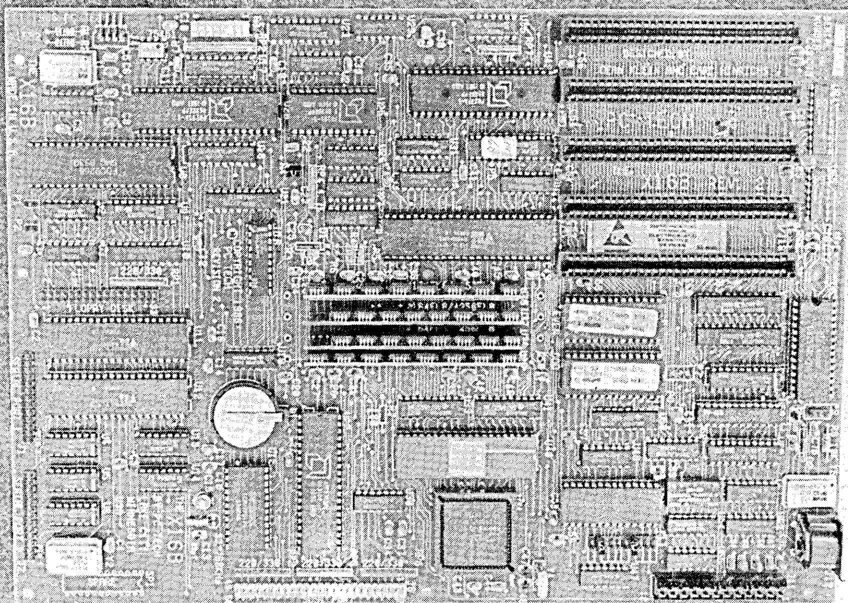
The Micro Technical Journal

April-May 1987 No. 35

Software Applications

Hardware Applications

10 MHz, 0 Wait States, American Made!



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

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

OTHER PRODUCTS

8087-1 Math Coprocessor for X16B

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

SCSI Port Option for X16B

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

FOUR MEGGER

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

2S1P

We found ourselves lacking for a flexible two serial ports and parallel printer port. So we designed the 2S1P (which is short for 2 Serial, 1 Parallel) board. It's small, inexpensive and made here, of course!

Memories

The PC Tech Memories board allows additional ROM or static RAM to be put in a PC, XT, AT, X16B or compatible. We designed it for diagnostics. You can use it for booting without disks, scraping your windshield or whatever!

PRICES!

X16B / 1 Meg / RTC $600.00
- 8 MHz version $540.00
- SCSI option $25.00
- Serial ports (2) $38.75
- 8087-1/82188 . . . . $340.00
- 8087-2/82188 . . . . $260.00

Four Megger $850.00

2S1P $99.00

Memories $95.00

Systems:
- The Box $799.00
- Mono System $1,099.00
- EGA System $1,849.00

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

PC Tech

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

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

COD, VISA, MASTER CARD, CERTIFIED FUNDS

WHY LOGITECH MODULA-2 IS MORE POWERFUL THAN PASCAL OR C.

"A clear winner… The integrated editor is a joy to use." BYTE Magazine, Jan. '87

APPRENTICE PACKAGE $99

WIZARDS' PACKAGE $199

- Separate Compilation w/inter-module typechecking
- Native Code Generation
- Large Memory Model Support
- Most Powerful Runtime Debugger
- Comprehensive Module Library
- Maintainability
- Translator from Turbo and ANSI Pascal

WIN A FREE TRIP TO Switzerland

HOMELAND OF MODULA-2

Return your Modula-2 Registration Card or a reasonable facsimile* postmarked between March 1, 1987 and May 31, 1987 to be included in a once-only drawing!

Grand Prize: One week excursion for 2 in Zurich, Switzerland including a guided tour of ETH, the University where Modula-2 was created by Niklaus Wirth. European customers may substitute a trip to Silicon Valley, California.

Second and Third Prizes: LOGITECH C7 Mouse or LOGITECH Bus Mouse with Paint & Draw software—a $219 value, absolutely free!

*Write to Logitech, Inc. for a registration card facsimile.

NEW!

APPRENTICE PACKAGE $99

Everything you need to begin producing reliable maintainable Modula-2 code. Includes the Compiler with 8087 support, integrated Editor, Linker, and BCD Module. We're also including FREE our Turbo Pascal to Modula-2 Translator!

NEW!

WIZARDS' PACKAGE $199

This package contains our Plus Compiler—for professional programmers or for those who just want the best. The Plus Compiler with Integrated Editor requires 512K and takes advantage of the larger memory to increase compilation speed by 50%. Our Turbo Pascal to Modula-2 Translator is also included at no extra charge.

NEW!

MAGIC TOOLKIT $99

We've put our most powerful development tools into one amazing Toolkit for use with either the Apprentice or Wizards' packages. Highlighted by our Runtime Debugger, the finest debugging tool available anywhere, the Toolkit also includes our Post Mortem Debugger, Disassembler, Cross Reference utility and Version which keeps track of different versions of one program. Our MAKE Utility figures out module dependencies and automatically selects those affected by code changes to minimize recompilation and relinking. We also provide source code of our major library modules for you to customize—or just play with.

WINDOW PACKAGE $49

Now you can build true windowing into your Modula-2 code. Features virtual screens, color support, overlapping windows and a variety of borders.

ROM PACKAGE AND CROSS RUN TIME DEBUGGER $299

For those who want to produce rommable code. You can even debug code running in ROM from your PC.

Turbo Pascal is a registered trademark of Borland International.

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

To place an order call toll-free:

800-231-7717

In California:

800-552-8885

YES! I want the spellbinding power of LOGITECH Modula-2!

- ☐ Apprentice Package $99
- ☐ Wizards' Package $199
- ☐ Magic Toolkit $99
- ☐ Window Package $49
- ☐ ROM Pkg/Cross RTD $299

Add $6.50 for shipping and handling. Calif. residents add applicable sales tax. Prices valid in U.S. only.

Total Enclosed $________

☐ VISA ☐ MasterCard ☐ Check Enclosed

Card Number ________ Expiration Date ________

Signature ________

Name ________

Address ________

City ________ State ________

Zip ________ Phone ________

LOGITECH

LOGITECH, Inc.
805 Veterans Blvd. Redwood City, CA 94063
Tel: 415-365-9852

In Europe:
LOGITECH SA, Switzerland
Tel: 41-21-879656 • Telex 458 217 Tech Ch

In Italy:
Tel: 39-2-215-5622

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.

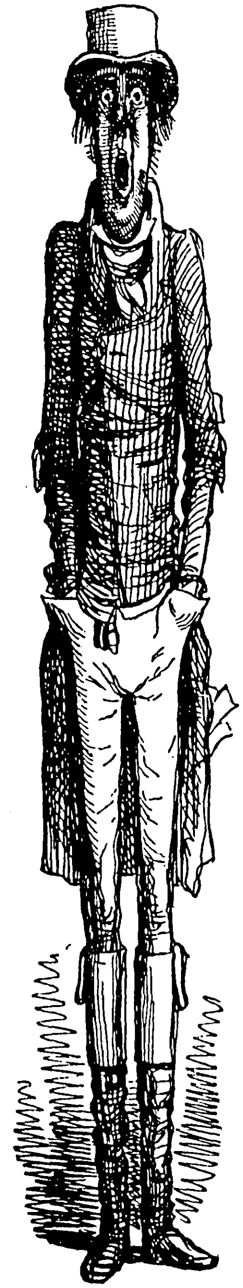
RBBS - 24 hrs. 300-1200-2400 baud
8 Bits, No Parity, 1 Stop Bit
503-382-7643

ISSN 0747-587X

AROUND THE BEND

By David Thompson

Sympathy Wanted



I'm bummed, burned–out, through with the humor and high jinks. Hilarity stinks. If you want fun, make your own.

Margret and I have been working day and night trying to desktop Micro C. We've created style sheets, added commands to articles, formatted listings, and generally enjoyed our new–found power. Everything we wanted to do (reverse video, lines, captions, illustrations...) we did. With Ventura Publisher.

Once we had our style sheets done, articles finished, boxes drawn, all we had to do was run Ventura's PostScript output through a Linotronic RIP and we'd have a magazine. So three weeks ago I packed up Sandy, the kids, and a system and headed for the nearest Linotronic.

That was 108 long distance calls (covering eight states), seven express packages (two lost), and six attempts at modem communications ago.

We still haven't found anyone (including Adobe, Ventura, Xerox, or Linotype) who can help us dump a small PostScript file through a Linotype RIP (Raster Image Processor or Rest In Peace, you take your pick).

We've badgered, berated, beleaguered, and bullied all 6 typesetting houses in the northwest that have PostScript RIPs. They were all certain they could do it. ("Hey Murphy, over here. Easy pickins.")

Everyone was surprised. All but Xerox (Xerox was unreachable) had suggestions:

"Remove the special headers we add to PostScript files when they're destined for the Apple Laserwriter," was the suggestion from both Linotype and Ventura. After deleting 194 lines of illegible hex the file still wouldn't run.

"You must be using more than 3 fonts. Of course, the internal fonts don't count." We were only using three fonts, internal or otherwise.

Some of the houses only had MACs. I should have skipped them altogether.

"Our MAC is automatically stripping carriage returns as your file comes in off the modem, and I'm getting little boxes prepended to each line. That must be the problem." The small boxes turned out to be linefeeds. Not a problem. But the RIP still couldn't output the file.

"Let's see, we take this file and output it to the RIP through the MAC's PostScript interpreter. That's always worked with Apple files."

Come On! It's already a PostScript file; if you PostScript it again it's not going to work.

Continued On Page 91

April-May 1987
Issue No. 35

CONTENTS

Features

Columns

Future Tense By Gary Entsminger

LETTERS

Ground Water?

I read Larry's comment about ground water (issue #34, "A Quiet Morning At Micro C") to my spouse, who immediately replied, "It's easy to grind water. First you dehydrate it...."

Also, even though it's not documented, you can use command line arguments with the MIX-C compiler. You must use a pointer to an array, and remember that the ^filename is the first (number zero) argument.

Gil Josephson
Square Dance Caller
719 S. Belgrade Rd.
Silver Spring, MD 20902

MIDI Info

This is in response to Thatcher Deane's letter in issue #33. The MIDI bus information is available from:

International MIDI Association
12439 Magnolia Blvd., Suite #104
North Hollywood, CA 91607

It's $35 a copy (ouch!). I did manage to get Roland Corp. (a synthesizer manufacturer) to send me information on MIDI, no charge. They also provided schematics for sample interfaces to several different processors. (Be warned; it took them many months to answer.) I'd send you a copy of the information, but I loaned my only copy out and it has disappeared.

MIDI is a relatively low speed serial bus. Data is sent at 31,250 bps. Each MIDI event (note on, note off, pitch bend, etc.) requires at least 16 bits of data, more if note velocity and pressure are included. The bus saturates at about 1950 events per second.

This may not sound like a restriction, but a single keyboard MIDI controller can momentarily fill the bus if it transmits all available information about pitch, pressure, volume, and continuous controller (pitch wheel, modulation wheel, breath controller) changes.

I've heard that the reasons for this low speed are that the designers wanted to be able to use cheap computers (Apple IIs, C-64s, etc.), and they wanted to prevent RFI in recording studios.

For enough information on MIDI to get started, BYTE's Computer and Music feature (June 1986) is a good source. There's an article by Jay Kubicky that describes MIDI in general, an IBM bus MIDI interface (full schematics), and a 16-track sequencer for IBMs and clones (source in C available from BYTE). The rest of the issue is a good introduction to computer applications in music.

Sorry I couldn't find anything to harass the editor about. (Change to a monthly format? Have a Modula-2 programming contest?) You're doing great work, keep it up. I look forward to hearing more about the PD-32 project (I'm looking into building one using Steve Ciarcia's SB180FX as a host).

Scott A. Rankin
14021 5th South
Seattle, WA 98168

More MIDI Info

At present, MIDI is a serial interface only. Although parallel MIDI research is underway, its implementation costs are prohibitive. Roland MPU-401 is the PC industry standard MIDI interface. There's a lot of MIDI software available for PC, Commodore, and Apple IIe computers. I've seen a few Turbo Pascal programs floating around which run under MS-DOS. Keyboard Magazine usually has MIDI utility programs. Here are some additional leads I've dug up:

MIDI For Musicians, by Craig Anderton. About the only published book out there that's even close to the current technology.

Musical Applications of Microprocessors, by H. Chamberlin. Mucho techie stuff.

BYTE magazine, June, 1986. A must!

Roland. 213-685-5141. MIDI interface manufacturer for PC. Also publishes a four volume set on The Synthesizer, plus audio cassette tutorials, etc.

Sonas Corp.
21430 Stratherm, Suite H
Canoga Park, CA 91304
818-702-0992

MIDI interface manufacturer for Commodore.

Passport Corp.
625 Mira Montes St.
Half Moon Bay, CA 94019
415-726-6238

MIDI interface manufacturer for PC.

Ferro Productions
228 Washington Ave.
Belleville, NJ 07109
201-751-6238

MIDI and music synthesis tutorials

Cherry Lane Music
PO Box 430
Port Chester, NY 10573
914-937-8601

Carries *MIDI and Related Interfaces* (a book), as well as hardware.

Syntech Corp.
23958 Craftsman Rd.
Calabasas, CA 91302
818-704-8509

MIDI hardware.

My Kaypro II is dying to get a MIDI interface device. Is there anyone out there who can design or point a baton in the direction of an RS-232 MIDI interface? My II will thank you. I will thank you.

Jerry Pinnell
PO Box 528
Newport Beach, CA 92661

Even More MIDI

The problem with "roll-your-own" hardware is the MIDI frequency — 31.25 Kbaud. Otherwise, it's just an average asynchronous serial current-loop-type signal, with 8 data bits and 1 stop bit; a MIDI line can be driven with a gate like a 74LS244 and two 220 ohm resistors. But the frequency's off-beat — the typical IBM serial card, for instance, using a WD8250 UART, doesn't produce that baud within 5%, and the MIDI standard calls for +/-1%.

One obvious alternative is bit–by–bit — i.e., no UART. I created such a system for my Kaypro 4, and it worked — several times. The bit time comes out to 32 microseconds, which is just enough. I used the parallel (printer) port and some circuitry, and was able to play and read MIDI (reading through the busy bit or something).

Another alternative on some computers is either to change the clock rate of the USART, or to use the USART in synchronous mode and connect your own clock to the clock inputs of the device. I followed the latter path with a second MIDI system which runs off of, of all things, an RS–422 port on this weird MP/M computer I ran into; the USART is a Zilog SIO.

One other thing — Electronic Musician Magazine contains a lot of MIDI–related text and advertising.

James Owen
35 Admiral St.
Port Jefferson Sta., NY 11776

DEBUG & Books

I bought issue #33 because of the articles about MS–DOS and MS–DOS DEBUG. The information about CONVERT came as a bonus.

The book which seems to have more about DEBUG than any other is *MS–DOS Power User's Manual*, by Jonathan Kamins, published by Sybex. You might have Earl Hinrichs review it. You may also have seen Best of Bix in the December BYTE, which mentions an assembler within DEBUG. No wonder operating systems have become so fat. Maybe there's room for a book that covers DEBUG and nothing else.

Incidentally, if you get down this way on business, Op–Amp Technical Books is well worth a visit. Allow about half a day for it. They specialize in computer books, but also have other things like mathematics, audio and recording, and building codes. The store is at 1033 N. Sycamore Ave. in Hollywood, inside a factory. They have the best selection of mass market computer books I've seen; only UCLA is better on straight computer science at the academic level. Take out a second mortgage on your house before you come.

Mike Fern, Jr.
P.O. Box 1105
Covina, CA 91722

Ultimate Speed Up

I believe I've managed the ultimate Kaypro speed up. I have my trusty old lunch bucket (II '83) going 23.7 MHz and running rings around everything else on the block. Not bad for a nominally 2.5 MHz Z80.

The trick is adequate cooling. I've jerry–rigged a liquid nitrogen system to keep my chips from frying. Everything seems to be under control. The nitrogen flow control requires surprisingly little monitoring — I'm using an abandoned AT to handle it — and the rewards are well worthwhile.

Not only does my "obsolete" machine now have unbeatable speed, the intense cold has contracted the data blocks on my floppies so my 192K drives hold over 600K each.

Joe Jacobson
827 Washington Ave., Apt. 14–D
St. Louis, MO 63101

Editor's note:
Frozen floppies aren't all they're cracked up to be.

Controlling The Real World

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

Build A Scanning Temperature Measurement System

Bruce builds on his A/D converter from issue #34. This time he selects one of eight inputs, adjusts the range and offset for each, creates a cheap temperature sensor, and then calibrates the whole mess. If you're interested in sensing with your system, this is the spot.

The physics department at UC Irvine insists that students take a senior lab. For mine, I created a solar distillation simulator, which was supposed to model a solar distiller (for turning salt water into fresh water). This model could then be used to help design cheaper, more efficient distillers.

It was a plexiglass and aluminum monstrosity, with tubes and wires coming out all over the place. It also contained a number of thermistors, to measure the temperature difference in the water flowing through the still.

To perform a calculation, I had to switch my meter to all the thermistors, write down the values, measure the electrical energy flowing to the heater, and measure the water flow with a beaker. Then I had to use my calculator to figure out what was going on, and finally, make a graph or some other visual picture.

Because of a mechanical design flaw, the model was a disaster — the water flowed in a stream right down the middle of the plate, instead of being nicely distributed. Years later, my father, tired of storing it in his garage, tossed it.

But part of the experience has never left me, at least not the feeling that some kind of machine should have been doing all those measurements AND calculations AND graphs. I have an obsessive dislike for boring, repetitive activities.

Automating Data Collection

In this article, I'll show you how to build a scanning data acquisition system, a front-end for the analog-to-digital (A/D) converter I presented in issue #34. The scanning part is really quite simple (just one chip), and it resembles the rotary switch I used to hook all my thermistors to a meter.

The challenging part of this project is getting the information from the real world. It's challenging because it involves analog electronics, a subject computer types often fear. I'll try to waylay that fear.

Intro To Op-Amps

Usually when you use an analog signal, you change it. You make it bigger (amplify it) or make it smaller (attenuate it). A device called an operational amplifier (or op-amp) is the key to keeping things simple. We don't have to know how they work to use them, so for now let's avoid confusion and just say they work.

One confusion we can't avoid is that textbook (ideal) op-amps don't look exactly like the physical devices they represent. Real chips often have many more terminals than the drawings.

The LM324 (our chip) has only three pins per op-amp. Good for us. And it's cheap (hooray for us), extremely common, and can operate with digital power supplies (0–5 V). It responds to frequencies up to 5 KHz (5000 cycles/second), so it satisfies many data-acquisition and control needs.

There are four op-amps in each LM324 package (Figure 1) plus two power pins (positive and negative or ground). Each op-amp has three pins, just like the ideal case (Figure 2). The two pins on the left (marked + and –) are the inputs, and the pin at the tip of the triangle is the output. The difference in voltage between the two

Figure 1 - LM324 Quad OP Amp

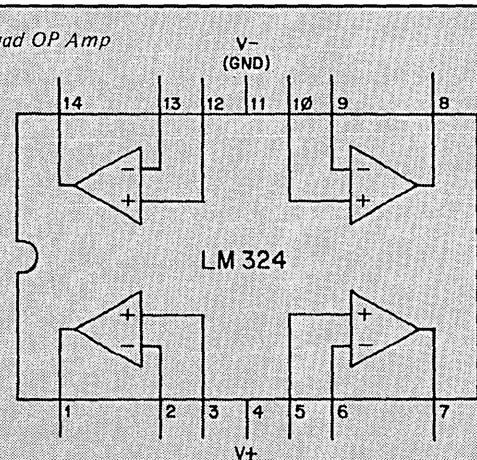
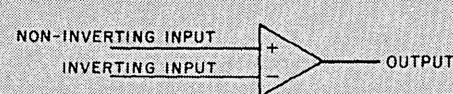


Figure 2 - Ideal OP - Amp



input pins is amplified and sent to the output.

Reality Checks & A Little Mantra

Op–amps are almost always hooked up like this — one of the inputs is connected to what you want to amplify, and the other input does a reality check on the output. This is called feedback (because the output is being fed back to an input). Here's a little mantra for thinking about feedback, the output changes to make the inputs the same.

An example: Figure 3 shows the output wired directly back to the – pin. The + pin is the input. When the input changes, what happens? The output changes to make the inputs the same — it outputs whatever it takes to make the voltage between the two inputs zero; in this case it changes until it equals the input voltage. Thus, the output voltage exactly follows the input voltage, which gives the circuit its name: voltage follower.

Sounds silly? Why not use a piece of wire instead, you ask?

Suppose you have a thermistor (resistance changes with temperature) and you're trying to measure temperature with it. If you pull any current through the thermistor, it will heat up and you'll be measuring the effect of trying to measure it, rather than the actual temperature.

Part of the beauty of an op–amp is that its inputs draw virtually no current. So a voltage follower will measure the voltage of a point without affecting that point, isolating or buffering (buffer is the other name for the circuit) what you're measuring from what you're measuring it with. You might notice that if you ignore the feedback wire, this looks a lot like a digital buffer.

As a second example, let's make a real amplifier. Figure 4 shows the feedback configuration for a variable–gain amplifier — the amplification is changed with a variable resistor. Let's see how it works. Remember, the output changes to make the inputs the same.

If the arrow on the variable resistor is right in the middle, the – input sees exactly half the output voltage. This means that to make the – input the same as the + input, the output has to be TWICE the + input voltage. Bingo! An amplifier which multiplies the input by two!

As the arrow is moved down on the resistor, the output has to go higher and higher to make the inputs the same (since there's more resistor between the output and the – input), i.e., it has to amplify the input signal more. As the arrow is moved up, the amplification (or gain) is reduced until it's just a voltage follower.

But don't jump to conclusions like, "I can build my own stereo amplifier!" The output can't do just anything; there are some limitations.

Hitting The Rails

When you hook up the positive and negative power wires, you establish how high and how low the output of the op–amp can go. (Sort of makes sense, doesn't it? How could it put out a voltage which is higher or lower than its supply?)

For example, if you use +15V and –15V power supplies (this is the limit the LM324 can handle), the output can get up to almost +15V and down to almost –15V. These limits (the high and low supply values) are called the rails. (The name was probably coined by guys who worked on electric trains or in power stations, where the current was carried in the rails.)

If you ask an op–amp to amplify more than it can, the output will just go to the appropriate rail (hit the rail) and stay there until you stop giving it such a big input signal. When using 0–5 V supplies, the rails are at 0.5 V and 3.5 V (the A/D converter must be modified to use this region only, as we shall see).

The op–amp won't put out an infinite amount of current. The LM324 will put out (source) about 20 mA and absorb (sink) about 8 mA, as long as it's not sitting on one of the rails. If you ask it for more, you won't get it.

Amplification/Attenuation

Figure 5 shows a third example, a circuit which will either amplify or attenuate the input signal, depending on where the variable resistor is set. If the arrow is right in the middle (equal resistance on both sides) and you put a positive voltage on the input, notice the output has to go negative by exactly the same amount in order to make the + input equal to the – (that is, zero) and satisfy our mantra.

This configuration is called an inverting amplifier — if the input signal goes up and down and up and down, the output will go down and up and ... you get the idea. (This is also referred to as being 180 degrees out of phase.) Inverting amplifiers generally require negative voltage supplies.

If you move the arrow toward the output, the output won't have to work as hard to make the + input zero, and the signal will be attenuated. If you move the arrow toward the input, the output will have to work harder and the signal will be amplified.

Figure 3 - Voltage Follower

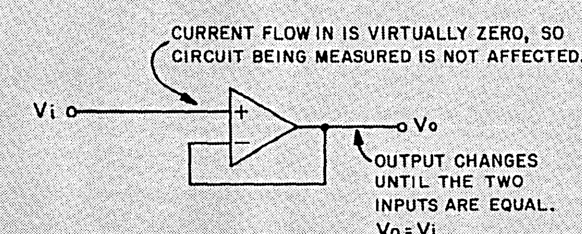
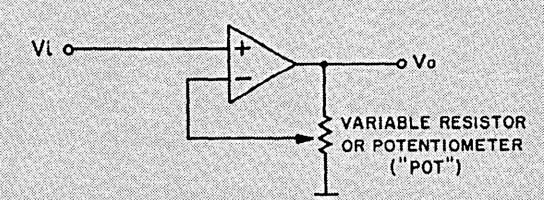


Figure 4 - Variable Gain Amplifier



(continued next page)

(continued from page 7)

Adding An Offset

Figure 6 shows a useful trick, an inverting amplifier with the – input held at some dc constant. Now the output of the op-amp will move around to make the + input equal to the dc constant at the – input. If the – input is 1 volt, the variable resistor VR1 will be right in the middle. If the input is 2 volts, the output changes to 0 volts to make the + input 1 volt (instead of –2 volts if the – input were at 0 volts).

This means the output will always have a dc voltage offset (controlled by the value at the – input) in addition to amplifying or attenuating the input signal. This can be very useful when preparing a signal for A/D conversion, since the signal is often not in the right range.

Final Amplifier

The amplifier used for the temperature scanner (Figure 9) is a composite. It's non-inverting (the input signal goes to the + input) since we don't have a negative supply, and it controls the gain and the offset simultaneously.

The offset is controlled by a resistor divider, with values selected to generate a voltage around 0.6 V (the nominal value generated by the base-emitter junction of the transistor temperature probe). The potentiometer lets you calibrate the offset. Notice that the gain resistors have a much higher resistance than the offset resistors — this insures that the current flowing through the gain section will be small so it won't have a noticeable effect on the offset value.

The gain control is a modification of something we've seen before. The – input is tapped into the middle of some resistance. The effect is the same as Figure 4, but the drawing has been rearranged a little.

There are tons of things you can do with op-amps (they're especially useful in control systems). You can add or subtract two signals, look for peak values, even take the logarithm of a signal (no time-consuming conversions — it's all done in analog!). And I leave it to you to discover some of them yourself. For now, let's return to our project.

The Scanner

The object of the scanner is to take eight analog inputs and connect them, one at a time, to the input of the A/D converter so the value of each can be measured. To do it, let's use a really nifty device called a CMOS analog switch.

These come in a number of configurations, but they're all really like those knife switches you see in old Frankenstein movies for connecting the lightning to the monster's body. The 4053 has three single-pole, double-throw (SPDT — one knife, two possible victims) switches, the 4052 has two single-pole, four-position switches (SPFT?) and the 4051 (which I'll use here) has one pole and eight positions. (This is one of the few cases I've seen where the part number seems to have anything to do with the function).

It would save some wiring if we were to instead use the ADC0809, an A/D converter which has a built-in eight channel scanner. I'm presenting the scanner as a separate device since the analog switch is useful by itself.

Inhibiting Igor

Figure 7 shows the scanning circuit. The inhibit pin on the 4051 keeps Igor (the mad scientist's demented assistant) in his cell and all the switches disconnected — we don't care about this, so we ground that pin.

Pins A, B, and C are digital selection bits 0, 1, and 2; they're connected to pins 0, 1, and 2 of the PIO's port B (the PIO was connected to the PC in issue #34). Since the PIO has TTL outputs and the 4051 expects CMOS levels at its inputs, we must shift the levels. The easiest way to do this is with HCT logic (lovely stuff — takes TTL or CMOS inputs, and will drive TTL or CMOS). I used 74HCT00s connected as inverters (note the compensation for the inversion in the program), but there are many other options.

Figure 5 - Inverting Amplifier

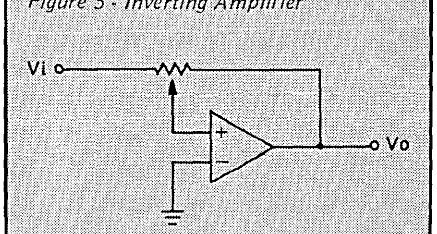


Figure 6 - Amplifier With Offset

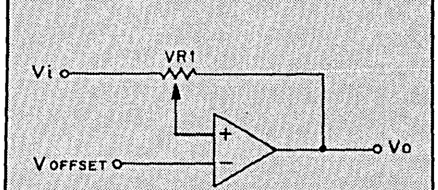
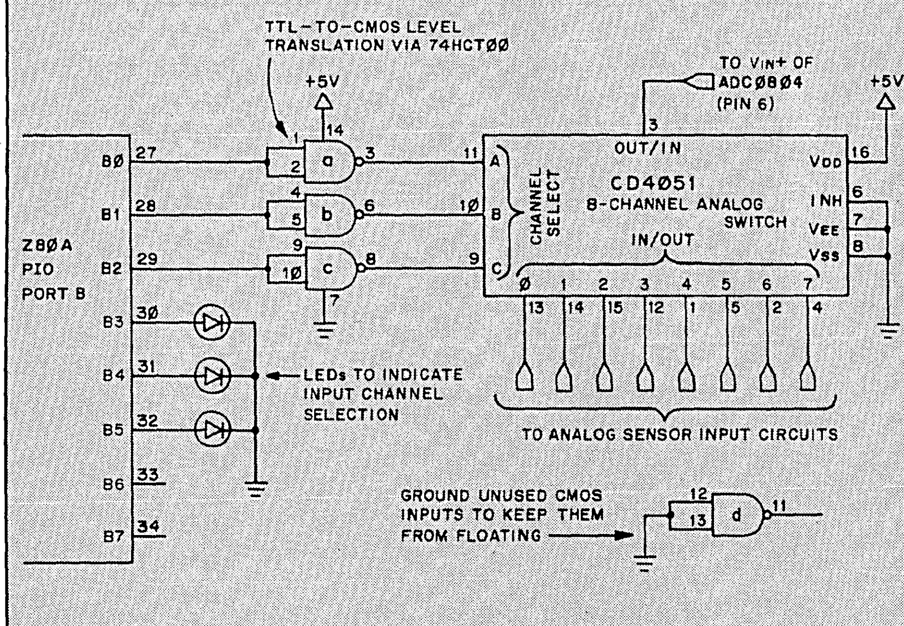


Figure 7 - Eight Channel Analog Scanner



I added 3 LEDs to indicate (in binary) the selected input channel.

To use the switch, you simply put the binary value of the input line you'd like to measure on the bottom three bits of port B. Igor, upon seeing this, will gallop off to throw the appropriate knife switch. It works better than a rotary switch since he doesn't click through all the in-between positions, and there are no switch contacts to bounce.

There is one problem. The analog input voltage can't go beyond either of the analog voltage rails.

Vdd is the high analog voltage limit, and Vee is the low analog voltage limit, which can be negative. Vss is the digital ground, and the digital switching threshold is determined by the value of Vdd.

If the analog input goes outside the rails, the switch will be damaged. Since I'm doing my best to make this simple, I want to use the computer's 5 V power supply (which was brought out in the last issue). This means Vee and Vss will be at ground and Vdd will be at 5 V. Our analog input voltage has to stay within these values.

If we were to simply connect one of the analog switch inputs to some sensor, it might see values which are outside the rails of the switch. To prevent this, the op-amp has a convenient feature, no matter what its input is (and the op-amp doesn't mind if its input is outside its supply rails) its output can't go outside the op-amp rails (which, because of internal limitations, are at 0.5 V and 3.5 V when the output of the op-amp is supplying current). Thus, the solution is to make sure there's an op-amp with 0 V and 5 V supplies between the sensor and the input to the analog switch.

A/D Converter Mods

The A/D converter connection (Figure 8) is just like the one in issue #34 (check there for timing diagrams, installing a PIO, and a description of why the converter was connected that way), with one exception. Since the output of the LM324 op-amp with 0 V and 5 V supplies can only swing from 0.5 V to 3.5 V, it would be a waste to have the range of the A/D go from 0 V to 5 V — only 3/5 of the numbers the converter can put out would ever be used, so you wouldn't get as much resolution.

We want the converter to read 0 at 0.5 V (the bottom of our range) and 255 at 3.5 V (the top). To do this, the reference and offset must be changed.

The converter takes the voltage difference between Vin+ and Vin- and turns it into a number. If the voltage between these two pins is 0 V, the value it returns is zero. If the voltage is equal to Vref (a value we determine with the Vref/2 pin), the converter returns 255.

The Vref voltage (reference voltage: largest measurable voltage between Vin+ and Vin-) is established by the voltage difference between the Vref/2 pin and the A GND (analog ground) pin. In our system, analog ground is at zero. We want the voltage range measured by the converter to be 3.5 V – 0.5 V = 3.0 V, so we must set the Vref/2 pin to 1.5 V.

To get the desired offset, we simply set the Vin- pin to the low end of our range (0.5 V).

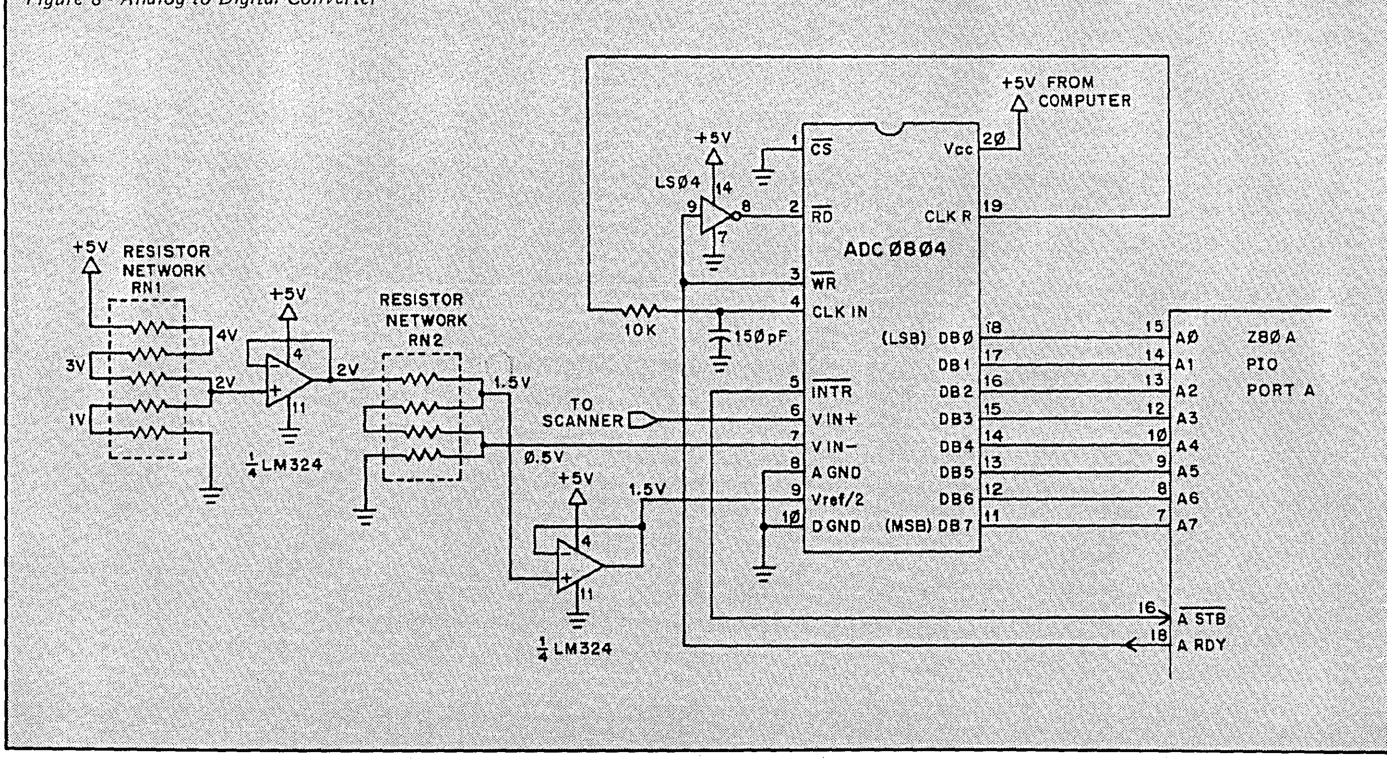
Voltage Dividers

To set these values, I use a couple of things called resistor networks and two op-amps connected as voltage followers. Here's why — Resistor networks come in DIP packages just like ICs — some of them have resistors connected from one pin across the package to the opposite pin, others have one end of all the resistors tied to a single pin. We'll be using the former type; check with an ohm meter to make sure.

Although the value of the resistors

(continued next page)

Figure 8 - Analog to Digital Converter



(continued from page 9)

in a package will be different from the advertised value (a package which says 100K may actually contain 102K resistors), the difference between one resistor and another in the same package is surprisingly small. Thus, resistor networks are a cheap way of making accurate voltage dividers. To create the desired reference voltages for the A/D, I've used two of these packages. Note that there are no values given on the resistors — you can use anything above 1K.

Resistor network RN1 is connected to divide the 5 V supply into 1 V steps. At the 2 V step, a voltage follower feeds into a second resistor network, which divides it into 1/2-volt steps so the desired 0.5 V and 1.5 V (through another follower) can be picked off. The voltage followers are used because if any significant current is drawn from any of the taps on the resistor networks, the voltage at that tap will change. The voltage follower holds its output at the value of the input without drawing any current from the input.

Accurate voltage references are for the general-purpose case. If you're calibrating your system (as we are for temperature), you can use a simpler circuit.

Temperature Sensor

Figure 9 shows the temperature sensor and analog input circuitry. The sensor is just a cheap transistor, and the op-amp circuit adjusts gain and offset.

The transistor is connected as a diode (a diode can be used, but a transistor works better) by connecting the base and collector as shown. The voltage between the positive input and ground is then the voltage across the PN base-emitter junction of the transistor.

At room temperature (25 C), the voltage across a silicon PN junction (i.e., a diode) is about 0.65 V. As the temperature changes, the voltage changes by about -2 mV/degree C (mV = millivolts). If we amplify this voltage change, we can convert it to a temperature reading.

We only want to amplify the voltage CHANGE, though, and not the offset value. The offset control shown allows removal of the transistor's normal junction voltage. The gain control allows us to adjust the amplification of the difference between the normal junction voltage and the change in voltage due to temperature.

Calibration

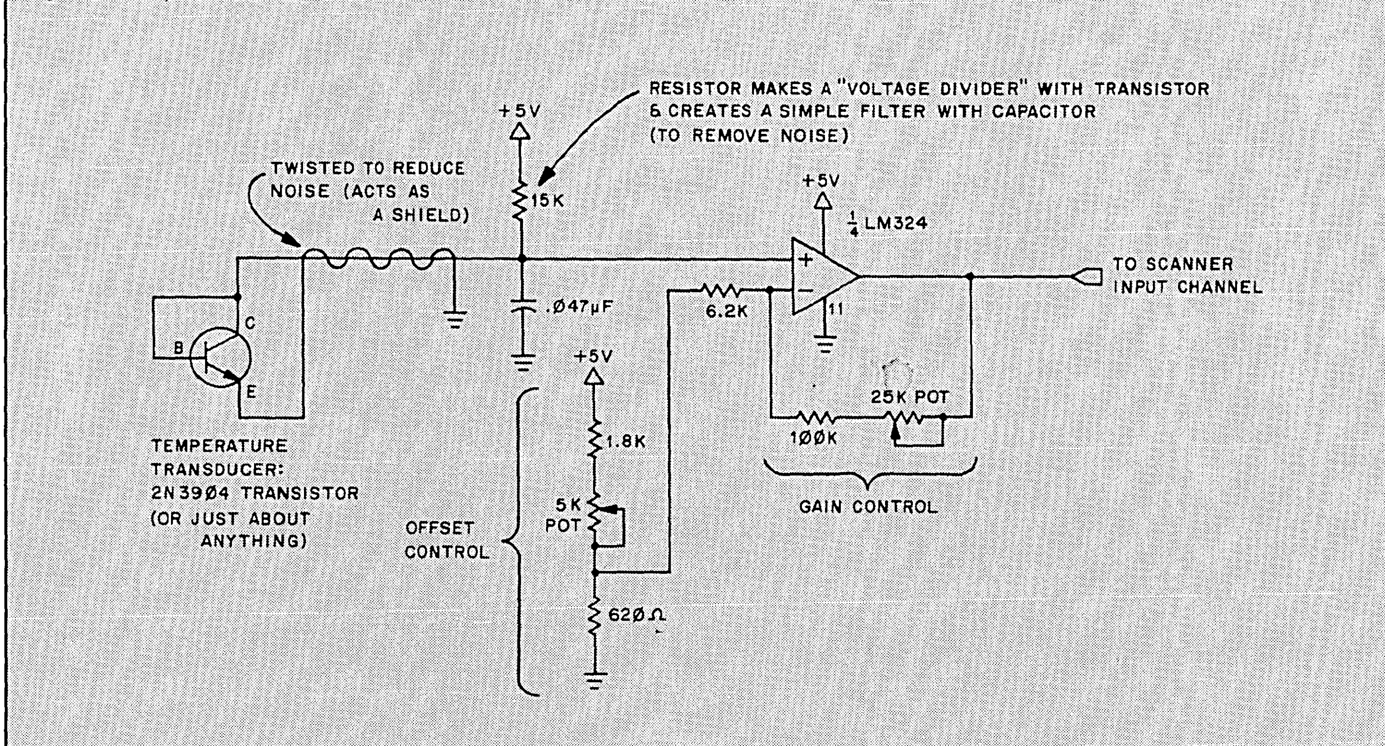
There are two places the system must be calibrated: on the board, and in the program.

The resistors shown for the gain control are chosen so the system will give results between freezing and boiling water. If the highest and lowest temperatures you want to measure are significantly different, you might want to modify the resistor values.

The Turbo Pascal program I've included (see Figures 10 and 11) has a record constant containing all the values for each probe, including name, location of screen output, slope, and intercept (assuming a simple linear model). I have one of the probes calibrated, and the rest have a slope of 1 and an intercept of 0 (which will just return the A/D converter value). Start with all the values of slope=1 and intercept=0.

Begin with the highest temperature you plan to use. (If you're going to immerse the probes, there may be leakage across the wires so you might want to use some sort of plastic spray.) Bring the probes up to that temperature, and

Figure 9 - Temperature Sensor Circuit



adjust the offset pot so the A/D reads slightly above zero.

Now bring the temperature down to the lowest value you see, and adjust the gain pot so the A/D reads slightly less than 255 (remember, the voltage goes UP as the temperature goes DOWN). Reheat the probes to the highest value, and make sure the A/D reads above zero. Keep changing the temperatures and adjusting the pots until both extremes give values inside the A/D's 0–255 count range.

Now go to the highest and lowest temperatures and write down the count as well as the actual temperature (I used a Fluke 52, but if you're using boiling and freezing at sea level, you don't need a thermometer). Since the probe isn't perfectly linear, I took a third value around where I was generally taking readings, and used that to determine the intercept. Figure 12 shows how to calibrate each probe using a linear model.

I suggest writing a program which records many A/D values and actual temperatures dynamically to form a curve for each probe. Linear interpolation can then be used for much more accurate results.

The Future & Thanks

Next time, I'll be playing with digital–to–analog converters (DACs). My eventual goal is to implement a digital control system.

And thanks to Brink Barr for teaching me most of what I know about electronics.

■■■

Figure 10 - Terminal Customization

```
{Include file TERMCAP.PAS. Creates character strings to invoke terminal
characteristics. Set up for Kaypro 84's. String constants wouldn't work.}
type
  termstr=string[6];
var
  cursor_off,cursor_on,dim,bright,reverse_video,normal_video,
  revdim,normal: termstr;

procedure TERMCAP;            {should make it a little easier to}
begin                         {customize for other computers.}
  cursor_off:=#27+'C4';
  cursor_on:=#27+'B4';
  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;
```

Figure 11 - Thermometer

```
program thermometer; {drives scanner, takes & displays readings}

{$I TERMCAP}
{Terminal customization file ala Unix TERMinal CAPabilities. NULL the
terminal-specific strings if your machine can't do them}

type
  probe_rec= record {holds calibration and display data for each probe}
             name:string[15];
             slope,intercept:real;
             xpos,ypos:integer;
             end;
const
  probe:array[0..7] of probe_rec= (
    (name:'Inside';slope:  1;intercept:0;xpos:20;ypos:5),
    (name:'Outside';slope:-0.6783;intercept:188.91;xpos:50;ypos:5),
    (name:'Basement';slope:1;intercept:0;xpos:20;ypos:10),
    (name:'Doghouse';slope:1;intercept:0;xpos:50;ypos:10),
    (name:'Hot Water';slope:1;intercept:0;xpos:20;ypos:15),
    (name:'Cold Water';slope:1;intercept:0;xpos:50;ypos:15),
    (name:'Wet Bulb';slope:  1;intercept:0;xpos:20;ypos:20),
    (name:'Dry Bulb';slope:  1;intercept:0;xpos:50;ypos:20)
    );
  A_CONTROL:byte=$22;     {Addresses in Kaypro 84 of extra PIO}
  A_DATA:byte=$20;        {(installed in issue #34)}
  B_CONTROL:byte=$23;
  B_DATA:byte=$21;

  INPUT:byte=$4f;         {01001111 PIO mode 1=input}
  OUTPUT:byte=$0f;        {00001111 PIO mode 0=output}
  INT_OFF:byte=$07;       {00000111 PIO interrupts disabled}
var
  KEY char;               {user input—to break out of loop}

function pause(delay_in_tenths:integer):char; {returns key value}
label done;
const delay_multiplier=300; {for tuning delay}
var CH:char;
    I:integer;
begin
      pause:='';
      for I:=1 to (delay_in_tenths*delay_multiplier) do
        if keypressed then begin
          read(kbd,CH);
          pause:=CH;
          goto done; {quick breakout when key is pressed}
        end;
done: end; {pause}

procedure ADC_delay(multiplier:integer); {delay for an A/D conversion}
var i,j:integer; {just kill some time here}
begin for i:=1 to multiplier do begin j:=0;j:=1;j:=2;end;end;

function sample(channel:integer):integer;
const average=10;  {Number of samples to take for noise reduction}
var sum,I,flush:integer;
begin
  {Remove value from last A/D conversion and start a new sample}
  for I:=1 to 3 do begin flush:=port[A_DATA];ADC_delay(1);end;
  sum:=0;
  for I:=1 to average do begin
    sum:=sum+port[A_DATA];
    ADC_delay(1);
  end;
  sample:=round(sum/average);
end; {sample}
```

(Thermometer program continued next page)

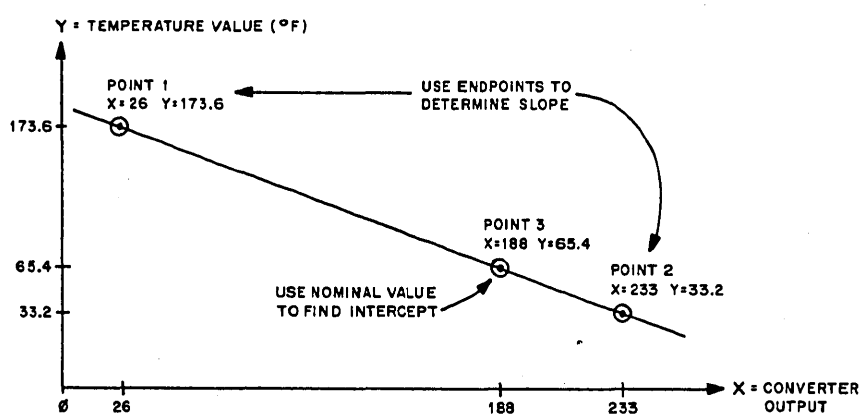
Figure 11 Continued

```
procedure display(channel,sample: integer);
  {adjusts sample with calibration data and displays value with label}
var result:real;
begin
with    probe[channel] do begin
        gotoXY(xpos,ypos); writeln(name);
        result:=slope*sample+intercept;
        gotoXY(xpos,ypos+1); write(revdim,result:4:1,normal,'F');
end;
end; {display}

procedure   channel_scan;
label   done;
const   wait=10;
var     J:integer;
begin
for J:=0 to 7 do begin     {The bottom three bits control the channel selection (inverted because of
                            the 74HCT00's).   The next three bits control the LED channel indicator.}
  port[B_DATA]:=byte(((not byte(J)) and 7) or (byte(j) shl 3));
  display(J,sample(J));
  KEY:=pause(wait);
  if KEY<>'' then goto done;
end;
done: end; {channel_scan}
{****************************** Main **********************************}
begin {initialize the PIO:}
  port[A_CONTROL]:=INPUT;     {Port A reads the A/D}
  port[A_CONTROL]:=INT_OFF;  {int is distinguished by its bit pattern}
  port[B_CONTROL]:=OUTPUT;   {Port B controls the scanner}
  port[B_CONTROL]:=INT_OFF;

TERMCAP; {initializes terminal-capability strings}
ClrScr; write(cursor_off);
KEY:='';
while KEY='' do channel_scan;
write(cursor_on);
end.
```

Figure 12 - Calibrating the Probe



1) $\text{SLOPE} = \dfrac{Y_2 - Y_1}{X_2 - X_1} = \dfrac{33.2 - 173.6}{233 - 26} = -0.6783$

2) TEMPERATURE = SLOPE * SAMPLE + INTERCEPT

$\therefore$ INTERCEPT = TEMPERATURE − SLOPE * SAMPLE

$= Y_3 - (-0.6783) * X_3$

$= 65.4 + 0.6783 * 188$

$= 192.91$

Swimming In The Relational Lake:

By Sandy Brabandt
6424 Sunnyfield Way
Sacramento, CA 95823

An Introduction To Database Programming, Part 1

I thought I knew a lot about data bases and a lot about dogs. Now, I have to confess, I've been d'based. (And who let Dr. Dobbs in here?) Don't miss this first episode starring that famous heroine, Dee Base. (I'm sorry, it only gets worse.)

It took 7 1/2 years to tabulate and analyze the data collected in the 1870 U.S. census. Due to significant population growth, it looked like the 1890 census would take even longer, and that the information would be obsolete by the time it was compiled.

Enter Herman Hollerith, who had invented a machine that would read census data from holes punched in cards, tabulate it, and then analyze it. Using this machine, the census bureau was able to release population totals only a month after the data collection was complete, and a more thorough analysis was finished in 2 1/2 years — one–fourth of the original time estimate.

This census tabulating machine was an important predecessor of the modern computer, and the punched cards were one of the first machine–readable statistical data files.

Data files still lie at the core of most computing — from a customer list kept by a newspaper carrier (on a Commodore 64) to the massive taxpayer records kept by the IRS.

But data files have always been considered a sort of ugly cousin to the more glamorous components of computer systems, the hardware and the programs. These get a great deal of attention, but data files, poor data files, are often created by haphazardly tacking data fields together as they're needed.

Files designed this way quickly become unwieldy and difficult. This means that programs become more complex in order to deal with the data. Updating a major system, once difficult, becomes impossible.

A Little Theory Goes A Ways

The database concept came about as an answer to these problems. The conceivers (or theorists) proposed the following simple guidelines for data files:

1) All the data used by an organization should be kept in one common base, and all systems should draw their data from this base.

2) The internal architecture of the data files should be kept independent from the applications. Each program should see only the data fields that it needs, and should navigate easily through the file structure. Changes to the files should have minimum impact on current applications.

3) The database should be easy to change and expand. You should be able to add and revise systems that use the currently existing files without having to tear apart the current file structures and start over.

4) There should be a structured design process for data files as rigorous as those for hardware and software. Data files should, in fact, be designed before the programs are written. In other words, system design should be data–driven rather than program–driven.

The theorists suggested several designs for file structures that would be easy to use and understand, and at the same time powerful and flexible enough to serve the needs of highly diverse systems. Among these was the Relational model for database structure introduced by E. F. Codd in 1970.

The Codd Model

In this model, data is kept in tables, which are related to each other via common data elements. This model gained support throughout the academic community for its flexibility and ease of use.

However, for a long time that support didn't extend into the business world, mainly because neither hardware nor software were powerful enough.

If you were working in data processing in 1970, you probably remember the card files, very similar to those invented by Herman Hollerith. The clean and elegant relational theory wasn't all that clean and elegant when the data came from trays of cards.

Even magnetic tape, which was a much faster, more accurate, and less bulky mode of data storage, was inadequate for relational databases, since the relational model required direct access storage devices. It wasn't until shops could afford large disk drives that they began using relational databases.

Even after the hardware was in place, it was years before software began supporting relational databases. Big Blue's DB2, a relational database product for mainframes, is just beginning to catch on. And, because it's an enormous resource hog, it's mainly used as a peripheral product to support end–user queries rather than as a day–to–day workhorse.

In general, mainframe–oriented data processing shops are just sticking their big toes into the relational database lake. Microcomputer users, however, have been happily swimming in the relational lake for a number of years.

Enter dBASE II

It all began with the introduction of a software product called dBASE II. Despite the fact that it was expensive and as buggy as an anthill, this product was so far ahead of anything else available at the time that it became an instant bestseller.

As micros acquired faster processors, faster hard disk drives, and more

RAM, a variety of dBASE II upgrades (such as dBASE III) and spinoffs (such as R:base 5000 and Paradox) hit the market.

Micro users had everything they needed to build sophisticated relational database systems. Unfortunately, most of them still lacked something. They didn't know how to use them well.

Dr. Dobbs & Relational Confusion

For example, let's look at a typical case of Relational Confusion: a hypothetical dBASE III user, a veterinarian named Dr. Dobbs (no relation).

Dobbs has decided to write a system to keep track of his practice. In between spaying and neutering his clientele, administering rabies vaccinations, and removing foxtails from noses, he's managed to read his dBASE III manual and do a bit of tinkering. He's eager to start programming.

A Bird In The Hand

So he sits down at his Eagle computer and cranks up dBASE to CREATE a data file. Adhering to traditional file design technique, he starts throwing data elements into it.

He starts with information about the pet owners. For each owner he wants to store a name, address, phone number, and current account balance. Then he needs information about the owner's pet: the type of animal, its name, breed, weight, and color.

Dobbs is a pretty sharp guy, and he immediately sees a shortcoming in this record layout: he's only defined fields for one pet per client, and many clients have more than one. He decides to make room in the record for more than one pet — in other words, to put a pet table within each client record.

After some consideration, he concludes that space for five pets per client should be adequate, and ends up with a record definition that looks like Figure 1.

Figure 1 - Table Within a Table

Clientname
Clientaddress
Clientphone
Clientcurrentaccountbalance

Pet1type	Pet2type	Pet3type
Pet1name	Pet2name	Pet3name
Pet1breed	Pet2breed	Pet3breed
Pet1weight	Pet2weight	Pet3weight
Pet1color	Pet2color	Pet3color
Pet4type	Pet5type	
Pet4name	Pet5name	
Pet4breed	Pet5breed	
Pet4weight	Pet5weight	
Pet4color	Pet5color	

Of course, most of his clients don't have five pets, which means he's wasted quite a bit of space. But he decides that, with his twenty–meg hard disk, he can live with some wasted space.

Things are going along fairly well until one day he goes to input the data for one client, Bowwow Kennels. Edith Bowwow owns 12 dogs and brings them all to Dr. Dobbs. Dobbs contemplates splitting the Bowwow dogs across three records, but quickly sees just how complicated the programming could get. He revises his scheme. He bases his files on the pets rather than their owners, and comes up with a one–record–per–pet design. It looks vaguely like Figure 2.

Now it takes twelve records to hold the information about Mrs. Bowwow and her dogs, and the information for Mrs. Bowwow is repeated all twelve times.

Dobbs is a little concerned about the extra space it will take to store all of that duplicate data, but he figures his hard disk can handle it. He also figures that his office staff can enter the Mrs. Bowwow data 12 times.

Soon after, Mrs. Bowwow calls to say she's moved, and gives them her new address. The office staff (after a minor revolt) has to change her address in twelve places. However, being human (can't Dobbs get anything right?), they miss making the change for the record holding information about her dog Phydeaux.

Later that month, she brings Phydeaux in for vaccinations, and as usual, they put the charges on her account. They then mail the bill to her old address.

The bill gets hopelessly mired in the Postal Service's forwarding routines, and Mrs. Bowwow never receives it. However, she does receive an eventual phone call from the office asking why she hasn't paid her bill. Since she has never even received the bill, she's outraged, and promptly takes all of her vet business elsewhere.

Dismayed Dobbs

Dobbs is dismayed. He bought The Computer to help his business, not destroy it. So, he decides to give in and hires a consultant to help design his system.

The consultant, Dolores Base (known to friends as Dee), examines Dobbs' two designs and announces that Dobbs has committed two database design crimes.

The first design contains a Repeating Group of data elements — the table–within–a–table. This kind of sub–table is hard to manage, wastes space if it's underutilized (if a client has fewer than five pets), and proves un–workable if it's inadequate in size (if a client has more than five pets).

The second design contains Redundant Data (Mrs. Bowwow's name and address repeated twelve times). Redundant data wastes space and effort by storing the same information over and over again, but more importantly, it causes an erosion of data integrity when it's not properly maintained (as when Mrs. Bowwow's address wasn't changed in Phydeaux's record).

Dobbs acknowledges the truth in everything Ms. Base is saying, but he wants to know one thing: how SHOULD his files be designed?

For the moment, let's leave Dobbs with his database and regress a bit.

To get to first base with Ms. Base, you first need to understand some basics about relational databases. Since dBASE III is the most commonly used relational database management system (DBMS), I'll use it for my examples.

Relational Database

Relational databases are composed of sets of tables. dBASE III keeps each table in a separate physical file, although other programs may not.

A row in a table corresponds to a

(continued next page)

Figure 2 - Data Base With Redundant Data

Pettype
Petname
Petbreed
Petweight
Petcolor
Petowner(clientname)
Clientaddress
Clientphone
Clientcurrentaccountbalance

(continued from page 15)

record in a traditional file, and a column corresponds to a field. Physically, a relational table resembles a traditional data file, but the ways they're used are very different.

Traditional file handling is record–oriented. The programs read the file a record at a time; each record is presented to the program as a string of data, and the program is responsible for mapping the fields in that string. This means that if the organization of data in the string is changed in any way, the mapping also has to be changed in every program that uses that file.

Relational table handling is more column–oriented. A program using a relational table will set a file pointer to the row it's interested in, and will then examine the data in that row by using the names of the column headers.

The column definitions are stored and managed by the DBMS. In dBASE III, for instance, the column header names, and the length and type of data element in each column, are stored at the beginning of the data file. This means that you can add and rearrange columns extensively without affecting your programs. The table tells the program where to find the data fields it needs. Thus, the order of columns in a table isn't important. Two tables are functionally the same if they have the same data, even though that data is in a different order.

Because the column mapping for a relational database is done by the DBMS, all relational databases are inextricably linked to the DBMS that created them. The DBMS provides the means to create a table definition and to access, update, add, and delete data in that table. This can, of course, be somewhat limiting.

Sharing Files

Most producers of relational DBMSs realize this and provide some way to make their tables readable by other languages or systems. For instance, IBM has provided a "pre–compiler" so that you can imbed statements from SQL (the data access language for IBM's DB2 databases) in programs written in several common mainframe languages.

dBASE III includes commands that will translate its table structures into other commonly–used data structures. In addition, due to dBASE III's popularity, many other software products can read dBASE III tables.

In exchange for this limitation, a relational DBMS will give you considerable power. As described above, it frees you from having to map data within your programs. It also provides the ability to do several operations that are critical to managing relational databases. Three of these operations — Select, Project, and Join — are so critical that a DBMS cannot really be considered "relational" without them.

The Select operation allows you to select certain rows within a table by specifying that certain data values must be present in specified columns. In dBASE III, there are many verbs that allow you to do a selection; all of them follow this syntax:

```
(verb)FOR(expression)
```

For example, the LOCATE verb allows you to select the first row that meets the specified criteria, as in:

```
LOCATE FOR LASTNAME="JOHNSON"
```

where LASTNAME is the name of a column in the table and "JOHNSON" is the value it must contain.

The LIST verb allows you to display or print a list of all rows that meet the given criteria:

```
LIST FOR AMOUNT>0
```

An alternative form of the Select function in dBASE III follows this syntax:

```
(verb)WHILE(expression)
```

The selection process in this case works like a standard DO WHILE structure: it tests the current row to see if the expression is true, and executes the verb if it is, then moves to the next row in the table. It continues to execute the verb and move the row pointer as long as the expression continues to be true, and stops once the expression becomes false.

The Project operation allows you to specify that you only want to deal with certain columns in the table, as if you were projecting a new table definition onto an existing table. dBASE III allows you to do this by specifying field lists as parts of commands.

For example, imagine that you have a customer table that contains a dozen fields, but you want to print a list that contains only two of these fields — the name and the current balance. You could use this command in dBASE III:

```
LIST CUSTNAME CURRBAL
```

This command will list the name and balance for every customer in the table. Note that the above command will continue to work in exactly the same way regardless of how many columns you add to the customer table, or what order you arrange them in. This is an example of how a column–oriented system allows your applications to become independent of the data file architecture.

The Select and Project operations can also be used together. If you wanted to make a list of customer names and account balances only when the balance is positive, you would use this command in dBASE III:

```
LIST CUSTNAME CURRBAL FOR CURRBAL>0
```

Select and Project operations aren't limited to verbs that do queries. Verbs that do updates, deletions, and mathematical calculations also use these operations. For example, the following command will sum all positive account balances in the Client table:

```
SUM CURRBAL TO TOTBAL FOR CURRBAL>0
```

where TOTBAL is the variable where the answer will be placed. This command will set all negative account balances in the Client table back to zero:

```
REPLACE CURRBAL WITH 0 FOR CURRBAL<0
```

As you can see, these operations are very powerful. With them, you operate on an entire table with a single command. (Of course, you can still access records one at a time.)

dBASE III has several other means of doing selections and projections, and other relational DBMSs use similar commands. By reducing a table to just the data you want, these operations contribute to the flexibility and navigability of relational file structures.

The third operation that a relational DBMS must have, the Join operation, works with two or more tables. This operation allows you to create a new table by joining other tables via common data elements.

It's these common data elements that are the core of relational database structure, for they determine the various ways that you can relate the tables to each other. (Thus the term relational databases.)

Back To The Future

For an example of how data elements control table relationships, let's return to Dr. Dobbs. His consultant, Dee Base, has offered the following advice: don't try to cram all of the data into a single table.

Create one table for the clients and another table for the pets. Then, place the pet owner's name in the Pet table, thus linking the pet back to the client. The tables will end up looking like Figure 3.

There will now be one row in the Client table for Mrs. Bowwow, and one row in the Pet table for each of Mrs. Bowwow's dogs, each of which will contain the name "BOWWOW" in the PET_OWNER column.

At this point, navigating through the files is easy. If, for example, Dobbs wants a list of pets for Mrs. Bowwow, he can use the Select operation against the Pet table:

```
LIST FOR PET_OWNER="BOWWOW"
```

If, on the other hand, he wants to know the address of Phydeaux's owner, he can first find Phydeaux's record in the Pet table:

```
LOCATE FOR PET_NAME="PHYDEAUX"
```

He can then use the contents of the PET_OWNER field (in Phydeaux's record, this will be "BOWWOW") to locate the client record in the Client table, and find the client's address.

Now he can add any number of pets for a given client, and if he changes the client's address, he need only change it in one place. Dobbs can see the logic of this, but he wonders what happens if, say, Mrs. Bowwow gets married and changes her name. Doesn't that mean he has to change her name in all of the records for her pets?

Ms. Base agrees and suggests that he assign unique numeric IDs to each client, and link the Client table to the Pet table via this numeric ID rather than the client name. The numeric ID would never change, eliminating the data integrity problem.

There would be additional benefits to this: first, a numeric ID would be shorter than the name and take up less space in the Pet table; second, a numeric ID would be unique, while a name might not be.

The new, improved tables look like Figure 4.

Dobbs can now execute the third major relational operation, the Join operation, on his files. For end-of-month reporting, he can create a new table by joining the two files into a third file by using the JOIN verb. To do this, he would open the client file in I/O area A and the pet file in I/O area B, and issue the command in Figure 5.

This will join the table in area A (Clients) with the table in area B (Pets), matching the Client ID field with the Pet Owner field, and creating a third table called "JOINFILE". This new table will end up looking like Dr. Dobbs' second design, where each client's name and address is attached to each pet record.

Thus, there will be twelve rows for the Bowwow dogs, one row for each dog, each containing Mrs. Bowwow's client information. All of the columns (from both tables) will be present in the new table. The two tables will be joined. Only records without matches — any client with no corresponding pet records or vice versa — will be left out of the new table.

Obviously the joined database contains a lot of redundant information (let me repeat that), but this redundant data isn't a problem. The new table need not be kept updated; it's simply re-created whenever it's needed.

Note that the Join command works in conjunction with the Select operation (selecting FOR CLIENT_ID = PET_OWNER). Other selection criteria can be added, if desired, to limit the records which are included in the Join.

In addition, the Project operation can be performed by adding a field list to the command. The new table will then only contain the columns named in the list.

Other relational DBMSs support a dynamic Join that does not actually create a third physical table, but rather, joins table rows dynamically as they are accessed, and makes the resulting joined row available to the application. (dBASE III does something like this with its SET RELATION command.) In any case, the Join operation removes some of the need for applications to navigation through related tables.

Because all of the relationships are defined by common data elements (as opposed to direct address pointers, as in hierarchical or network databases), it's extremely important that all desired relationships be defined up front. That way, the linking elements aren't left out.

In an extremely simple two-table database like the one we've been dealing with, most of the design can be based on intuition. It's easy to see that information about pets and clients should be stored in separate tables, and

(continued next page)

Figure 3 - Two Tables Linked By Client Name

CLIENTTABLE

CL_NAME	(Clientname)
CL_ADDR	(Clientaddress)
CL_PHONE	(Clientphone)
CL_CURRBAL	(Currentaccountbalance)

PETTABLE

PET_NAME	(Petname)
PET_TYPE	(Typeofanimal)
PET_BREED	(Petbreed)
PET_WEIGHT	(Petweight)
PET_COLOR	(Petcolor)
PET_OWNER	(Thenameofthepetowner)

Figure 4 - Two Tables Linked By ID Number

CLIENTTABLE

CL_ID	(UniquenumericID)
CL_NAME	(Clientname)
CL_ADDR	(Clientaddress)
CL_PHONE	(Clientphone)
CL_CURRBAL	(Currentaccountbalance)

PETTABLE

PET_NAME	(Petname)
PET_TYPE	(Typeofanimal)
PET_BREED	(Petbreed)
PET_WEIGHT	(Petweight)
PET_COLOR	(Petcolor)
PET_OWNER	(ThenumericIDofthepetowner)

Figure 5 - Joining Files

```
SELECTA
JOINWITHBTOJOINFILEFORA->CL_ID=
B->PET_OWNER
```

(continued from page 17)

that you put the client ID in the pet record in order see who the pet belongs to. However, for something more complex you may need a more structured design approach.

Ms. Base knows that Dr. Dobbs will quickly outgrow the two tables as he uses the computer more. So she encourages him to sit down with her and go through a formal database design process.

Next Time

I'll describe (with Dee's help) the detailed database design process and answer the following questions and more —

Will Dobbs finally understand his relations?

What are Ms. Base's designs on Dobbs?

Will Dobbs' business go to the dogs?

And what about Naomi?

(Editor's note: Don't blame me, I warned you.)

■■■

The Micro C Half–Fast Computer

By Larry Fogg
Micro C Staff

It's Snow Joke

OK, OK, before you wiseguys suggest that this is just another Micro C April Fool, consider this: You spend 3 to 6 months writing and compiling a super piece of code and what happens? Your 16 MHz 386 runs it so fast you miss the results. Even your 7 MHz 8088 hardly takes a breath.

I don't think that's fair. That processor ought to sweat a little, really appreciate the task you've given it. So, you have two options, write the code in BASIC or slow down the processor. Slowing down the processor is, no doubt, the least destructive of the two.

I spent this New Year's eve in a snow cave in the heart of Oregon's Cascade Mountains. Nestled among close friends (you'd be amazed how close your friends get when it's blizzarding outside) I enjoyed the gradual cooling and slowing of my mental functions as 1987 approached.

"What does this have to do with computer shenanigans?" you ask. Well, I usually figure that what's good for me is good for my computer. Short of hot tubbing, that is. And a slow system would have lots of advantages.

BASIC could take its rightful place alongside assembler. I mean, who cares whether a benchmark takes three days or four to run?

So into the freezer went my clone, but it refused to slow down. And typing was difficult in the dark. (Yes, the little light really does go out when you close the door.)

Slow Down A Minute

But wait! Why not do the speedup from our issue #31 with a slower crystal instead of a faster one? That way those programs which require a 4.77 MHz clock would still run but you'd finally have the upper hand on those pesty games. (Imagine playing Pac Man at 100 Hz.)

So I took a wheelbarrow full of slow crystals home at lunch and got to work. Low frequency electronics is fun stuff. You don't have to worry so much about crosstalk and the length of connecting wires. As a matter of fact, the jumper cables from my pick–up truck worked great for connecting the new crystals. (Unfortunately, the reverse doesn't work. Wire–wrap wire is worthless for jump–starting my truck.)

Imagine my disappointment when the PC would only slow down to 2.04 MHz (using a 6.12 MHz crystal). That's half–fast if you ask me.

The system did slow down, but it wasn't slow enough for me to keep up with the system. I tried using a standard IBM ROM in place of the Z–nix chip in my clone, thinking that surely anything with "IBM" printed on it would be slow. Still not slow enough. Even BASIC programs still beat me.

Back To Reality

There is a point to all this nonsense. What if you could slow down a computer enough to see it think? At SOG IV, Rick Prime talked about doing just that to a Z80 controller board he'd designed. Seems like he was able to completely stop the processor.

Think of the possibilities for debugging and education. Step the system through a single T–state and dig around with your voltmeter or logic probe. No longer could the skulking computer keep secrets.

What if you were invited to an Apple meeting? If you showed up with something slower than the Macintosh you'd surely be invited back.

Why Doesn't It Work?

If Rick's Z80 board went down to 0 Hz, why won't the PC?

The first problem comes from RAM. Dynamic RAM must be refreshed every few milliseconds. (And I thought I had a bad memory.) When the clock speed becomes so slow that RAM isn't refreshed often enough, the computer forgets just about everything. (I can't recall what it does remember.)

Rick's board survived by using static RAM. Once this type of RAM has been written to, it remembers the data until power down. It doesn't care what the clock rate is. ATs have battery backed static RAM, which retains system configuration information even when the computer's unplugged.

But there's a way around the refresh problem. In the PC, one channel of the 8253 programmable interval timer controls RAM refresh. The 8253 gets its timing reference from PCLK, a signal generated by the system clock chip. And, as luck would have it, our piggyback slowdown doesn't change PCLK. So RAM refresh continues at the normal rate no matter how slow the system clock ticks.

The Processor Is Too Fast

The second problem is the insurmountable one. A dip into the Intel Component Data Catalog and the NEC V–20 spec sheet shows a maximum clock period of 500 nsec for the 8088 and the V–20. There's the rub. These guys simply can't take time out. They can't lean back with a paper and a cup of coffee by a warm fire. That's why we have to do it.

Anyway, these guys are just too fast. (Never thought I'd say that about an 8088.) The 500 nsec period translates to a minimum clock speed of 2 MHz. That's pretty close to my 2.04 MHz.

According to the Zilog data book, our old friend the Z80 doesn't have a maximum period. Sure, signals have to rise and fall snappily but these requirements can be met by a nice short pulse every day or so. That leaves plenty of time to poke around and see what the processor's doing.

And So ...

Unfortunately, even though the processor will stop, any Z80 based micro will still be stuck with the RAM refresh problem. And unless you're willing to build a static RAM into your Z80 machine, or fool with a separate RAM refresh circuit, the 0 MHz machine will remain just another April Fool's folly.

■■■

Expert System

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Knowledge + Inference Engine

Expert systems have emerged from the ivory tower of Artificial Intelligence research, and are beginning to find work in businesses as well as laboratories. Many of the problems at this stage of the AI game lie not with programmers but with their ability to connect with experts from other fields. And this IS a problem.

Still, as long as internal memory continues to grow, and knowledge domains and the inference engines grow more powerful, the use of expert systems will spread far and wide.

Rule–based expert systems have been around since 1965, when Nobel Prize–winning chemist Joshua Lederberg joined with Edward Feigenbaum and Bruce Buchanan to develop a computer system which used heuristics (or rules) to analyze mass spectrometry data.

Their system, programmed in LISP and called DENDRAL, was a 15 man–year undertaking which eventually attracted a great deal of attention to expert systems. Since then, much has been attempted in this field, now known as knowledge engineering, which joins a computer–driven inference engine with a knowledge base (or domain).

What Experts Know

In a rule–based system, we "engineer knowledge" by combining an inference engine, which executes (or tests) rules, and a knowledge domain, which consists of the tests or rules.

The inference engine for DENDRAL was the programming language LISP. The knowledge domain contained the rules for analyzing mass spectrometry data.

A typical DENDRAL rule looked something like this:

IF the spectrum for the molecule has two peaks at masses X1 and X2 such that:

X1 + X2 = M + 28,
X1 – 28 is a high peak,
X2 – 28 is a high peak,
At least one of X1 or X2 is high,

THEN the molecule contains a ketone group.

It's the kind of rule a human expert in mass spectrometry might use to classify groups.

You should easily recognize how this kind of rule could be executed on a computer. Consider a Pascal function (or sub–program) for testing a rule.

```
FunctionRule1(m,X1,X2,hp,h:integer):boolean;
  begin
  IF(X1+X2=m+28)THEN
    IFX1-28=hpTHEN
      IFX2-28=hpTHEN
        IF(X1=h)or(X2=h)THEN
          Rule1:=true
        ELSERule1:=false
      ELSERule1:=false
    ELSERule1:=false
  ELSERule1:=false
  end; {rule1}
```

Our inference engine in this case is a Pascal program consisting of many similar functions — in other words, the inference engine and the knowledge base are one and the same. We modify (or broaden) the knowledge base by adding more functions (or rules).

This method, which doesn't separate inference engine and knowledge base, is primitive and cumbersome to use. Since knowledge must be hard–wired (or programmed) and recompiled, anyone other than a programmer will have trouble with it.

What's needed, of course, is separation of power: a programmer's part and a user's part. The programmer writes and perfects the inference engine which the user or "human expert" uses, but doesn't have to understand. The inference engine is sufficiently generalized that it can infer solutions to many different problems, so long as the knowledge domains are structured for the engine.

This separation of inference engine and knowledge base has led to modern *expert system shells*, inference engines which in their finest states include (at least):

- Explanation mechanisms
- Knowledge domain updating during execution
- Menu or natural language driven user interface

A few of these shells are beginning to be marketed, and may cost as much as $60,000. Any good system selling for under $1000 is a bargain.

I believe we're going to see, hear about, and use many expert systems (shells + knowledge domains) in the near future, so in this and my next article I'll try to present enough knowledge to enable you to understand and/or build an expert system.

Back To The Beginning

I've already shown you how a hard–wired rule would look in Pascal. For reasons which I hope will soon become clear, I think a good expert system can be programmed easily in PROLOG.

One key difference between Pascal and PROLOG is that PROLOG is itself an inference engine. (You could argue that Pascal is as well, but let's save that argument.) An IF > THEN process looks like this in PROLOG:

```
Plus(Num1,Num2,Sum):-
  Sum=Num1+Num2.
```

Here's the equivalent function in Pascal:

```
FunctionPlus(Num1,Num2,Sum:integer):boolean;
  begin
    IFSum=Num1+Num2
    THENPlus:=true
    ELSEPlus:=false
  end;
```

There's one way in which this function differs greatly from the PROLOG version. In PROLOG, you don't have to pass values for all the parameters. If we don't pass a value for one of the parameters, PROLOG will automatically "assign" it a value which will make the condition true.

In PROLOG, the IF > THEN is implicit. The rule in this case is:

IF (:-) the condition, Sum = Num1 + Num2, is true, THEN the rule, Plus(Num1,Num2,Sum), is true. Backwards, you might say.

(Or, "If you're trying to learn PROLOG and having difficulties, turn around, observe your learning, and proceed").

And a rule can have several conditions —

```
List_Sum(Num1,Num2,Sum):-
   Sum=Num1+Num2,
   Write(Sum).
```

THEN List_Sum(Num1,Num2,Sum) IF two conditions are met —

- Sum = Num1 + Num2
- Write(Sum).

So, a PROLOG "function" is in effect a rule, and a PROLOG program is a set of rules whose success depends on the completion (or satisfaction) of those rules.

If we defined a rule which could not be satisfied based on the current knowledge domain, the rule would fail. For example, if we wrote this program —

```
Main:-
   Num1=5,
   List_Sum(Num1,Sum).

List_Sum(Num1,Sum):-
   Sum=Num1+Num2,
   Write(Sum).
```

where main is equivalent to main program in Pascal and List_Sum is equivalent to a function, our inference engine (PROLOG) would begin by testing the first rule — Main.

The first condition of the rule, Main, is to add knowledge into the knowledge base of the program — Num1 = 5.

The second is to List_Sum(Num1,Sum), which is not only a condition but a rule. So we pass the parameters Num1 and Sum to the rule, List_Sum(Num1,Sum). We know the value for Num1, so we pass it. We don't know the value of Sum, so we pass a variable.

Then PROLOG tries to satisfy the List_Sum rule, whose first condition is to add Num1 and Num2. But we don't know what Num2 is, so the condition can't be met (is false) and the rule, List_Sum, fails.

PROLOG backtracks to Main, where the rule (List_Sum) is a condition, and since the rule failed, the condition can't be met so the rule Main fails.

This should be enough information to give you the flavor, so let's build a very primitive expert system by using PROLOG as an inference engine and a knowledge domain or set of rules about classifying animals.

A Little EXPRT

For simplicity's sake, our expert system won't do much (at least at first) — just determine whether an animal is a mammal or bird. Let's call this kind of expert system a rule-based classification system.

We'll hardwire the rules, but rely on user input to determine a rule's success or failure.

So let's suppose our knowledge domain consists of the rules in Figure 1, a PROLOG program we'll call EXPRT.

EXPRT's first rule is again Main, a place to start.

Once the rules are compiled, we run the program by entering "Main". In effect saying, infer what you can from the first rule (Main) in the knowledge base.

That rule (Main) states — IF condition, animal_is(Animal) is true, THEN Main is true. But condition, animal_is(Animal) is again also a rule, so the PROLOG inference engine must test (infer from) it, so it calls (or tests) the rule (or subprogram, or function) — animal_is(Animal).

animal_is(Animal) is a slightly more complicated rule, made up of three different SETS OF CONDITIONS, one each for determining whether an animal is a mammal, a bird, or neither.

PROLOG tries to fulfill the first set of conditions —

```
animal_is(Animal):-
   mammal(Animal)
```

Again, a condition is a rule, so in order to determine whether it's true, we need to investigate it. So we move to the rule —

```
mammal(Animal):-
   has_hair(Animal)
```

Another condition means another rule. So we move to the rule —

```
has_hair(Animal):-
   write("Does the animal have hair?"),
   readchar(Answer),
   Answer='y'.
```

OK, looks like we're getting somewhere. The first condition is write("Does the animal have hair?"). And what is "write"? Well, it's another rule (built into PROLOG) which is true when the text within its parameters is written to the screen. So, we test it by writing and by succeeding (presumably), satisfying condition one.

Move to condition 2, read-

(continued next page)

Figure 1 - Animal

```
main:-
   animal_is(Animal).

animal_is(Animal):-
   mammal(Animal),!,nl,
   write("Animal is a",Animal).
animal_is(Animal):-
   bird(Animal),!,nl,
   write("Animal is a",Animal).
animal_is(Animal):-
   Animal=unknown,nl,
   write(Animal,"Neither bird nor mammal").

mammal(Animal):-
   has_hair(Animal),
   gives_milk(Animal),
   Animal=mammal.

bird(Animal):-
   has_feathers(Animal),
   Animal=bird.

has_hair(Animal):-
   write("Does the animal have hair?"),
   readchar(Answer),
   Answer='y'.

has_feathers(Animal):-
   write("Does the animal have feathers?"),
   readchar(Answer),
   Answer='y'.

gives_milk(Animal):-
   write("Does the animal give milk?"),
   readchar(Answer),
   Answer='y'.
gives_milk(Animal):-fail.
```

(continued from page 23)

char(Answer), another built in rule, which is true when a character is read from the keyboard.

Move to condition 3, which is an identity test. Answer = 'y'.

IF Answer = 'y', all three conditions are true, which makes the rule, has_hair(Animal) true, which was a condition of the rule, mammal(Animal), which is now true.

Now we move to the next condition, gives_milk(Animal), which is a rule, and so on.

If Answer is anything but 'y', the rule has_hair(Animal) fails, which makes the rule mammal() fail. When mammal() fails, animal_is(Animal) fails. That is — the first rule for animal_is(Animal) fails; we have three.

So if the first fails, we try the second. If the second fails, we try the third, and so on, until we run out of rules or tell the inference engine we're satisfied with an answer. This relentless search for solutions is called backtracking, and it's the essence of PROLOG.

Backtracking & Cutting It Quits

Imagine an animal_is(Animal) path, or branch (see Figure 2).

Beginning with the top branch, rule execution proceeds from left to right. When a rule fails (or succeeds), we backtrack along the branch until we find a new rule to test, drop down one branch, and continue.

If you've been following closely, you're probably saying, "Yes, but if PROLOG keeps searching for alternate solutions, it will eventually decide that our animal is a mammal, and a bird, and is unknown."

Good point. We need to tell PROLOG, our inference engine, when we're willing to accept an answer. And we do this with the "!", called "The Cut". And it means exactly that, saying in effect, "if you're backtracking for more answers and you see me, CUT!, we'll call what you've got a take, and move on."

Look again at the code in Figure 1. In the rule, animal_is(Animal), if mammal(Animal) succeeds, it moves on, past the cut, past the "nl", which means newline, to the rule, write().

Write succeeds, and PROLOG backtracks to find more solutions, back over the nl, and smack up to the ! (the CUT). But PROLOG can't go past the cut, so it accepts its current knowledge, i.e. Animal is equal to mammal, and makes no more attempts to use the rule, animal_is().

PROLOG returns an A-OK as a condition to Main, and quits.

Notice that if mammal() hadn't succeeded, backtracking would have begun before the program reached the !, so the next set of conditions for animal_is() would have been tested.

To illustrate —

If rule, mammal is true:
animal_is(Animal) > mammal(Animal) > ! > nl > write > backtrack > nl > !

If rule, mammal is false:
animal_is(Animal) > mammal(Animal) > backtrack > animal_is(Animal) > bird(Animal)

If rule, bird is true:
animal_is(Animal) > bird(Animal) > ! > nl > write > backtrack > nl > !

Understanding how the "!" works is absolutely essential (in other words very important!) if you're going to understand PROLOG.

PC Diagnostics

Now that we have a working, albeit simple expert system, let's apply it (in principle) to a different knowledge domain, PC Diagnostics. See Figure 3.

As you can see, the systems are very similar. Primarily, we've changed the knowledge domain (rules) to cover questions about the PC, and again we're using PROLOG as the inference engine.

If you continued adding rules, you could create a reasonably complete (usable) inference system to handle PC diagnostics, without any more fanfare than what I've shown.

But the limitations should be apparent — the rules are hardwired, we

Figure 2 - Horizontal Branch

```
animal_is()>mammal()>has_hair()>
gives_milk()
animal_is()>bird() >has_feathers
animal_is()>neither
```

Figure 3 - PC Diagnosis

```
main:-
  pc_problem_is(Problem).

pc_problem_is(Problem) :-
  startup(Problem),!,nl,
  write("Problem is a ",Problem).
pc_problem_is(Problem) :-
  drive(Problem),!,nl,
  write("Problem is a ",Problem).
pc_problem_is(Problem) :-
  Problem=unknown,nl,
  write(Problem,"Neither startup nor drive").

startup(Problem) :-
  power_light_on(Problem),
  nothing_works(Problem),
  nothing_on_screen(Problem),
  Problem=startup.

drive(Problem) :-
  power_light_on(Problem),
  won't_boot(Problem),
  Problem=drive,
  read_write(Problem).

read_write(Problem) :-
  !,write("Have you tried booting another disk?"),
  readchar(Answer),
  Answer='y',
  write("Have you tried reselecting the drives?")
  write("i.e. A=B,B=A and rebooting?"),
  readchar(Answer2),
  Answer2='y',
  write("Did the switch work?"),
  readchar(Answer3),
  Answer3='y',
  one_drive(Problem).
read_write(Problem) :-
  both_drives(Problem).

one_drive(Problem).
both_drives(Problem).

power_light_on(Problem):-
  write("Is the power light on?"),
  readchar(Answer),
  Answer='y'.

nothing_works(Problem) :-
  write("Does anything work?"),
  readchar(Answer),
  Answer<>'n'.

nothing_on_screen(Problem) :-
  write("Is there anything on the screen?"),
  readchar(Answer),
  Answer<>'n'.

won't_boot(Problem):-
  write("Will the system boot?"),
  readchar(Answer),
  Answer<>'n'.
```

can't change one without recompiling the entire system (because we're rewriting the program). Good enough for programmers and hackers, of course, but probably not for users.

A better approach would be a program or set of rules which allows the user to enter rules (programs or databases of knowledge) during execution. This is expert system shell shine time. It supplies the user a friendlier interface for loading and inferring from rules than the PROLOG interpreter (or compiler) alone.

This approach loads sets (or databases) of rules during execution, allows updating of knowledge, explains itself, etc. You might think of it as a program (which it is), or set of rules, which executes another set of rules (or program).

Wrap Up

In my two examples, ANIMAL and PC, we saved time by bypassing the expert shell and using PROLOG to infer from a set of rules. So our system is primitive. There's no way to update the knowledge base, explain its reasoning, find multiple solutions, or a whole bunch of other useful things. It's primitive, but I hope instructive.

References

Fischler, M.A., and Firschein,O. *Intelligence: The Eye, the Brain, and the Computer.* Addison–Wesley, 1987.

Harmon,P., and King,D. *Expert Systems.* Wiley & Sons, 1985.

Winston, P.H., and Prendergast, K.A., eds. *The AI Business.* MIT Press, 1984.

■■■

By Cecil Stump
17733 205TH Ave NE
Woodinville, WA 98072

A Moonlit Software Business

About a month ago Cecil called; he was looking for a way to resolve some problems he was having selling Express. Shipments of the editor were delayed by fixes he and Laine were doing, and by the cumbersome way he was handling orders. The delays were upsetting his customers, his answering service, and him. Here's what happened.

Let's start this right off with a confession. I'm not really out "on my own," and don't intend to be in the foreseeable future. After eleven years with a major corporation I earn a reasonable salary, have a company vehicle, four weeks a year paid vacation, a pension plan, and medical and dental insurance. The company I work for is a leading manufacturer of medical equipment which means they're going to remain in business for many years. This all adds up to an overwhelming level of job security, in a job I like.

So why am I writing an "On Your Own"? Several years ago, being very disenchanted with the annoying selection of full screen text editors on the market, my brother Laine (yes, the one who writes "86 World") and I, decided we'd write our own. I could verbalize for hours on our software philosophy, but instead let's say that the result of our latent perfectionism and impatience produced a program we like to compare with a finely tuned sports car. VVVRRROOOMMM! How could we keep something like that to ourselves? In July 1985, TCI was born and the EXPRESS full screen editor was on the market.

The problem was with my "golden handcuffs" and Laine's desire to bring the entire Turkish nation out of the computer dark ages (ever hear the expression, blood from a rock?). Neither of us could seriously consider running a full time business. Part time, however, I would be able to handle the mail orders and he could take care of upgrades and bug corrections. Simple enough, huh?

Although things in general have worked out well, we've had to change our original business philosophy. I'd like to share with you some of the experiences and difficulties specific to a part time operation.

The Customer's Perception

One big consideration in a part time business is how to address your customers' expectations. They know you only through an advertisement in a magazine. You're offering them a product and they expect no less service than if you were a major corporation. It's not their problem you already have a full time job which takes up a ten hour day. Nor is it their problem that you don't have a full time order department, shipping department, and twelve customer service representatives. In spite of your part time status, if you intend to run a successful business you must give the same service as a big company.

Phone Orders — Mail Orders

After a good product, the most important thing to a customer is speedy delivery. Our original thinking on taking orders was that, as a part time operation, we would take orders only through the mail. Without bankcard merchantship we would have to wait for a check anyway. The drawback here was that it takes up to a week for orders to reach us, another ten to fourteen days for the checks to clear the bank, and yet another week for the products to reach the customers. For many customers, this was not acceptable.

There are a number of ways this time can be reduced. One is to ignore the advice of your accountant and ship the product before the check clears the bank. This is not without risk, and when you get a bad check you may as well kiss the bucks goodbye. It's a tough decision. You either get a reputation for slow service or you eat bad checks. A number of major companies have decided their reputations are worth the small percentage lost.

The recommendation here is to do whatever you must to get bankcard merchantship and take phone orders. Since we began taking phone orders and accepting bankcards, eighty percent of our business is done this way.

In Defense Of The Post Office

How you ship the order makes a big difference in delivery time. After trying the various shipping methods, we would rate them as follows:
First Class Mail — Our choice
UPS — Also good
Third Class Mail — Never

Yes, the Post Office has a credibility problem. However, First Class mail within the continental US is always airmail. Our packages are about eleven ounces, and our experience is that delivery time is in most cases equal to or better than UPS, and the rates are roughly the same. The major advantage of First Class mail for us is that the postal service comes by every day, UPS could care less.

Answering Services

Once you get your bankcard merchant status and take phone orders, another problem arises. Taking phone orders when you're not home can be tricky. One solution is an answering service to take phone orders for you. They can ask all the right questions and take the customer's order. You can just sit back and fill the orders as they come in, right? I don't want to rain on your parade, but I think I feel a thundershower coming on.

Answering service operators are not stupid. The problem is that the operator who's taking orders for software is also taking messages for doctors, lawyers, and even orders for steak knives advertised on late night television. It's impossible for anyone to know every buzzword for every profession. So it's no surprise when I get an order for a Seepy Matey on a Kageltoo. (Translation: CP/M 80 on a Kaypro II.)

The real problem occurs when they don't let the caller know they're an answering service. One prospective customer called several times during the business day, expecting to find me in. On each occasion, he left a message that he'd call back at a certain time. Needless to say, this customer was furious that I was never in to answer a simple question. His last message said to forget it (quite emphatically).

Have we dumped the answering service? Not entirely. However, I've had another line installed into my office at home, which my wife answers when she's available. When she has to leave, an answering machine takes messages and also gives the caller the number for the answering service (for orders only). This is working out much better.

Time Efficiency

The key to getting your product out the door rapidly is minimizing the time it takes to process an order. For example, we used to personalize each EXPRESS so it displayed the purchaser's name. We also printed computer generated invoices.

The problem is setup time. The time to fire up the computers, set up the printer for invoices, run the invoice program, format the disk, copy the files over, and personalize the disk; all added up. On the average, it took about fifteen minutes per order.

To speed things up, we decided to stop personalizing disks and have begun handwriting invoices. The disks can be created in advance and packaged with the manual. Processing an order now amounts to handwriting an invoice and taking a package off the shelf. We package three orders in the time we used to spend on one.

One real time gobbler is the non-standard order. We can create an EXPRESS for almost any CP/M 80, CP/M 86, or MS-DOS computer. We've even ported EXPRESS over to an Apple II (it works great).

Unfortunately, the Apple II doesn't use a normal disk controller. They do it with software, which means you can't read or write Apple disks on a non-Apple computer.

To make matters even worse, not all Apples have the same CP/M disk format, and the one we used wouldn't read the CCP on a warm boot. We had to make a special version of EXPRESS which didn't overwrite the CCP so CP/M wouldn't crash when EXPRESS ended. I could go on and on, but the point I'm trying to make is that you may be wiser to turn some business away, especially if it consumes time and effort without promise of significant return.

Communications, A Case History

Last fall, after EXPRESS 2.2 for CP/M 80 had been out about two months, we discovered a couple of significant bugs. We held up shipping until we'd found and fixed them.

This was a good decision, but we made a big mistake. We invested all our effort in isolating the bug and didn't notify the customers that their orders had been delayed. Most were patient, but others were not.

Using business math, if you add delayed shipment to an answering service operator who can only take orders, the result is a very irritated customer. (And rightly so.) A valuable lesson learned the hard way is that when there will be a delay in shipment, you should immediately let the customer know what, why, and when. Most customers are very understanding but you have to give them something to understand.

(As for the bugs, we found they were limited to the CP/M 80 version, and the owners of the affected copies either have already or will shortly receive patch information.)

Summary

You don't have to go totally out on your own to market a product. It's possible to do it on a part time basis, but you need to remember that a good product will not guarantee customer satisfaction. You must also provide good customer service in the form of speedy delivery and good communications. Regardless of your part-time status, your customers will expect no less service from you than they would from a major corporation.

Being a part time small business has its advantages. Lower overhead means you can put a lower price on your product than the guys who have to pay for a huge building and hundreds of employees. You don't have to have a huge cash flow, or worry about whether you're going to sell enough to put food on the table. However, your part time business is limited by your time. You'll never hit the big time with a part time business. But it can be a good stepping stone if things work out well, without major financial risk if they don't.

Choosing full time versus part time operation is a tough decision. It depends on your product and the marketplace. More importantly, it depends on you. If you're not a risk taker, but are willing and able to devote lots of spare time, a part time operation may be right up your alley.

■■■

The Rochester Data Dynatyper

By Don McClimans
108 Boardman St.
Rochester, NY 14607

An Idea Whose Time Went Quickly

Two issues ago I asked for projects which didn't pan out. This is one of the responses. It's particularly interesting because occasionally I still get requests from misguided persons who want to hook their computers to Selectrics (and maybe writing articles). For these people I know where there are some red, white, and blue plastic plungers. Cheap!

In 1978, two engineers at Litton Industries, Bill Murphy and Bill Smith, had a great idea for a new type of computer printer. A box containing 52 plungers could be mounted on top of an electric typewriter. It would be connected to a computer, and would mechanically "type" documents.

When Litton Industries proved to be uninterested in the idea, Smith and Murphy took it to Bob Giese and Tim Maloney, of RG Engineering. Bob agreed to finance a new company, and the four engineers founded Rochester Data Inc., with Bob as the president. They expected the Dynatyper to be the first in a long line of ingenious products. They were going to revolutionize the computer industry. They were going to become rich.

But first, they had to make the damned thing work.

It took almost two years to work the kinks out of the Dynatyper, and when they were finished, they had a product that could type almost as fast and evenly as the typewriter it was attached to. Its fame was instantaneous (along with 1/4 page ads in Byte), but within a year those automatic fingers were walking headlong into the beginning of the Japanese computer attack. That was the year Epson began shipping low price, high speed, reliable, dot matrix printers.

Development

The Dynatyper is simple in principle, and appeared, at first, to be a product with a future. It was portable, inexpensive, and required no modifications to the typewriter. So the Dynatyper didn't effect service agreements on the underlying typewriter. Dynatyper sold for under $500, which compared very favorably with $3500 for the other letter quality printer of that era — the Diablo daisywheel printer.

Simple in principle, yes. But before production could begin, engineering problems had to be solved. Metal plungers worked well with the solenoids, but destroyed the typewriter keys. No problem. To save the typewriter keys, they changed to nylon plungers bonded to a metal shaft. But during heavy use, solenoid heat softened the nylon. They tried Delrin (another type of plastic) but it expanded too much as it warmed, sticking tightly inside the metal shafts. So they built new plunger molds.

Meanwhile, they were adjusting space between plungers because of differences in keyboard layout. Length of the plungers was also a problem.

Different Keyboards

The IBM Selectric, with its sculptured keyboard, required a different length plunger than a flat keyboard. The designers eventually color-coded the plungers. The short plungers (red) struck keys on the top and bottom rows. The medium length plungers (white) struck the middle two rows, and the longest plungers (blue) struck the return, backspace, etc.

Thus, customers could match plungers to specific typewriters.

How hard should each plunger press? Not a simple question. Typewriters differ in the amount of force required to actuate a key, so the solenoid voltages and pulse lengths had to be set by the customer.

Feeling The Heat

Solenoids, drive transistors, and power supplies all overheated.

"We had a lot of min–max (optimization) curves going, and as you would slide down one of the curves, you would slide up another one," Giese said.

"You could change the pulse shape to reduce heating in the solenoids, for example, but doing so caused more heat in the drive transistors."

Since there was no feedback from the typewriter to tell when a character had printed, delays were generated by software. Different typewriters required different delays, and even on one typewriter, different delays were required for character printing, and for each "mechanical" function, such as tab, backspace, and return.

The driver software could handle just about any delay on any key and customers could precisely adjust the delays for their own typewriters.

Interfaces

There were more computers back then (now it's two, clone and non-clone). Rochester Data wrote printer interfaces for the TRS-80, Apple, PET/CBM, OSI, Northstar, Heath, etc. Most of the drivers were distributed on tape. (Remember cassette tape?) The drivers translated the ASCII character codes into X-Y keyboard positions and fired the solenoids.

When combined with ancient wall-chiselers like Electric Pencil, the Dynatyper was slow, cranky and generally abominable. But it was revolutionary.

Production

Initial design problems are the easy part. Just ask any production engineer. Once you've made one or two, you have to be able to make a hundred.

Rochester Data didn't have enough

room in their development offices, so they set up manufacturing on the second floor of an old warehouse in an industrial area.

"It wasn't a fancy place," Giese recalls.

"In the winter, the wind would rattle the windows, and the elevator was so creaky you didn't know whether it would make it to the top or not."

They hired a production manager to oversee the new facility. In fact, they hired 3 or 4, because none of them seemed to work out.

One didn't bother to pay the electric bill and no one else was aware of the problem until the electric company called to inform them that their power had been turned off.

Interestingly enough, everything was fine; the lights were on, the machines running. It turned out they'd been running the entire production line off the meter belonging to the neighboring parachute manufacturer. (One must carefully avoid having a falling out with a parachute manufacturer.)

Another production manager kept running out of parts. Giese says, "We'd joke with him, saying, 'Bob, maybe we should put the whole operation on a large semi, and drive it around town. When you run out of material, we'll just drive over and pick them up.' "

"The time and effort involved in producing the Dynatyper made graduating from engineering school, or getting the professional engineering license, look like a snap," Giese recalls.

Customer Complaints

With production, there were new problems — customers. But customers weren't all alike, and flexibility often turned into disaster. Customers would turn up the voltage until the solenoids cooked. Or they'd remove the cover and flip the whole assembly upside down. It wasn't hard to imagine those teeny little parts lying scattered over the table when the phone rang: "Where do the blue plungers go?"

Many customers were homebrew experimenters, setting up their systems in whatever corner was available. The typewriters–now–printers were often placed on rickety card tables. When "return" was pressed, the table would continue to wobble long after the carriage had stopped, making the first few characters on the next line look like they'd been typed during an earthquake. Extra delays had to be added to accommodate card table users.

Rochester Data employees became experts at diagnosing Dynatyper failures. By looking at which characters were printing incorrectly or not at all, they could tell which bit in the ASCII character code was inverted, or which drive transistor had failed. "After 2 or 3 years," says Giese, "I could look at faulty output and know exactly what was wrong with it, and I still can to this day." (Not an especially marketable skill.)

Contrary to advertising claims, reliability of the early units left something to be desired. A major electronics manufacturer inquired about using Dynatypers to test their keyboard's reliability. "It was a joke," says Giese, since the keyboards would easily outlast the early Dynatypers.

After improving the service life of the Dynatyper we began running into reliability problems with the typewriters. Typewriters weren't designed for continuous use. Even Rochester Data had problems getting a reliable typewriter — they tested the newly manufactured Dynatypers on an office Selectric. They were careful not to let the Selectric's service contract expire, since the typewriter died every few weeks. They practically had a standing order with IBM service.

But slowly, they solved the problems. They improved reliability, designed better driver and testing software, and wrote clearer manuals. They began shipping products, sales reached $250,000 a year.

Then the roof fell in. The Japanese introduced dot matrix printers designed to sell in very large volumes at low cost. Seeing the printer output on the wall, Rochester Data closed its doors in late 1981.

Giese lost his $70,000 investment, but all suppliers were paid, and paid–for orders were shipped. Warranty and service for Dynatypers continued for more than a year after the company closed, a remarkable feat for a small start–up company.

The remains of Rochester Data are stored in the loft of a large barn 25 miles south of Rochester. A few bare PC boards and power supplies, and several hundred coils, sit gathering dust, along with thousands of red, white, and blue plastic plungers.

■■■

A Bleeping PC

By Larry Fogg
Micro C Staff

Counter Culture Inside The 8253

Larry's understanding of smart ICs has never been so sound. In fact, we can usually tell when he's between tech calls from the "music" which warbles forth from his corner of the office.

This article started life as an attempt to add voices to the PC's severely limited sound capabilities. I never did come up with a choir on my clone, but the other information which surfaced made the whole thing worthwhile.

The Search Begins

How does the PC generate sound? The first source I dug out was Peter Norton's *Inside the IBM PC*. When will I ever learn? Norton is a "Techno-Tease." His explanations never go quite far enough (like most of my dates). He covers lots of topics, but none thoroughly. And in the case of the speaker circuit, some of his information is just plain wrong.

The whole story comes out in IBM's *Technical Reference*, Intel's *Component Data Catalog*, and in Sargent and Shoemaker's *The IBM Personal Computer From The Inside Out*. I can't praise the latter book highly enough. If you're at all interested in the innards of the PC, buy it.

The Intel 8253 programmable interval timer (PIT) controls the PC's voice. This little gem contains three counter/timers. When a timer is active, each input clock cycle decrements the timer's count by one. In the PC, all three counters get the same input clock. It's generated by dividing PCLK (4.77 MHz on a standard PC) by four.

Each counter also has a gate input which controls the counting. On the PC, the gates on counters 0 and 1 are tied high. This means that counting is always enabled for those counters. The signal controlling counter 2's gate comes from PIOb bit 0. (Intel calls its 8255 a "programmable peripheral interface," but I can't say "PPI" with a straight face. So let's stick with PIOb to designate port b of the parallel I/O chip.)

Programming The 8253

The control port for the 8253 lives at 43h. We send a byte to that port to set up any of the three counters. Bits 7 and 6 (the two most significant bits) of that byte determine which counter we're programming. Think of them as a two-bit binary number with a value between 0 and 2.

Bits 5 and 4 configure the reading and loading of data at the data port. 00b latches the counter value into a storage register so that it may be read on the fly. 10b sets up a read or load of the most significant byte only, 01b does the least significant byte only, and 11b does least significant first, then most significant.

The timers operate in six modes. Bits 3 through 1 select modes 0 through 5.

In mode 0, output stays low throughout the countdown, then it goes high. The count (and the low output) begin as soon the timer has been programmed. Mode 1 is the same as mode 0 except that the countdown (and low output) begin as soon as the gate input goes high.

Mode 2 outputs a low pulse for a duration of one clock cycle when the count reaches one. It then reloads the counter and repeats.

Mode 3 outputs a square wave. It does this by setting its output high and decrementing the count by 2 at each clock cycle. When the count reaches 0, the counter gets reloaded and the output goes low. After double stepping down through the count again, the output goes high and the whole process repeats. The resulting square wave will be symmetric if the initial number is even.

So, loading the counter with 16 means that the timer would output a complete cycle or square wave (high and low) in 16 input clock cycles. Essentially, it divides the input clock by the counter's value.

Mode 4 is a software-triggered strobe which goes low for one clock cycle when the count reaches zero. Counting begins as soon as the counter is loaded. Mode 5 works the same way but starts counting in response to a rising gate signal. It generates the same low strobe at zero.

Modes 0, 1, 4, and 5 are single event modes. They do not automatically repeat like modes 2 and 3.

Finally, the least significant bit of the control byte tells the 8253 whether it will be fed data in binary (bit 0 = 0) or binary coded decimal form (bit 0 = 1).

Rolling it all together, a write of 54h (01011000b) to the control port means that timer 1 (01b) should expect to get only the least significant byte of its count (01b). It will operate in mode 4 (100b) and receive data in binary form (0).

Back To Sound

Counter two of the 8253 drives the speaker, but its output doesn't go there directly. First it gets ANDed with PIOb bit 1.

This gives us three ways to generate sound. First we can turn on the PIO bit and program the 8253 with various frequency square waves using mode 3. (See Figure 1.) Bit 0 of PIOb plays a role here also. It supplies the gate input for counter two, and must be high for the counter to count.

Or use mode 0 to program the 8253 to go high after one count. (The timer thinks of zero as 65536, so use one – it's pretty close to zero.) Now turn PIOb bit 1 on and off to run the speaker. (See Figure 2.)

The third possibility involves both techniques at once. After programming the 8253 to generate a tone, modulate that tone by turning it on and off with the PIO bit. When the PIO's frequency is slow, we'll get a "beep, beep, beep ..." from the speaker. Interesting things start to happen when you increase the PIO's frequency. As it approaches the 8253's rate, the two begin to beat against each other to produce some fun results. (See Figure 3.)

Incidentally

We've only talked about one of the three counters. It turns out that the other two are doing a lot more than just beep.

One of the nice things about "intelligent," or programmable chips like the 8253, is that they rescue the processor from lots of system drudgery. Consider RAM refresh. Pretty boring job. But all the 8088 knows is that at power up it programmed channel 1 of the 8253 to take care of refresh.

Since then, every 15 microseconds channel 1 has been using mode 2 to send a pulse to the 8237 DMA controller. The 8237 then refreshes a portion of memory. And all this time, the 8088 has cooked along with whatever it needed to do, if anything.

Last, but not least, the output of counter 0 nudges the 8259 programmable interrupt controller every 65536 clock cycles. That's about 18.2 times per second. Mode 3 is used here. The 8259 in turn interrupts the processor and tells it to execute the time of day interrupt.

I did get a bit sidetracked from my original intent here, but diversion is the spice of life. And it's great fun to discover intelligence inside the PC. We'll be taking a look at more of this intelligence in upcoming issues.

■■■

Figure 1 - Sound From The 8253

```
PROCEDURETimerSound;{seriesofnotes(hightolow)usingthe8253}

CONST                      {theseconstantsusedinfigure1-3}
  TimerData0=$40;          {dataportaddressforcounter0-timeofday}
  TimerData1=$41;          {counter1-memoryrefresh}
  TimerData2=$42;          {counter2-speaker}
  TimerControl=$43;        {portaddressforcontrolbyte}
  PIOb=$61;

VAR
  I,J:Byte;

BEGIN
  Port[PIOb]:=Port[PIOb]OR3;      {turnon2LSbitstoactivatespeaker}
  Port[TimerControl]:=$B6;        {setupcounter2;
                                   LSBthenMSB;mode3;binary}
  FORI:=0TO10DO                   {thisloopcontrolstheMSBofthecount}
    BEGIN
      FORJ:=0TO255DO              {thisonecontrolstheLSBofthecount}
        BEGIN
          Port[TimerData2]:=J;    {loadthecounter}
          Port[TimerData2]:=I;
          Delay(1);               {generatethistoneforashorttime}
        END;
    END;
  Port[PIOb]:=Port[PIOb]AND252;   {turnoff2LSbitstoturnoffspkr}
END;    {TimerSound}
```

Figure 2 - Sound From The PIO

```
PROCEDUREPIOSound;   {similartoTimerSound,butusesthePIObit}

VAR
  I,J:Integer;

BEGIN
  Port[TimerControl]:=$91;   {setupcounter2;LSBonly;mode0;binary}
  Port[TimerData2]:=1;       {wantoutputhighrightawaybutcan'tuse0}
  Port[PIOb]:=Port[PIOb]OR1;          {turnonbit0toenablecounter}
```

Figure 2 Continued

```
  FORI:=1TO500DO                       {lowIgiveshighfrequency}
    BEGIN
      Port[PIOb]:=Port[PIOb]OR2;       {turnonbit1}
      FORJ:=1TOIDO;                    {thesedelayscontrolthefrequency}
      Port[PIOb]:=Port[PIOb]AND$FD;    {turnitoff}
      FORJ:=1TOIDO;
    END;
END;  {PIOSound}
```

Figure 3 - Modulated Sound

```
PROCEDUREModulatedSound;{generatesanotethenmodulatesit
                         withnotes(lowtohigh)fromthePIO}

VAR
  I,J,K:Integer;

BEGIN
  Port[PIOb]:=Port[PIOb]OR3;      {turnon2LSbitstoenablespeaker}
  Port[TimerControl]:=$B6;        {setupcounter2;
                                   LSBthenMSB;mode3;binary}
  Port[TimerData2]:=0;            {loadthecounter}
  Port[TimerData2]:=20;
  Delay(1000);                    {letthetimersoundgoforawhile}
  FORI:=400DOWNTO1DO              {nowaddthePIOmodulation}
    FORK:=1TO15DO                 {controlsrateoffrequencychange}
      BEGIN
        Port[PIOb]:=Port[PIOb]OR2;  {turnonPIObbit1}
        FORJ:=1TOIDO;  {delayloopsdeterminethefrequency}
        Port[PIOb]:=Port[PIOb]AND$FD;  {turnitoff}
        FORJ:=1TOIDO;
      END;
  Port[PIOb]:=Port[PIOb]AND252;   {turnoff2LSbitstoturnoffspkr}
END;
```

S–100 Bargains

By Alexander Wright
Viasyn Corporation
3481 Arden Rd.
Hayward, CA 94545

A Low Cost S–100 Card Cage And Motherboard

CompuPro/Viasyn is selling a number of its boards at steep discounts on a support–it–yourself basis. With that in mind, I thought some of you dyed–in–the–wool S–100 addicts might appreciate some details on these bargains. Here's Alex's piece.

CompuPro is offering a combination card cage and twelve–slot motherboard for $139. When combined with an XT/AT type power supply, it forms the basis for a very low cost S–100 system. This type of system uses regulated bus voltages, and therefore may require some modifications to standard S–100 cards.

I'll be describing the card cage and motherboard, how to hook it up to a power supply, and how to modify your S–100 boards if necessary.

Regulated Bus Voltages

When the S–100 bus was first designed, the only practical way to provide enough power for up to 22 cards was with a large unregulated power supply. Regulation was done on the cards, since at that time large, reliable switching power supplies didn't exist.

Fortunately, the power supply situation has improved. Switching power supplies are now very reliable and affordable, so it makes sense to use regulated voltages on the bus and eliminate the on–board regulators. The IEEE 696 working group is currently trying to add a regulated bus voltage option to the S–100 Standard.

Modifying existing S–100 boards to use regulated bus voltages is relatively easy, and I'll talk more about this later.

Card Cage/Motherboard

The motherboard has twelve slots and active termination at both ends. The layout uses interleaved ground traces between signal lines to minimize crosstalk and insure high speed operation. Active termination has long been recognized as superior to passive termination for active signal lines.

A ten–pin connector is used to connect a power source to the motherboard. A pre–wired mating connector is included so you don't have to search all over for one. Two–pin connectors are located on the motherboard for a reset button and a power indicator LED.

The card cage is constructed from 18 gauge steel and finished in gold. The motherboard comes already mounted to the base of the card cage assembly and above it, there are plastic card guides.

Hooking Up a Power Supply

For use with regulated bus voltages, the most economical power supply I know of is the type normally used for XT/AT type clones. These are priced under $80 for a 135–150 watt unit. This size supply is more than adequate for an average set of S–100 boards, and has power to spare for disk drives.

The motherboard comes with a pre–wired cable and mating connector. The five red wires are the +5 V lines, the gray wire is the +12 V line, the purple wire is the –12 V line, and the four black wires are ground. Be sure to check the documentation which comes with the supply so you can match these up correctly.

You'll probably have to remove the connector on the wires coming from the supply and solder the wires directly to the cable from the motherboard. Be sure to insulate these connections. The ten–pin connector plugs into J3 on the motherboard, and the single red wire plugs onto J14. This extra wire helps distribute the +5 V to the other end of the motherboard more efficiently.

XT/AT power supplies include pre–wired connectors for powering 5 1/4" drives.

Reset Switch & LED Connections

At one corner of the motherboard are two connectors for a reset switch (J2) and a power indicator LED (J1). Hook a momentary, normally open SPST switch between the two pins of J2. You can hook any standard LED between the two pins of J1. The current limiting resistor is built into the motherboard. The cathode (usually the flattened side) of the LED goes to the pin closest to the "Reset" legend on the motherboard. You may want to use a combination LED/pushbutton switch.

To modify a board for use with regulated bus voltages, you should only have to remove the regulators and connect the input to output pads together with a piece of insulated 24 AWG wire.

Figure 1 shows some common regulator pinouts which may be useful in identifying the input and output connections. Be sure to conspicuously label this board for use with regulated bus voltages. I recommend red labels reading "+5 VOLTS!" placed where the regulators used to be. Plugging a modified board into an older S–100 system would be hazardous to its health!

Some boards have circuits that detect power failure by looking at the bus voltages. Component values may have to be changed to make these circuits work correctly. The CompuPro System Support 1 is one such board.

To modify this board, remove resistors R1 and R3. Install a 390 ohm 1/4 watt 5% resistor in location R1. Install one end of a 470 ohm 1/4 watt 5% resistor into the hole at location R3 closest to R5. Standing the resistor up on the board, solder the other end to the leg of

R5 closest to R3. (Of course, the regulators must be removed and jumpered as well.)

Finally

CompuPro is offering other low cost hacker boards (like a CPU 286 for $249). With a little ingenuity and an XT/AT power supply, you can put together a very inexpensive S100 system.

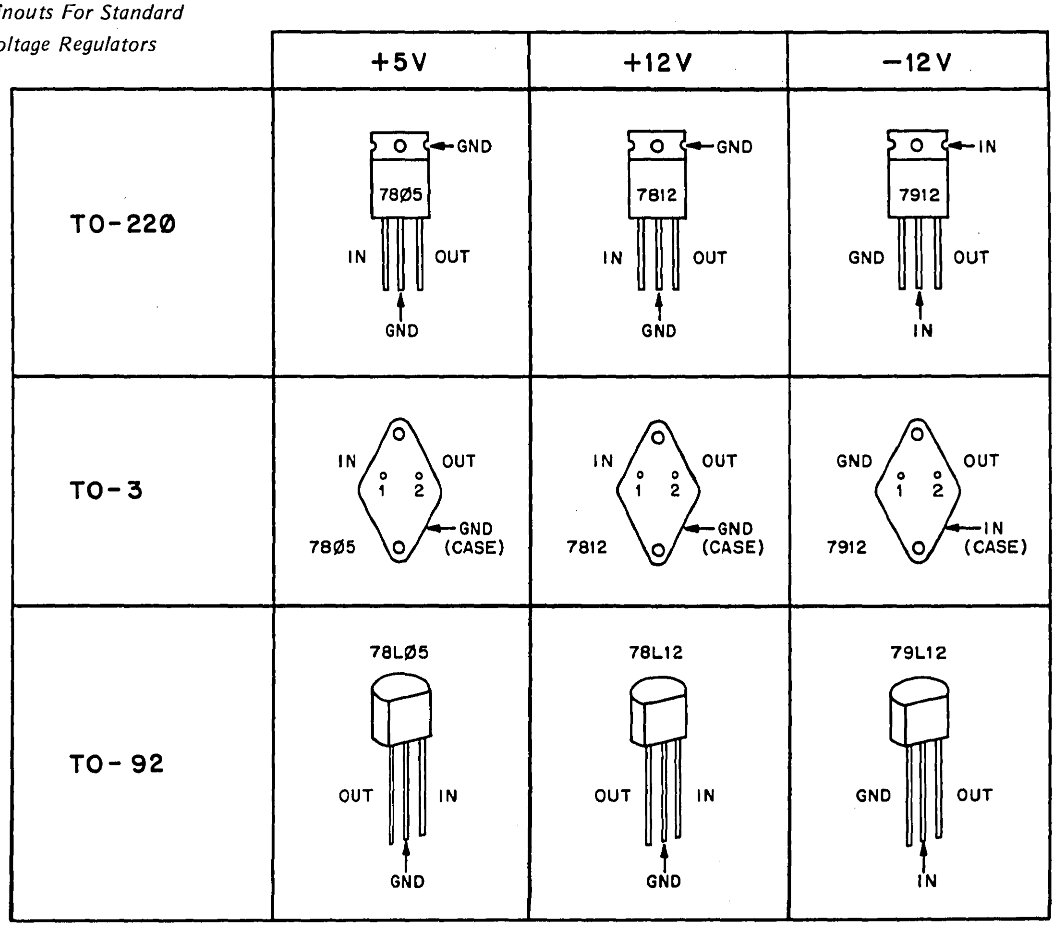
You can also get more information about the board from the CompuPro vendor-support conference on BIX (the BYTE information exchange).

Sources

Viasyn Corp.
26538 Danti Ct.
Hayward, CA 94545–3999
415–786–0909

■■■

Figure 1 - Pinouts For Standard Voltage Regulators



Instructions:

1. This guide does not apply to adjustable regulators.
2. In all types remove regulator & jumper input to output (being careful not to connect to ground).
3. Reference: Nat'l Semiconductor Regulator Handbook.

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:

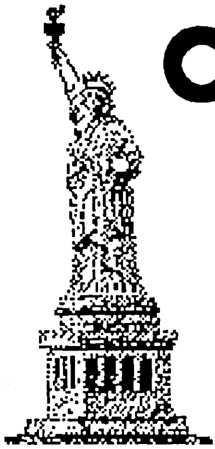
Introducing a new Shareware version of our best seller, the ConIX Operating System. Included is our regular distribution software less the Archiver, On-Line Manual, Menu source code, and some satellite utilities.

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

By Mark J. Boyd
Computer Science Department
University of North Carolina at Asheville
Asheville, NC 28804

Z80 Modula–2: Turbo VS FTL

Not all Modulas are alike. In this column Mark details the differences between Turbo Modula–2 and FTL Modula–2.

Modula–2 is a language for the eighties. It has almost all the virtues of Pascal, but few of its limitations. Ada is the other language in this category. Ada is more powerful, and is now available for PCs, but only PCs with megabytes of add–on memory. Modula–2 is small and elegantly simple.

Modula–2 "feels" like an extended Pascal, but with new features which place it in an entirely different language category. Modula–2 is much more powerful than Pascal, yet retains much of Pascal's simplicity and elegance.

Two Z80 Modula–2 Compilers

We now have two Modula–2 systems available for the Z80: FTL Modula–2 from Workman and Associates, and Turbo Modula–2 from Echelon, Inc. FTL Modula–2 actually originated in Australia — David Moore, whose company is Cerenkof Computing, developed and packaged it.

Turbo Modula–2 was developed by Borland International, but Echelon is marketing it. There is an 8086 version of FTL, and I assume Borland will eventually have an 8086 version of Turbo Modula–2. *(Editor's note: Don't hold your breath.)*

FTL Modula–2 was apparently derived from a PDP–11 minicomputer version of Modula–2. Turbo Modula–2 shows its Turbo Pascal heritage. Both are excellent software development systems, but with different strengths.

Both Turbo and FTL Modula–2 use an integrated editor/compiler system which allows highly interactive coding and debugging. They both support essentially standard Modula–2 (Turbo has some problems in this regard). Both compilers also offer extensions, but the extensions are quite different.

FTL includes its own assembler for creating assembly language modules. Turbo supplies a program to convert M80 (and similar) .REL files to Turbo's link format. Neither system includes a real debugger, but both will trap and identify run time errors (e.g. real overflow). Both systems produce fast, relatively compact code.

I consider these to be the two best systems available for developing Z80 application programs to run under CP/M. Now let's look at the pros and cons of the two systems.

Turbo Modula–2 Pros:

Turbo Modula–2 is small; the entire package requiring under 200K. I can run it comfortably using a single DS/DD drive.

Turbo Modula–2 is fast — a Sieve of Eratosthenes, compiled and linked to a .COM file, runs almost three times as fast as the same program compiled by Turbo Pascal! In fact, the Sieve runs almost as fast on my 5 MHz Z80 as on my 8 MHz 68000 Atari 1040ST (compiled by TDI Modula–2).

Turbo Modula–2 is well documented. The users guide includes 200 pages of tutorial material and 280 pages of "reference directory." The reference directory is an "alphabetical directory to Turbo Modula–2's standard identifiers, extensions, library procedures and modules, and reserved words." Very nice.

Turbo Modula–2 has several extensions which make it similar to Turbo Pascal, and therefore easier to learn if you already know Turbo Pascal. You could be programming in Turbo Modula–2 fairly quickly using nothing but the manual as a reference. Unfortunately, you wouldn't really be programming in Modula–2, but rather in an enhanced Turbo Pascal. Modula–2 is not an extended Pascal and programming in Modula–2 is quite different from programming in Pascal. If you get Turbo Modula–2 you should also consult a good Modula–2 text and spend some time learning the language.

Turbo Modula–2 Cons:

On the other hand, Turbo Modula–2 is not a general purpose language. It is a language for producing application programs which run under CP/M. The .COM files it produces consist of a run time support package linked with the Modula–2 modules. There doesn't seem to be any way to produce code that doesn't load at 100h.

Turbo Modula–2 is not standard Modula–2. Even without its extensions, it uses non–standard names for several important procedures (even for a standard library module). Every module I brought over from other Modula–2 systems had to be modified slightly to work with Turbo.

If you use the extensions (and they are very nice extensions so it's hard not to use them) the resulting code is not portable. It wouldn't even be practical, given the nature of the extensions, to modify the code for another system. You would pretty much have to start from scratch.

FTL Modula–2 Pros:

FTL Modula–2 is standard Modula–2. All of the standard Modula–2 code that I've copied from texts compiled without modification.

FTL Modula–2 includes source for all its library modules. The code really helped me understand how the modules work and how I could incorporate them into my own code.

FTL produces small .COM files. They can be compiled and linked to run anywhere in RAM (or ROM) with any desired initial value for the stack pointer.

This system, with its assembler, is all I need for systems programming on the Z80. In fact, it's adequate for any kind of Z80 programming. Like C or assembler, you can write anything the computer is capable of running. Unlike C or assembler, FTL also helps you create reliable, maintainable, code.

FTL produces fast code, faster than Turbo Pascal's, but not as fast as Turbo Modula-2's code. The code produced is very compact. FTL's PRIME.COM is only one fourth the size of the Turbo Modula-2 .COM file. I've read several articles which said, "Compiled Modula-2 code will always be bigger and slower than compiled C code." That's not true for FTL Modula-2.

FTL Modula-2 Cons:

On the other hand, FTL's user manual is not up to the standard set by Turbo. It's 200 pages in a plastic ring binder that's really too small for it. It's a pain to use.

I like the writing style of this manual, it's witty and informative, but it does assume some sophistication. Definitely not for the neophyte.

The best documentation in this package is the source code for the standard library. If you don't already know Modula-2, studying this code is difficult. There's some tutorial material, but you need a good text on Modula-2 to go with it. For experienced programers, I recommend Ford and Wiener's, *MODULA-2 A Software Development Approach.*

I often refer to the definition files for the library modules because the manual doesn't really cover them. To be fair, the user's manual suggests you print these out for reference, but it's much easier to use Turbo's reference directory. Since these modules are normally kept in a library, I have to leave the FTL editor and run the library program to access this info. I have the printed copies somewhere on my desk, but I can never find them without hunting for several minutes.

Benchmark Comparisons

The first benchmark I used (Figure 1) was supplied with Turbo Modula-2. It is named prime.mod, and is a fairly standard Sieve of Eratosthenes program, running ten iterations. I modified the Turbo code to Modula-2 that would compile with a standard system. I imported the I/O procedures from the Terminal module. I compiled, linked, and ran it with both FTL and Turbo on my 5 MHz Z80, and with TDI Modula-2 on my 8 MHz 68000 (Atari 1040ST). All compiles and links used a RAM disk. I later modified the code to Pascal (it was closer to Pascal than standard Modula-2 to start with) and checked it out with Turbo Pascal on the Kaypro.

Note that FTL compiles faster, Turbo links faster, and Turbo runs almost as fast as the 68000 version. I've seen similar benchmarks for IBM PC Modula-2's running in the 15 to 18 second range. (How does Borland get a Z80 to run with a 68000?)

FTL isn't as fast, though it's still a lot faster than Turbo Pascal or the PC Modula-2s. However, it's really nice for producing compact code. Note how slow the interpreted m-code is, and how small its code is! This is the normal operating mode for Turbo. The compiler produces an intermediate code file (remember p-code for UCSD Pascal?) which is then interpreted. I thought my system had crashed when I first ran the m-code for prime.

Turbo suffers a big size penalty on small programs because it creates .COM files by linking its "run time system" with the compiled (native) code. FTL and TDI both produce code for .COM files directly (no "run time system"). Turbo's size penalty is not a problem for large programs; in fact the run time system approach may become an advantage when the program gets large. However, the run time system must start at 100h, so this method effectively prevents using Turbo Modula-2 for systems programming.

To test disk I/O and performance without a RAM disk I wrote a simple

(continued next page)

Figure 1 - Sieve of Eratosthenes Benchmark

SYSTEM	CODE-SIZE	COMPILE	LINK	RUN
TDI-1040ST	3166bytes	9sec.	19sec.	5.7sec.
FTL-KayPro	2367bytes	2.5sec.	6.5sec.	12.0sec.
Turbo-KayPro				
(native)	9984bytes	8.25sec.	3sec.	6.9sec.
(m-code)	136bytes	5.5sec.	N.A.	100sec.
Turbo Pascal (Kaypro)	8498bytes	1.5sec.	N.A.	18.9sec.

Figure 2 - File Copy Benchmark

SYSTEM	.COM FileSIZE	COMPILE	LINK	RUN
Turbo				
(native)	18K	20sec.	31sec.	80sec.
(m-code)	N.A.	17sec.	N.A.	82sec.
FTL	12K	12sec.	68sec.	99sec.

(continued from page 37)

file copy program. This program used InOut routines to read and write a 24K text file. See Figure 2 for the results.

Once again we see that Turbo has a speed advantage and FTL has a size advantage. When running these tests, it was obvious from the drive activity that FTL used a smaller disk buffer. I could have changed FTL rather easily, but not Turbo. Then FTL would have had the speed advantage.

By way of comparison, WASH, a utility program that uses a large buffer, did the same copy in 16 seconds. These compile and link times are typical for small programs.

Note that the m-code is almost as fast as the .COM file here. That's because the CPU is mostly just sitting around waiting for the disk drives.

In terms of interactive program development, the time to exit the editor, enter the compiler, detect an error, and return to the editor is very important. Both systems do well in this regard, but Turbo has the edge. Without a RAM disk, and with a 4K Modula-2 file with an error in its declarations, it took 16 seconds to complete this edit, compile, edit cycle. FTL took 22 seconds for the same cycle.

Turbo's compiler always quits at the first error, FTL gives you the option of continuing to find other errors. Turbo's compiler continues its work after the file is edited. FTL's compiler is restarted after the edit. Both systems provide an excellent interactive environment for program development. FTL feels slightly faster, perhaps because of faster compiles. On the other hand, the whole Turbo system will fit comfortably on my 250K RAM disk and thereby work faster than FTL. FTL requires a lot more space.

The Editors

I spend most of my programming time bumping along in an editor. Turbo's editor is quite similar to the Turbo Pascal editor, a real plus for many people, but I think the FTL editor is much more powerful.

FTL's editor allows you to have up to three files open at once and to move blocks of text between them. It also allows you to define elaborate macros as you work, and it automatically saves and reloads those macros each time you edit.

The FTL editor acts much more like a minicomputer editor than a WordStar subset. It uses the WordStar command set, with some extensions, and has a neat window for selecting other functions.

You do pay a price for all this. The FTL editor is a 34K .COM file. That's a lot larger than the 0 – 15K hunk Turbo swaps in and out when going from the editor to the compiler and back. This is one of the reasons the edit–compile–edit cycle takes longer (for short files) on FTL.

Also, the FTL editor can only handle files up to about 20K. Normally this isn't a problem, but it can get annoying when you want to edit two or three large files simultaneously. (For $30 you can purchase the source of the FTL editor. Get it if you get FTL Modula-2 and then check out my editor mod article in Micro C.)

The Libraries

Libraries of modules are an inherent part of the Modula-2 concept. FTL Modula-2 comes with several libraries; Turbo comes with only one. However, Turbo's librarian allows you to create your own libraries and to modify the system library.

FTL comes with LSWEEP, which helps you get files out of its libraries (including all that lovely source code), but it has no program for creating or modifying libraries. Since Modula-2 generates lots of little files, you can rapidly run out of directory entries on a CP/M system. I really think FTL should provide support for user libraries. Turbo's librarian is excellent.

FTL Extensions

Both Turbo and FTL offer some extensions beyond standard Modula-2. Turbo's extensions are more extensive but I'll start with FTL.

FTL's most useful extension is that sets can have up to 1024 elements and the bottom element does not have to have an ordinal value of 0. Thus, unlike Turbo or standard Modula-2, you can have sets of CHAR. Another useful extension is the initialization of variables and arrays when they're declared. Neither of these extensions is a big deal. They don't accomplish anything that can't be done in standard Modula-2, but they make programming a bit easier.

Also, FTL's compiler is very forgiving about semicolons. I like this feature, but it means that FTL accepts code which a standard Modula-2 compiler would reject.

Turbo Extensions

Turbo has several extensions. The first group makes the language more like Turbo Pascal. These include READ, READLN, WRITE, and WRITELN procedures, along with the Texts library module and the u=l (upper case = lower case) compiler option. These extensions allow Pascal-like I/O and single case source code. They also prevent compilation of most standard Modula-2 code.

READ and WRITE apparently interfere with Read and Write, the standard character I/O procedures, so they've been renamed ReadChar and WriteChar. STORAGE also replaces Storage — I don't know why.

I know Borland wanted to make it easy for Turbo Pascal programmers to come over to Turbo Modula-2, but I feel these extensions really mean that Turbo Modula-2 isn't Modula-2. This is not because of what they added, but rather because of what they took away in order to add the Pascal I/O and the u=l option.

The second Turbo extension really is an extension. Modula-2 is a good software engineering language, but it lacks exception handling. Turbo Modula-2 adds this feature. It's nicely done and it doesn't interfere with the standard. Of course, if you use it, the resulting code will not be portable.

This is a somewhat moot point in view of the non-standard I/O. But I/O can be modified to work on a standard Modula-2 system without extensive redesign. If you use the exception handling extension, it will probably take major redesign to convert the code to a standard Modula-2.

There are several other minor, but useful extensions. Strings are more fully supported, and Turbo Modula-2 allows multidimensional open arrays. The linker supports a nice scheme for overlays, like Turbo Pascal's, but more powerful. These are useful extensions, and code using them can be converted to standard Modula-2 fairly easily.

Bugs

Turbo had a serious problem with code that used long arrays of characters, for instance, the scanner code in issue #34 of Micro C, p. 62. I'm not sure what's happening, but I assume part of the code is getting overwritten.

The .MCD files (m-code) were just like bad C programs — no error messages, no nothing, it just didn't do anything. No way to debug it, it wouldn't even print a "HELLO" placed right after the begin. Turning on the run-time array bounds checking didn't help. When I produced .COM files, they "sort of" worked, but they crashed and I was unable to debug them using Turbo's run time error location.

The same code (not counting the Turbo modifications) works fine with the FTL and TDI compilers. I'd like to have a reserved word capitalization preprocessor for Turbo, as I have for TDI (I incorporated this function into the FTL editor), but I'm currently stumped by this bug.

I've had no problems with the FTL compiler or linker, but the editor occasionally runs out of memory before it should have. It doesn't handle this very gracefully, and I've lost some work. No longer do I edit several files in succession without exiting. I'm also careful to consider the sizes of the files when I open multiple buffers.

Conclusion

If I were only going to do application programming on a Z80 system, I'd probably use Turbo because of its speed and its exception handling. For systems programming and code portability, FTL is the only choice.

FTL is also the system of choice for those of us who like to get inside our software tools and tinker with them. It is possible to do this with Turbo, but without the source for the system library modules, it's much more difficult.

To put it another way, Turbo Modula-2 should replace Turbo Pascal for serious Z80 programmers. It really fills the same niche, and it allows development of larger, more sophisticated software systems.

FTL Modula-2 could replace not only Turbo Pascal but also C and assembler. I think FTL is a better system for hackers and programmers who want a true Modula-2 compiler for code portability. FTL has become my universal Z80 programming system.

Sources

FTL Modula-2
Workman and Associates
1925 East Mountain St.
Pasadena, CA 91104
818-791-7979

Turbo Modula-2
Echelon, Inc.
885 N. San Antonio Rd.
Los Altos, CA 94022
415-948-3820

■■■

IN THE PUBLIC DOMAIN

By Stephen Leon

200 Winston Drive
Cliffside Park, NJ 07010

Shocks At SIG/M

Major changes at SIG/M as Steve resigns his position as the group's disk editor. Steve tells us what he expects in the CP/M public domain arena. If he's right, the outlook is bleak.

Never a dull moment in the world of public domain software! The recognition by SIG/M that the rest of the world was shifting away from CP/M (Micro C, Issue #32) was applauded by most of us as the only way to save the SIG/M library from the fate of most other CP/M libraries. We thought that for SIG/M to survive, it had to expand to include both CP/M and PC-DOS/MS-DOS material. We wanted source code, hacker type material, and certainly no beggarware.

After the initial decision was made at a SIG/M meeting to go into other areas, a few of the diehards insisted that, since SIG/M was a part of the Amateur Computer Group of New Jersey, the ACGNJ Board should have the final say. The Board voted unanimously to let SIG/M expand.

Some people, unfortunately, never give up, so the battle continued. When it reached a point where, for me, it was no longer fun being disk editor, I did what every volunteer has the right to do — I said forget it!

Keith Peterson, who runs the CP/M SIG on Genie, agreed to take over the job. I certainly wish him well — but I have my doubts about what will happen.

Being a SYSOP on one of the services can pay very well. SYSOPs receive a percentage of what SIG users pay for access time and rumor has it that some of them may be making close to six figures. I suspect that what goes into the library may well be determined by what sells, and to my mind that means the end of SIG/M as we now know it.

(Editor's note: On the other hand, if a program's popular on Genie, there could be a good reason for it.)

Great New SYSOP Software

Many Micro C readers have been calling our BBS (201-886-8041), and we've enjoyed chatting with you whenever possible. We now have 160 megs of storage on a network consisting of an AT and 2 Turbo XTs. All of you are welcome to call. We have no preregistration requirements, and you get full access to the system the first time you're on.

In addition to the SIG/M and PC/BLUE libraries, we have plenty of CP/M and PC-DOS public domain software. We also have a very large selection of Amiga software. The daily time allocation is currently 90 minutes.

With more than 1500 files and 12 separate directories, I got ambitious and wrote a dBASE3+ program (compiled with Clipper) which automates the handling of BBS directories. The program keeps a master data base of all files on the system. Within each record, you can select up to ten subdirectories for that program. As new files are uploaded to the system, you can read in the upload file directory, edit the entries to indicate placement, and then create the various directory files. All told, it takes the program six minutes to generate my dozen directories for the complete system and various categories in alphabetic order. It also generates a reverse chronological directory with a cutoff of any date.

It seems to work rather well. The only problem I have is that it's so easy to generate new directories, I sometimes generate them a couple of times a day. The compiled program, the database files, and the dBASE source code can be downloaded from our board as BBDIR.ARC.

Bulletin Board Changes

If you've called the Micro C board lately (503-382-7643), you will have noticed a shift to commercial bulletin board software (PC Board). We've also shifted — from the public domain RBBS to Dan Doman's dBBS. RBBS, from the Capitol PC User's Group, is probably one of the finest pieces of public domain software available (PC/BLUE Volumes 249 and 250).

We were running two copies of RBBS on one machine using Quarterdeck's DESQview. It worked fine — but even with a Turbo Board the speed left something to be desired. We therefore looked at both PC Board and dBBS as true multi-user alternatives. We settled on dBBS on almost a toss of a coin. The deciding factor was that the dBBS people were in New York near us, but the PC Board people were all the way out in Salt Lake City. In addition, it was much simpler to convert from RBBS to dBBS than to PC Board.

The Last Shock Of The Column

The other day I sent a check for $35 to register my copy of ProComm (PC/BLUE 269). Once before I've paid for beggarware, but then I sent Tom Rettig a check for a copy of dGenerate. I treated that as a direct purchase. Here, I've been using ProComm, having discovered the value of its Ymodem Batch mode, without the slightest guilt or hesitation. After all, I have no legal obligation to pay for the software (notwithstanding some of the statements in some of the beggarware).

ProComm is just such a good piece of software, it made me feel good to send them a check — so I did!

No New SIG/M Releases

With all the problems at SIG/M, quite naturally, there are no new releases to report. Whether there ever again will be new releases is something we can only speculate on.

For those of you looking for CP/M software, there are still many gems in the library. You won't find a dBASE clone, but I know several companies that are doing quite well with DIMS (Dan's Information Management System — SIG/M 61. Updated on SIG/M 248). CP/M users are still running Bulletin Board Systems, and RBBS (SIG/M Volumes 249 and 250, the same numbers as in the PC/BLUE library) is still a great package.

MEX (218, 219, 220) and IMP (272, 273, 274) meet your modem needs. I get several requests a week for volumes 800 and 801 — the Commodore 128 CP/M sampler. (Just send me two blank disks and stamps for return postage, and I'll send them to you.) So if you're still running CP/M, be sure to get yourself a copy of the SIG/M catalog (or download it from our BBS) and discover how many people went out of their way to give you some really fine public domain software. It may be dead in the future, but their contribution lives on.

New PC/BLUE Releases

Lets discuss the good and the bad in the new releases first. NYWord version 2.1 on PC/BLUE 272 is a great word processor. Mark Adler's program is as good as any commercial word processor on the market. He's also most conscientious about keeping up the quality of the program. If you need a word processor, try his first before you go for the high priced spread.

On the other extreme is PC/BLUE 277 — AMTAX86. Don't get me wrong. AMTAX86 is a good piece of software. It will calculate your 1986 taxes. Their 1985 program was also quite good, and I used it as a double check against the commercial software I purchase to do my taxes.

However, AMTAX86 is a demo! It really irks me to see a tax program that says you must first pay a fee if you expect to print a tax return. The "shareware" release doesn't print the tax forms!!!!

Back to the good stuff. Volume 270 contains DIS86 — a full screen disassembler which will also handle V–20 and 80386 code. Volume 271 has Horse Race Handicapping and Mailing List Management in R:base 5000, as well as a Morse Code tutor and a Space War game.

Volume 273 has an update of QModem (version 2.3). One of the features I like in Qmodem is similar to Ymodem Batch in ProComm — namely you can download a whole list of files from a BBS. The problem with this volume is that the beta copy of version 2.4 is already out and I would suggest downloading that from a BBS rather than getting the PC/BLUE volume.

Volumes 274, 275, and 276 contain DREAM — Data Retrieval, Entry, and Management (DBMS). There are certainly a lot of files for the system — 107 different files occupying 1,176K. Very impressive in size, and the specifications aren't that bad: 60 fields, 32,767 records per file, 79 characters per field, maximum record length of 1,580 characters, a 255 character sort field, and 9 files per report.

The documentation could be better, but if it meets your requirements the price is right.

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 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 BBS system (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. (Except for SIG/M 284 and 285, which are .ARC files since they're in PC–DOS.) In addition, any SIG/M or PC/BLUE volume (or any file from any volume) will be put up on request.

Unless you have a friend at the telephone company, you might prefer to order directly or through a local SIG/M or PC/BLUE distribution point.

■■■

Who's Making Great Hard Drives

By David Thompson
Staff Driver

And Great Drive Accessories

I wasn't planning to do another hard drive article, not just yet. But I've heard some disturbing things lately, things I think you should know about.

Who's making the best, most popular 20 meg drive? Seagate, right? Well, they're making the most popular 20 meg drive, but it may not be the best. Whether it's even a consideration for your new system depends on who you talk to.

Seagate 225s

Kaypro Corp. has stopped installing Seagates on their XT clones. Distributors are seeing a rising percentage of ST 225s and 238s returning from dealers, and the dealers I've talked to are being pestered by unhappy customers (in some cases half the drives are failing). Yet just 9 months ago Seagate 225s were an industry standard. What's the problem?

Seagate reportedly keeps the current track number in a register. That's great as long as the number in the register matches the track number the head is reading. When the controller requests a track, the drive checks the difference between the register and the requested track number and then steps the head appropriately.

Unfortunately, the location of the head and the contents of the register sometimes don't match. When this happens, the track number read from the disk doesn't match the track number inside the register. The controller issues an error and the drive retracts the head to track -1, then steps the head out once to track 0. At the same time it also resets the register to 0.

That's fine, except the drive calculates how many times it must step to reach track -1 from the contents of the register. If the register value is greater than the head position, the head tries to step past track -1, which is OK. When everything is finished the head is at 0, the register contains 0.

However, if the register value is lower than the number of the track the head is reading, then the head won't step back enough times to reach track -1, and the read errors continue. One person I talked to said the only way to recover from this type of fault was to power-down the system. However, I've also seen that kind of error disappear after a reset.

Often, a half-dozen retries will also correct the problem. Either way, a sector test program run immediately after recovery will find nothing wrong.

Better Drives

You notice that I picked on the two Seagates with the stepper motors. The full-height Seagates with the voice coil steppers (notably the 4038s) have remained very fast and very, very solid.

I've had a prejudice against the 3 1/2" hard drives, there's just not enough room there for reasonably sized bits, but the little Lapines have changed my mind. They're cheap (the same price as the Seagates), quiet (much, much quieter than the sewing-machine-powered Seagates), and so far they've been absolutely reliable.

RLL

The run-length limited method of storing information on the disk allows recording 50% more data without significantly increasing the cost of the drive. The Seagate 238 has been the most popular mate for the RLL type of controller.

Unfortunately I've heard some very discouraging reports from dealers. They're saying that RLL generates repeat business; every six months, customers come back for a replacement.

Brian Garrison reported that the 238s which come back can't read track 0. He says the scuttlebutt around the industry blames the problem on media damage caused by the high write current.

I talked to Cal-Aveco, a major distributor for Seagate, Miniscribe, and Microscience. They rated Seagate #1, Miniscribe #2, and Microscience #3. For RLL, they said the Seagate 238 and Microscience 40 megs were very popular, and they were having good luck with them. They wouldn't specify what "good luck" meant.

I asked what types of failures they were seeing, and they indicated that on the ST 225s, either one of the heads failed, or the drive became very noisy. The problems were showing up within the first few months of service.

They particularly like the Seagate 225, the Seagate 4000 series (auto-park), and the Miniscribe 6053 (40 meg).

Emerald Microware

Brian Garrison of Emerald Microware had a different story. He's seen a lot of stepper problems with the Seagate 225 and 238. Plus, he reports the 225s are often noisy, and most of the 238s have been defective.

"I returned eighteen 238s yesterday — most of the ones we had left. It's pretty interesting; the distributor is telling us not to sell them.

"Rodimes (203E) have been very good, only 3 returns out of the 60 or 70 I've sold, but Microscience is a different story. I've had 17 returns out of 20, and I expect to get the other three back because they have four months left on their warranties. I replaced the Microsciences with Seagate FT 238s, so I figure I'll be seeing them again."

Brian went on to say that Miniscribes look very good, especially the 6053s. Their price has been coming down rapidly. They are 44 meg, 28 ms drives with a voice coil, that gets the heads from one end to the other very quickly, but track to track time is a paltry 6 ms. Price is $600.

I asked him about the Lapines, and though he hasn't tried any, he said he's heard very good things about them.

SpeedStor

This combination formatter, partition editor, and device driver lets you create a single hard drive partition up to about 500 meg. This is really important if you have large drives and have files approaching MS-DOS's 32 meg limit.

It's important enough that the program's publisher, Storage Dimensions has sold 50,000 copies.

The program works with the standard high density techniques, RLL and ESDI, as well as the later versions of MS-DOS including unreleased V 3.3. (Lance Robinson wouldn't tell me what's new in 3.3, maybe there's nothing.)

Storage Dimensions
981 University Ave
Los Gatos CA 95030
408–395–2688
List price $99.95

MicroSolutions

Brian is particularly excited about the MicroSolutions cards which support 8" double density, 5" 1.2 meg, 3 1/2" and 5" 720K, Apple, and Northstar (hard sector) on a PC.

Their Matchpoint card and Uniform combination ($170) handles Apple and Northstar compatibility. You use your normal floppy controller card to control the PC's drives, the Matchpoint card occupies its own PC bus socket. The PC's drives can then read the Apple and Northstar disks.

The Compaticard (also $170) can either replace the standard controller or be co-resident with the PC's floppy controller. This card handles the other MS-DOS compatible formats such as 1.2 meg, 3 1/2", and 8".

MicroSolutions
132 West Lincoln Hwy
Dekalb Illinois 60115
815–756–3411

Konan Drive Maximizer Boards

These hard drive controller cards (only available for the XT at this point) compress data before sending it to the drive. Plus, they pack data into clusters so the maximum wasted space for any one file is 511 bytes.

Konan includes software which buffers disk data in RAM. The memory can be part of the machine's 640K or on an extended RAM card.

A Konan board will turn your 20 meg drive into a 30 meg drive (with 1.5 to 1 compression) or a 40 meg drive (with 2 to 1 compression). The trade off is speed. The 1.5 to 1 compression is faster.

The Konan's extended error correction code improves data reliability, and its transparent data packing makes retrieval faster. The price is $169. (I purchased one of these boards last week, and I'll get it installed as soon as this issue is finished....)

Konan Corporation supplied the drive configuration and size tables in Figures 1 and 2.

Konan Corporation
4720 S Ash Ave
Tempe, AZ 85282
602–345–1300

Solid Drives

Bill McDonald is Kaypro's drive specialist, and I touch bases with him from time to time just to find out what's happening. When he was having good luck with Seagate 225s, we had good luck with them; now that the new 225s are having problems I thought I'd call Bill.

He gave me the low-down on the stepper problems with the 225s, and then told me what they're using now.

He really likes the Priam 42 meg drives they're putting in the 286 and 386 systems. They cost more than other drives, but they come formatted and they come with a program like SpeedStor which lets you create 40 meg partitions.

Kaypro is also shipping NEC hard drives, and the failure rate is running under .1%. Panasonic and Toshiba floppy drives have been 99.1% good.

Finally

If you're planning to purchase a hard drive for your clone, don't be afraid to nose around a bit. It'll be time well spent.

On the other hand, even if you do get a lemon, it's very likely that you'll find out well within the warranty period. If you chose a good dealer and a solid manufacturer, you'll get a new drive.

Also, if you learn something, drop us a letter. Then we'll all know.

Figures 1 & 2 begin on page 46.

■■■

Figure 1

APPENDIX A

The unique KONAN configuration menu screens allow the operator to enter information about the model of hard disk being installed, below is a list of parameters.

Number of sectors = 18
Cylinder to start spares = OFF
Correction span = 11

Manufacturer	Model	# of Heads	# of Tracks	Precomp Track	Redwrc Track	Step Mode 0,6,or7
Atasi	3020	3	645	320	none	7
	3033	5	645	320	none	7
	3046	7	645	320	none	7
	3051	7	704	350	none	7
	3053	7	733	365	none	7
	3085	8	1024	512	none	7
CDC	94155-21	3	697	128	none	7
	94155-25	4	697	128	none	7
	94155-28	4	697	128	none	7
	94155-36	5	697	128	none	7
	94155-38	5	733	128	none	7
	94155-48	5	925	0	none	7
	94155-57	6	925	0	none	7
	94155-67	7	925	0	none	7
	94155-77	8	925	0	none	7
	94155-86	9	925	0	none	7
	94156-48	5	925	none	none	7
	94156-67	7	925	none	none	7
	94156-86	9	925	none	none	7
	94205-51	5	989	128	none	7
	94166-101	5	969	none	none	7
	94166-141	7	969	none	none	7
	94166-182	9	969	none	none	7
CMI	CM5205	2	256	128	128	7
	CM5410	4	256	128	128	7
	CM5616	6	256	128	128	7
	CM5206	2	306	128	none	7
	CM5412	4	306	128	none	7
	CM5619	6	306	128	none	7
	CM6213	2	640	256	none	7
	CM6426	4	640	256	none	7
	CM6640	6	640	256	none	7
	CM7660	6	960	512	none	7
	CM7880	8	960	512	none	7
Cynthia	D520	4	640	none	none	0
Disctron	D526	8	306	128	128	0
DMA	360	2	612	400	none	7
IMI	5006	2	306	214	none	0
	5012	4	306	214	none	0
	5018	6	306	214	none	0
Lapine	3522	4	306	0	none	7
	LT10	2	615	0	none	7
	LT20	4	615	0	none	7
Maxtor	XT-1065	7	918	none	none	7
	XT-1105	11	918	none	none	7
	XT-1140	15	918	none	none	7
	XT-2190	15	1224	none	none	7
Memorex	321	2	320	128	none	0
	322	4	320	128	none	0
	323	6	320	128	none	7
	324	8	320	128	none	7
	450	2	612	350	none	7
	512	3	961	480	none	7
	513	5	961	480	none	7
	514	7	961	480	none	7
Micropolis	1303	5	830	none	none	7
	1304	6	830	none	none	7
	1323	4	1024	none	none	7
	1323A	5	1024	none	none	7
	1324	6	1024	none	none	7
	1324A	7	1024	none	none	7
	1325	8	1024	none	none	7
Microscience	HH-612	4	306	none	none	0
	HH-725	4	612	none	none	0
	HH-312	4	306	none	none	0
	HH-325	4	612	none	none	0
	HH-315	2	612	none	none	0
	HH-712	2	612	none	none	0
	HH-1050	5	1024	none	none	6
Miniscribe	3212	2	612	0	off	7
	3425	4	612	0	off	7
	6212	2	612	0	off	7
	6032	3	1024	512	off	7
	6053	5	1024	512	off	7
	6085	8	1024	512	off	7
	8212	2	615	128	off	7
	8412	4	306	0	off	7
	8425	4	612	0	off	7
	8438	4	615	128	off	7
MMI	M106	2	306	128	none	7
	M112	4	306	128	none	7
	M125	8	306	128	none	7
	M206	2	306	128	none	7
	M212	4	306	128	none	7
	M225	8	306	128	none	7
	M306	2	306	128	none	7
	M312	4	306	128	none	7
	M325	8	306	128	none	7
NEC	D3126	4	615	256	none	7
	D5126	4	615	128	none	7
	D5146	8	615	128	none	7
Priam	V130	3	987	none	none	7
	V150	5	987	none	none	7
	V170	7	987	none	none	7
	V185	7	1166	none	none	7
	S14	11	1224	none	none	7
	S19	15	1224	none	none	7

Manufacturer	Model	# of Heads	# of Tracks	Precomp Track	Redwrc Track	Step Mode 0,6,or7
Quantum	Q510	2	512	256	none	7
	Q520	4	512	256	none	7
	Q530	6	512	256	none	7
	Q540	8	512	256	none	7
Ricoh	5130	2	612	400	none	7
Rodime	RO101	2	192	0	96	0
	RO102	4	192	0	96	0
	RO103	6	192	0	96	0
	RO104	8	192	0	96	0
	RO201	2	321	0	132	7
	RO202	4	321	0	132	7
	RO203	6	321	0	132	7
	RO204	8	321	0	132	7
	RO201E	2	640	0	none	7
	RO202E	4	640	0	none	7
	RO203E	6	640	0	none	7
	RO204E	8	640	0	none	7
	RO3055	6	872	0	none	7
	RO3045	5	872	0	none	7
RMS	RMS503	2	153	77	77	0
Seagate	ST506	4	153	128	128	7
	ST406	2	306	128	128	7
	ST412	4	306	128	128	7
	ST419	6	306	128	128	7
	ST213	2	615	300	none	7
	ST212	4	306	128	128	7
	ST225	4	615	300	none	7
	ST238	4	615	300	none	7
	ST251	6	820	none	none	7
	ST4026	4	615	300	none	6
	ST4038	5	733	300	none	7
	ST4096	9	1024	none	none	7
	ST4144R	9	1024	none	none	7
Shugart	SA604	4	159	128	128	0
	SA606	6	159	128	128	0
	SA612	4	311	128	none	0
Syquest	SQ306RD	2	306	none	none	7
	SQ312RD	2	615	none	none	7
	SQ325F	4	612	none	none	7
	SQ338F	6	612	none	none	7
Tandon	TM252	4	306	none	none	7
	TM262	4	615	none	none	7
	TM362	4	615	none	none	7
	TM501	2	306	153	128	0
	TM502	4	306	153	128	0
	TM503	6	306	153	128	0
	TM602S	4	153	none	128	0
	TM603S	6	153	none	128	0
	TM603SE	6	230	128	128	0
	TM702	4	615	none	none	7
	TM703	5	733	none	none	7
	TM755	5	981	none	none	7
Teac	SD-510	4	306	128	128	7
	SD-520	4	615	128	128	7
Toshiba	MK53FA	5	830	512	none	7
	MK54FA	7	830	512	none	7
	MK56FA	10	830	512	none	7
Western Dynex	WD505	2	310	none	none	7

Manufacturer	Model	# of Heads	# of Tracks	Precomp Track	Redwrc Track	Step Mode 0.6,or7
Tulin	TL226	4	640	none	none	7
	TL240	6	640	none	none	7
	TL326	4	640	none	none	7
	TL340	6	640	none	none	7

Figure 2

APPENDIX B

Storage Capacity Table

	# Cylinders	100	200	300	400	500	600	700	800	900	1000
#	2	1.8	3.6	5.4	7.2	9.0	10.8	12.6	14.4	16.2	18.0.
H	3	2.7	5.4	8.1	10.8	13.5	16.2	18.9	21.6	24.3	27.0
E	4	3.6	7.2	10.8	14.4	18.0	21.6	25.2	28.8	32.4	36.0
A	5	4.5	9.0	13.5	18.0	22.5	27.0	31.5	36.0	40.5	45.0
D	6	5.4	10.8	16.2	21.6	27.0	32.4	37.8	43.2	48.6	54.0
S	7	6.3	12.6	18.9	25.2	31.5	37.8	44.1	50.4	56.7	63.0
	8	7.2	14.4	21.6	28.8	36.0	43.2	50.4	57.6	64.8	72.0
	9	8.1	16.2	24.3	32.4	40.5	48.6	56.7	64.8	72.9	81.0
	11	9.9	19.8	.29.7	39.6	49.5	59.4	69.3	79.2	89.1	99.0
	15	13.5	27.0	40.5	54.0	67.5	81.0	94.5	108.0	121.5	135.0

Capacity in Megabytes

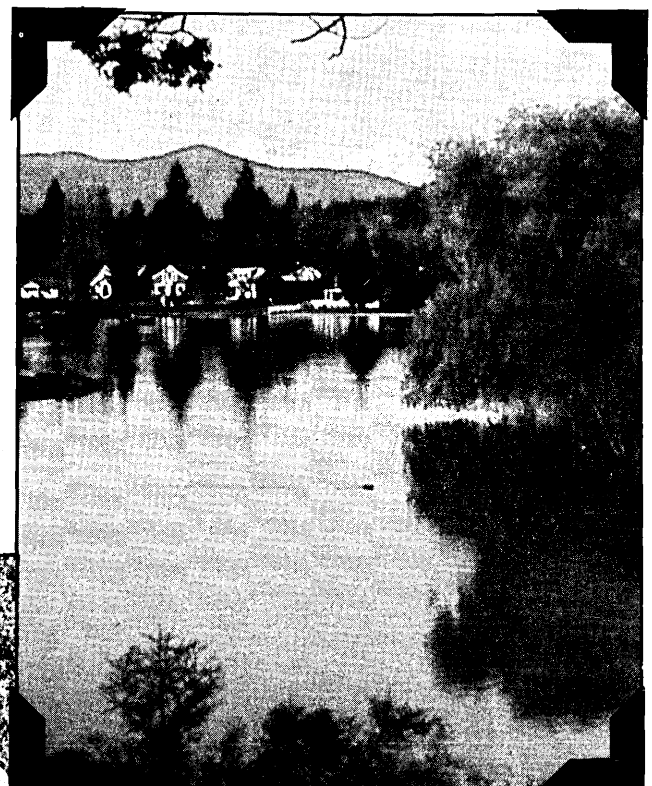
End of Listings

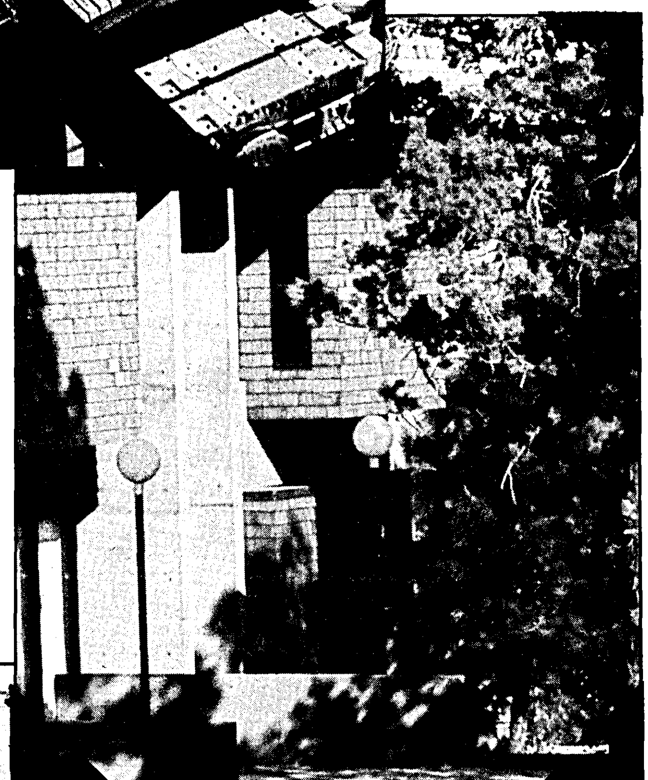
■■■

SOG

(Semi Official Get Together)

You'll make a scrapbook of memories!





Make your reservations today!

MICRO CORNUCOPIA'S SIXTH ANNUAL

SOG

(Semi Official Get Together)

at
Central Oregon
Community College
Bend, Oregon

SOG Registration
Check-in at
The Pinckney Center
C.O.C.C.
beginning at 9 a.m.
on Thursday

Four Days of Fun
and
In-depth Technical
Discussions

July 30 - Aug. 2, 1987
Thursday - Sunday

Dorm Check-in
beginning at 1 p.m.
on Wednesday
at the C.O.C.C. dormitory

SOG REGISTRATION INFORMATION

The sixth annual SEMI-OFFICIAL GET TOGETHER (or SOG as we call it) is coming and we need to know your plans! SOG will be held on the scenic campus of Central Oregon Community College in Bend, Oregon, July 30 through August 2, 1987.

Activities start Thursday with whitewater rafting and an Old Fashioned Barbecue. You'll find this the perfect opportunity to meet people and have some fun (volleyball got pretty wild last year). This is followed by two full days of technical conferences, demonstrations, workshops, and forums. There will also be new product displays and a swap meet. To help defer the cost of the conference this year, we'd appreciate donations in any amount.

REGISTRATION
Fill out and return the attached registration form so we'll know what activities you would like to participate in at SOG VI. Be sure and read everything carefully. Don't forget to include your payment when returning the registration form. Those arriving Wednesday and staying in the C.O.C.C. dorms should check in at the dormitory upon arrival for room assignments. If arriving Thursday, check in at the Pinckney Center to receive your SOG packet. (The SOG packet will include your tickets to the barbecue, Saturday night banquet tickets, and a speaker's schedule.) If you have any questions, give us a call at 503-382-8048.

DORMS
College dorm rooms for Wednesday night through Sunday noon are $85 double occupancy. Let us know if you wish to share and we'll try to fix you up with a roomie at $42.50 per person. People planning on arriving late Wednesday night must make special arrangements regarding check in. All rooms must be paid for by June 1, 1987.

CAFETERIA
The cafeteria will be offering all-you-can-eat buffet meals ($3.50-$4.50) throughout the SOG. Breakfast will be served early on Thursday to give all-day rafters the opportunity to eat before they leave. A leisurely continental breakfast will be served late Sunday so everyone will have a chance to sleep in.

CHILD CARE
There will be a supervised playroom for the kids Friday and Saturday from 9-4.

TRANSPORTATION
The nearest commercial airports are in Redmond (15 miles), Eugene (120 miles), and Portland (165 miles). Shuttles run between Bend and the Redmond and Portland airports. Trailways Bus also serves Bend. If you need travel information, call Cascade Travel, 503-382-3772.

Register by July 15th.

REGISTRATION FORM

______Number attending in your group.

NUMBER	EVENTS	PRICE EACH	TOTAL
	Thursday All-day raft trip & Old-Fashioned BBQ	$65.00	
	Thursday 2½ hour raft trip & Old-Fashioned BBQ	$30.00	
____ adult ____ child	Thursday night Old Fashioned BBQ only (included in raft trips)	Adult $10.00 Child $6.00	
	Dorms 4 nights 2 people per room Willing to share? ______ Do you smoke? ____	$85.00	
____ adult ____ child	Saturday night SOG Banquet	Adult $10.00 Child $6.00	
	Semi-Official SOG T-Shirt Adult sizes ☐ small ☐ med. ☐ large ☐ X-large	$6.50	
	Semi-Official SOG T-Shirt Youth sizes ☐ small ☐ med. ☐ large	$5.00	
	Donations		

☐ Check enclosed

TOTAL ENCLOSED ______

Exp.__________ Signature ____________________

Card No ______-______-______-______

Name ____________________

Company ____________________

Address ____________________

City __________ **State** ______ **Zip** ______

MICRO CORNUCOPIA • P.O. BOX 223 • BEND, OR 97709 • 503-382-5060

TOPICS

Here's a preview of Micro C's sixth annual microtechnical conference. Once again, we'll have a baker's two dozen speakers who'll have some very interesting things to talk about—

How To Design Systems with State-Of-The-Art Microprocessers: 80386, 68020, TI34010;
How To Interface Computers To The Real-World (for business & pleasure);
How To Engineer Knowledge;
How To Design And Program Expert Systems;
How To Unravel The Secrets Of MS-DOS;
How To Build Cheap (but very fast) Computers;
How To Develop Relational Databases;
How To Repair Aching Drives (and other parts);
How To Market Your Microcomputer Skills & Products:
And lots, lots more.
You might say it's a how-to conference. Come contribute (and soak up) expertise!

SOG ACTIVITIES (JULY 30-AUGUST 2)

THURSDAY
We start SOG VI off by offering two different exhilarating voyages down the Deschutes River. You can choose between...

A 2½ hour raft trip down 3 miles of class I-III rapids on the Deschutes River only minutes from town. $30 per person includes the Old Fashioned Barbecue. Check in at the Pinckney Center registration table before 11 a.m. Thursday for raft scheduling.

OR...

An all day expedition exploring the lower Deschutes River Canyon approximately 1½ hours from Bend. This section of the river has class I-IV whitewater including 7 major rapids. $65 per person includes a deli lunch and the Old Fashioned Barbecue. Buses leave at 7:30 a.m. sharp in the parking lot below the dorms.

THE OLD FASHIONED BARBECUE will be put on by the Micro C staff this year. We figure this is the only way to insure everybody gets enough to eat. (Micro C Staffers know how to eat!) Head burger flipper will be none other than Dave Thompson. Prices for the barbecue are $10 per person, $6 for children under 12. Thursday Night Barbecue is included in the price of all raft trips.

FRIDAY
The talks begin Friday with a variety of speakers covering all aspects of the computer world.

We are offering a sight-seeing trip to the beautiful Three Creeks Wilderness area for those less interested in computers. Next we'll spend a couple hours browsing the quaint shops in the rustic town of Sisters, 20 miles west of Bend.

SATURDAY
The workshops and forums continue... while those not so technically inclined can enjoy a trip to the Deschutes County Fair, a place for old-fashioned fun and entertainment. Located 15 miles north of Bend, the Deschutes County Fair will take you back to the days of the small time county fairs.

The SATURDAY NIGHT BANQUET is an all-you-can-eat affair at the college again this year. Price is $10 for adults, $6 for children under 12.

DEADLINES
Dorm reservations must be paid in full by June 1, 1987. All other activities must be paid by July 15, 1987 (except excursion trips).

THURSDAY OLD-FASHIONED BBQ

Thursday, July 30th
(after rafting about 6 p.m.)

Hamburgers
Teriyaki Chicken
Potato Salad
Cole Slaw
Chips
Watermelon
Cake
Brownies
Soft Drinks
Beer

Adults $10 — Child $6 Under 12

SATURDAY NIGHT BANQUET

August 1st

All-you-can-eat
Buffet Dinner

Roast Turkey
Ham
Baked Potato
Baby Carrots
Cranberry Sauce
Rolls
Dinner Salad
Cottage Cheese
Cheesecake
Chocolate Cake

Adults $10 — Child $6 Under 12

The 68000

By Margret Rosenberg
Sysop

Messages From The Board

Lately, the 68000 conference has been the hottest place on the Micro C RBBS. Begun at SOG V, this 68000 interested group has really gotten into some interesting projects.

Take a look with Margret as she scans through the messages from the board, or better yet, take a look for yourself. If you're not a regular user of the RBBS you're really missing out. (You get full privileges on your first log–in.)

The 68K conference is generating almost as much interest as the PD32 conference on Micro C's RBBS (503–382–7643, 8 bits, no parity, 1 stop bit). For more information about the 68000 project, see "A Cheap 68000 Operating System" on page 12 of issue #34. By the way, we printed the wrong phone number for Joe Bartel in that issue. The correct number is 503–254–2005.

Once you've logged onto the board, type "j 3" to join the 68K conference. Then an "r 1+" will display the messages, starting with the first one. An "ns" after the first message will keep the board from pausing after each message; you may want to use this if you're calling long distance and capturing the session on disk.

Here we go.

CONNECT
Micro Cornucopia Bulletin Board
PCBoard Software - Ver. 10.0 - 01/08/86

System up at 17:06 on 02–22–1987 at 1200 baud.

Please enter a C/R [C/R]=no?

Micro Cornucopia Magazine RBBS
** 300/1200/2400 Baud – 24 Hours/day **

Larry Fogg << Sysops >> Margret Rosenberg
** We are back to 300/1200/2400 baud with a new ROM in the Courier. **

What is your first name? Margret
What is your last name? Rosenberg
Checking user file. Please wait ...
Password (Dots Will Echo)? ......

Good evening Margret. Check your mail today [C/R]=yes? n

(117 min. left) Main Board Command? j 3

Welcome to the 68K Conference Margret!
Conference file directories are now active.

View other conference members [C/R]=no? n *(Don't bother viewing other conference members unless you have a spare hour.)*
Check your conference mail [C/R]=yes? n

(117 min. left) 68K Conference Command? r 1+

Date: 11–06–86 (17:15)
To: ALL
From: JOE BARTEL
Subj: SOFTWARE

Now that there is a conference, I am interested in any public software. A version of Small C would be very nice and I am willing to help anyone who wants to try and get it going. That would make the C user group library available.

More: [Y]es, [N]o, [NS]non–stop, [T]hread, [#], [RE]ply? ns

Date: 11–07–86 (19:35)
To: JOE BARTEL
From: FRANK HENRIQUEZ
Subj: 68000 COMPUTER

I have a 68000 computer design that I'm currently wiring up on an IBM PC wire–wrap board. The system as designed so far consists of:

68000 running at 12.5 MHZ without waitstates.

1 meg of dynamic ram.

Fujitsu MB 1422 memory controller (could be replaced by AMD or Nat Semi 8419 controllers).

2 serial ports, using the 68681 dual serial chip.

1 Western Digital 2797 floppy disk controller. (I intend to use 3.5" drives.)

The board has the traces for the memory chip, so wiring for the memory section is minimal. The hardest part was to find a memory controller that would run at 12.5 MHZ without wait states. Now, I need to ask a few questions:

1) Should the board act as a co–processor board for an IBM PC? (if so, the disk controller could be removed).

2) Should the board be designed as a stand–alone machine, with some capability to use IBM cards?

Right now, I'm just drawing 5 volts from the PC bus. As I planned to design the board originally, it would read a bootstrap loader from the first sector of a disk, then load an operating system (I was going to use CP/M–68K). If I make it an IBM co–processor, all the disk and video I/O (in fact ALL I/O) would have to go through the PC, and the concept of having to deal with an Intel processor makes me ill...

I'd really appreciate any comments, and I can upload the preliminary schematics as MacPaint files, if anyone is interested.

Frank

Date: 11–24–86 (15:08)
To: ALL
From: HAROLD MILLER
Subj: CORVUS

I just picked up a Corvus Constellation system. It's 68K based, but the factory just closed here in So. Oregon. There should be a flood of their products soon. I got my 512K ram, 20Meg HD system for less than $500, software included. Is there a users group anywhere in existence? If not, I have a couple of ex–Corvus techs here at the company and we could start one. Got any suggestions ? Please give me a call during working hours at 503–479–1248. I am connecting a few Apples into the net, and hope to add a IBM clone soon. Thanks.

Harold Miller

Date: 11–24–86 (17:54)
To: ALL
From: BOB DRENNON
Subj: 68010 UNIX SYSTEM

I hesitate to leave this msg after reading the preceding one, anyhow: I am negotiating for the acquisition of several partially complete 68010 based systems. It may be possible to obtain the design rights also, as the original company is now defunct. The box is of the "tower" style, and the partial system includes a VME bus cardcage, power supply, 1/2 height floppy, 68010 cpu board with provision for 1 meg memory, 80186 based I/O board (which also runs MS–DOS stand alone or as a process under UNIX), a SCSI port, and the UNIX/XENIX system V.

One of the boards has a Motorola 6845 CRT and drives a composite monitor. This monitor and an IBM PC type keyboard act as the main terminal and may also be for the purpose of running IBM video specific programs.

To have a complete running system one needs memory chips, a composite monitor, a keyboard, and a hard disk with SCSI interface.

I suggest these for you hacker types because this is an orphan system since the company has changed hands.

There are only four (4) semi complete systems available. There are 40 extra 80186 I/O boards. I could sell the partial systems for $1400 each.

If interested in the partial systems or design rights (group project??), contact me ASAP! Bob Drennon, my home phone is 707–526–0934, or thru E–mail on the Amiga Magic BBS at 707–579–0523.

Date: 12–05–86 (16:48)
To: ALL
From: DANIEL POIROT
Subj: C68K.ARC

WHOOPS!

I UPLOADED THE PD C COMPILER THAT I TALKED TO JOE ABOUT. AT THIS TIME IT RUNS UNDER PC–DOS AND CREATES 68000 ASM FILES. THE AUTHOR IS WILLING TO HELP CONVERT TO ANOTHER SYSTEM. THIS LOOKS LIKE A WORTHWHILE ADDITION TO THE 68000 SYSTEM.

DAN

Date: 12–08–86 (16:35)
To: ALL
From: DANIEL POIROT
Subj: TINY BASIC SOURCE

HELLO,

I HAVE UPLOADED THE .ASM SOURCE TO THE TIMELESS (OR USELESS) TINY BASIC AS IMPLEMENTED ON A 68000 WITH CP/M HOST. I GUESS THAT ANY SOURCE CODE IS BETTER THAN NONE. I RECALL SEEING THE C SOURCE TO A 68K CROSS ASSEMBLER ONCE UPON A TIME ON A BBS FAR, FAR AWAY. MY PLAN IS TO GET A 68K DEVELOPMENT SYSTEM RUNNING ON MY PC (HARD DISK, USABLE EDITORS, ETC....) AND THEN SEND THE OBJECT FILE TO A 68K TARGET. HAS ANYONE SEEN THIS ASSEMBLER? ANY HELP WOULD BE APPRECIATED.

DANIEL POIROT

Date: 12–15–86 (22:09)
To: ALL
From: SYSOP
Subj: HT–68K BOARD

I just received a newsletter from Joe Bartel and he says he's found a 68000 computer system that will be a hardware base for this whole thing. It's the size of a 5" drive and can be mounted anywhere. It's 8 MHz, 68000 with floppy controller, two RS–232, Centronics, 128K of RAM, and it costs $395 with the operating system.

The board can be expanded to 512K, and more RAM and ports can be added via an expansion board.

It needs a power supply (PC type is fine), drive(s), cabinet (PC type is fine), and a terminal.

The operating system includes an assembler, compiler, and line editor. The compiler is his HTPL, FORTH–Pascal cross.

More compilers etc. should be available as people get into this system.

Date: 01–05–86 (20:52)
To: ALL
From: JEFFREY YUAN
Subj: SINCLAIR QL

I have just recently purchased a Sinclair QL computer to play around with and I am very pleased with it. It is a 68008 based machine the size of a PC keyboard, with the following specs:

1: QDOS – a full multitasking operating system with built in device drivers
2: DOS and BASIC in ROM
3: 128k expandable to 640k of user RAM
4: ROM expansion slot
5: 256x256 pixel x 8 primary colors to 256 colors with mixing
6: 512x256 pixels x 4 primary colors to 256 colors with mixing
7: Built in LAN to allow 64 QLs to be tied together
8: 2 microdrives – they look like 1 inch square tape streamers – each microdrive disk holds about 100k
9: 2 serial ports and 2 joystick/mice ports
10: 68008 running 7–8 MHz
11: full size keyboard
12: RGB, composite, RF modulated TV, and PAL outputs
13: expansion slot for more memory and disk drives

(continued next page)

(continued from page 55)

I am sorry if this is starting to sound like an advertisement but I am very zealous about my new machine. The greatest thing about it is that I paid about $160 for it and it came bundled with a word processor, spreadsheet, business graphics program, and database. It also supports up to 32767 windows and all routines are directly accessible by TRAP instructions. I am using this machine to learn 68k assembly. The machine seems to be a real sleeper here in the U.S., though it is apparently quite popular in Europe. The downside to this machine is that the microdrives are a bit slow, though RAM caching does speed things up, and that the microdrive disks are rather expensive (4 for $9). Otherwise I am very happy, and there seems to be much software for it, such as Lattice C, and all the Metacomco languages. I would like to hear from anyone else who has one of these machines.

Date: 02–02–87 (23:04)
To: ALL
From: AL SZYMANSKI
Subj: SMALL–C 68K PROJECT

Joe, I have the code generating part of the compiler done and I think it works properly. Now I am working on the math/logical library. What fun!!!! Do you get Dr. Dobbs? Check out page 117. of the Feb. 87 issue. Looks almost too good to be true. *(Editor's note: Check ad for the QL in this issue of Micro C.)*

catch you later//al

Date: 02–05–87 (20:22)
To: ALL
From: DAVID WITTEN
Subj: 68K HACK SYSTEM

I just today received K–OS one from HT. It is so specifically what I have been looking for that I am amazed. I have wirewrapped a 68008 board from Motorola application notes and had an addressing PAL burned to bring up just the sort of system this OS addresses. My initial plan was to run native FIG–FORTH but found out just recently that this old implementation is so buggy that FIG recently dropped it, leaving me somewhat in the lurch. I will be following the progress of this conference with interest.

Date: 02–06–87 (21:07)
To: DAVID THOMPSON
From: CHRIS JOHNSTON
Subj: HT–68K BOARD

Thanks! I saw the ad in the latest Micro C for the little 68K board. It looks neat. I'll contact Joe Bartel at Hawthorne Tech. Thanks again!

Date: 02–12–87 (22:45)
To: ALL
From: GERALD WINSLOW
Subj: HT 68K BOARD

I WOULD BE VERY INTERESTED IN HEARING FROM ANYONE WHO HAS THE HT. 68K BOARD; IT REALLY SOUNDS AND LOOKS NEAT. HARD TO IMAGINE SO MUCH POWER ON ONE SMALL BOARD. BY THE WAY, IF ANYONE HAS LOOKED AT THE DOCS. SENT OUT BY HT, YOU'LL NOTE THAT THERE IS A 64K BYTE HOLE IN THEIR MEMORY MAP. THIS IS NOT THE CURRENT MAP, ACCORDING TO HT. AS I UNDERSTAND IT, IT'S CONTINUOUS THRU 4M BYTES.

Date: 02–17–87 (02:40)
To: ALL
From: FRANK HAUSMAN
Subj: SINCLAIR QL

I was intrigued by the ad on page 117 of Dr. Dobbs February 87 issue – in fact, I checked the cover to make sure it wasn't the April issue. I called Quantum Computing in Dover New Jersey <very> early Monday morning and Frank the main man of Quantum Computing answered the phone. He is genuinely zealous about the machine! He did not high–pressure sell it, he did know the machine quite well, and he was genuinely fun to talk to. What I'm interested in is seeing one in action before I buy it. I live in Mountain View, California, and my message phone # is 415–968–1515. If you have a QL or know someone who does near here (San Francisco area), please give me a call. If this is the same Sinclair who did the ZX–80, why have THEY not advertised this machine in the states? It's obvious that there is a need for some kind of standard 68000 machine to rise above the Atari/Amiga/Mac proprietary bullshit wars, so what gives?

Give me a hoot...

Frank Hausman

Date: 02–19–87 (00:42)
To: FRANK HENRIQUEZ
From: ALLEN MORRIS
Subj: 68000 COMPUTER

How about you toss the IBM motherboard, make a motherboard with the 68K on it. Put in two or three slots that are IBM compatible (i.e. wire the I/O bit to a memory address. Humm this could get complicated.) And you could end up with a 68K machine that you could get cheap peripherals for. What an idea. I have really come to hate programming in C on the Intel chips. Wish I had my PDP11–15 back.

Date: 02–19–87 (00:44)
To: HAROLD MILLER
From: ALLEN MORRIS
Subj: CORVUS

I'm looking to get a 68000 machine, could you direct me to a good source to get one of these Corvus machines? I'm in the SF bay area.

(104 min. left) 68K Conference Command? g

Time Logged: 13 minutes
Time Used: 13 minutes

Thanks for calling Margret ...

■■■

RP/M2™ creates

Z80®

CP/M©2.2 compatible

IBM PC

Now available for Decmation's Blue Thunder softcard, PC RP/M2 is an **operating system.** Either standalone or with DOS present, PC RP/M2 provides the solid base of a genuine operating system reliably distinct from the facade created by an MSDOS interface. **All 2.2 system and CBIOS calls are supported,** with 56.5k TPA, file date and time stamping, fast virtual disk, iobyte redirection, terminal emulation, color console display, auto relog, COM path, addressable SAVE, single key phrase recall. SETDISK redefines a drive to any of over 80 CP/M formats. DOSDISK invokes built-in access to DOS drives. **System disk with manual $129. Blue Thunder™ softcard $149. Shipping $5 ($10 nonUS)**

Micro Methods, Inc.
118 SW First St. - Box G
Warrenton, OR 97146
(503)861-1765

Quelo® 68000 Software Development Tools

Quelo Cross Assembler Packages are **Motorola compatible.** Each package includes a macro assembler, linker/locator, object librarian, utilities for producing **ROMable code,** extensive indexed typeset manuals and produces **S-records,** Intel hex, **extended TEK hex, UNIX COFF** and symbol cross references. **Portable source** written in "C" is available. It has been ported to a variety of mainframes and minis including **VAX, Sun, Apollo, Masscomp,** and **Charles River.** Native versions available for **Amiga** and **Atari ST.**

68020 Cross Assembler Package

Supports 68000, 68010, 68020, 68881and 68851.
For CP/M-68K and MS/PC-DOS, $ 750

68000/68010 Cross Assembler Package

For CP/M-80, -86, -68K and MS/PC-DOS, $595

68000/68010 **Simulator**

For MS/PC-DOS by Big Bang Software, Inc.

SIM68K is a cost-effective software simulator for debugging 68000/68010 code on an IBM-PC or compatible, $285.

Call **Patrick Adams** today:

COD, Visa, MasterCard

Quelo, Inc.
2464 33rd W. Suite #173
Seattle, WA USA 98199
Phone 206/285-2528
Telex 910-333-8171

Trademarks: CP/M, Digital Research; MS, Microsoft Corporation; Quelo, Quelo, Inc.

ALL SALES SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. FREE COPY UPON REQUEST.

CANON 80 COLUMN PRINTER - $39.95

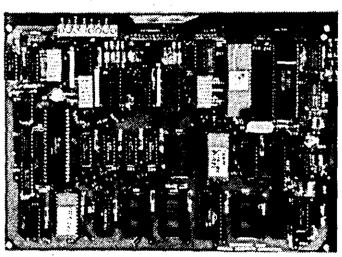
ORIGINALLY MANUFACTURED FOR THE PC JR. BUT WITH OPTIONAL CONNECTOR WILL WORK WITH PC, XT, OR AT. REQUIRES SERIAL I/O. THIS THERMAL PRINTER IS QUIET AND USES EASY TO GET 8½ IN. ROLLS OF PAPER. 50 C.P.S., UPPER AND LOWER CASE, PLUS GRAPHICS. ORIGINAL LIST PRICE $199.00. ADD $3.00 FOR PC/XT CONNECTOR. ADD $5.00 UPS.

THE NEW 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.



$89.95 #ZRT-80
(COMPLETE KIT, 2K VIDEO RAM)

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

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

Digital Research Computers

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

Call or write for a free catalog on Z-80 or 6809 Single Board Computers, SS-50 Boards, and other S-100 products.

TERMS: Add $3.00 postage. We pay balance. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCard. Texas Res. add 6-1/4% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance.

C CODE FOR THE PC

source code, of course

Panache C Program Generator	$125
The Profiler	$100
QC88 C Compiler	$90
Concurrent C	$45
Biggerstaff's System Tools	$40
Coder's Prolog in C	$45
Translate Rules to C	$30
ICON String Processing Language	$25
LEX	$25
YACC & PREP	$25
tiny-c interpreter & shell	$20
C Tools	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

Free shipping on prepaid orders MasterCard/VISA

By Laine Stump

Development Foundation of Turkey
Tunali Hilmi Cad. 22
Ankara, Turkey

On The Walls Of Old Constantinople

Laine takes us to Istanbul to visit the ruins, mentions the DuangJan 1.2 editor, and waxes eloquent on PC Tech's new products. Finally, he runs 720K drives on AT–style systems.

Good day! Welcome once again to the Micro C Adventure Column. This month I'm back on the bus between Istanbul and Ankara. This is the eighth time in ten weeks that I've spent my weekend in Istanbul. I'm not quite sure why I keep going; all I know is it's a heck of a lot less boring than sticking around in Smogville all the time. There's just something magnetic about the "Center of the World."

Last weekend I managed to make it into a business trip. I took the midnight bus on Thursday and spent all day Friday and the following Monday talking to typesetting companies and newspapers, gathering information about what equipment is currently being used in Turkey.

I'm gathering all this information mainly for two reasons. The first is to assist a company in Istanbul in transferring documents from their PCs to the typesetting machine used by their printing company; eventually they may even do all their typesetting in–house, but it will be a gradual transition. The second reason is to get ideas on how to set up a large scale typesetting and printing house to serve newspapers in several small towns along the Black Sea Coast.

Looking at what people are using here, I was at once surprised at how modern some of the techniques and equipment actually are, and also at how primitive some other aspects are. For instance, the print shops have color separation down to an art (any newspaper worth its salt in Turkey has color pictures and headlines on every page), but the reporters still use typewriters and give their copy to the typesetting department on sheets of paper, ready to be typed a second time, and layout is done with scissors and glue (are you listening, Dave?).

After using a computer for all my writing for more than 5 years, I can't imagine how they get anything done. Every day, though, I see the newspapers on the stands, complete with half page color pictures of scantily clad women. Turkish papers must be seen to be truly appreciated — kind of like a double–sized National Enquirer. I bet you've never seen a front page headline that says: "Brooke Shields Loses Her Virginity!" have you?

But that was all last weekend. This weekend was purely pleasure. I came over with another resident of "The Fiery Pits of Hell" (Ankara) to visit a mutual friend who also happens to be a student of archeology. Our weekend was a wonderful combination of carpet shops, a Russian restaurant (complete with lemon vodka) and other fine foods, and an afternoon tramping along the top of the Old City wall, set up hundreds of years ago to protect the old city of Constantinople from the invading hordes.

Normally this would have been just another hike across a bunch of old rocks, but our guide was able to spice it up nicely with his knowledge of medieval battle tactics, the anatomy of the Constantinople defense system (2 walls working in combination with a moat), and countless little tidbits of history ("This is the well in which one of the Sultans was deposited after he was deposed").

Neat Stuff

I got a reply out of my column on writing international software! Jirapol Pobukadee sent me a flyer about his company's multi–lingual word processor.

DuangJan 1.2 from MegaChomp is available in European, Cambodian, Greek, Hindi, Laotian, Punjabi, Russian, Armenian, Tamil, Thai, and Vietnamese versions. They are also open to creating versions for any other language which has no more than 97 characters and reads from left to right.

The really amazing part? It's only sixty bucks! I saw an ad for something very similar in a magazine just this week and it was over $300! But it could also do Arabic and Hebrew....

MegaChomp Company
3524 Cottman Ave.
Philadelphia, PA 19149–1606
(215) 331–2748

More Neat Stuff

I also got a wonderful little family newsletter from the folks down in "Valley X." It seems that PC Tech has lots of neat new things they've released recently, or will release in the next few weeks.

For starters, the X–16 ROM BIOS has been updated to fully support 720K and 1.2M disk drives. You could use a 720K (96 tpi) drive on an X–16 before, but only to read 720K disks. With the latest versions of the X–16 ROM, the X–16 will automatically "double step" to allow reading 360k diskettes on a 720K drive. If you have an older X–16, call PC Tech for information on getting an update for your ROM.

There are also lots of new hardware gizmos and whizbangs on their latest dinner menu. First on the list is the "Jetta V40," a board similar to the X–16, lacking a SCSI port and 8 bits on the data bus, as well as one of the serial ports. This board is a bit slower than the X–16, and isn't as flexible due to the lack of SCSI, but it's slightly cheaper, and because of its NEC V40 it can run 8080 software. I, being the speed freak I am, would rather have a 10 MHz X–16 with an OMTI SCSI controller on the winnie, but some may be content with speed somewhere between a Turbo Board and an X–16.

Another interesting new board that I was told about last week is a memory board with 16 Megabytes (no, that is NOT a misprint) of memory that can be used either as EMS extended memory or as expanded memory on an AT running under Xenix or Unix. (Is it extended and expanded? or expanded and extended? I always get mixed up...) All these megs are packed on the board in much the same way as the 4–Meg board — Dean uses those wonderful little 256K SIP DRAM packages.

In the meantime, the price of the X–16 has been constantly dropping. See the ad in this issue for the latest (or better yet, call your stock broker for an up to the minute quote).

Now if they would just get back to work on that 386 machine they hinted about at SOG....

96 TPI

PC Tech's newsletter said, "a 360K diskette cannot be formatted on a 720K drive, even though the DOS manual says so." Since my ROMs haven't been updated I haven't tried it out, but I've heard a rumor somewhere that it can be done. You just use DISKCOPY to copy an existing 360K diskette and specify the 720K drive as the destination. Since DISKCOPY formats as it copies, this will give you a 360K diskette that can be safely written to on a 720K drive. You must use a diskette which has not been previously formatted.

(Editor's note: I haven't tried it, but I suspect you'd get a 360K disk that only a quad drive would love. Assuming DISKCOPY isn't smart enough to double–step the 720K head between each track, it would create a 360K disk all right, but not one a 360K drive could read.)

By the way, you can install a 720K drive on most any clone board, but unless you write or buy some extra software, none of this double tracking stuff will work. Double tracking is a function of the ROM BIOS, and it's only included on AT compatibles and the newest X–16 (and possibly one or two other high performance boards similar to the X–16).

DRIVPARM

Another hint about using 720K or 1.2M drives. The MS–DOS 3.2 manual says it's necessary to install their DRIVER.SYS device driver in order to properly use and format high capacity disks. This isn't true, and complicates system operations needlessly by forcing you to refer to your floppies as drives C and D (or E and F, or whatever) when formatting. It's much easier to use the DRIVPARM command in CONFIG.SYS.

I have no idea how DRIVPARM manages to interface to the BIOS, or even where the command is documented, but I know it works. I learned about DRIVPARM from Earl Hinrichs at PC Tech. Since there's no mention of DRIVPARM in the DOS manual, I'll tell you how to use it and give a couple of examples.

DRIVPARM tells DOS (or BIOS or somebody) what kind of drive is connected at which physical address. This information is then used when formatting. There are three types of drives supported that I know of: 0 – 360K, 1 – 1.2M, 2 – 720K. You tell DOS which kind of drive is in which position by putting the following command in CONFIG.SYS:

```
DRIVPARM=/D:d /F:f
```

where d is the position of the drive (i.e. A=0, B=1) and f is the type of drive (as given above). For instance, if drive A was 720K and drive B was 360K, you would use:

```
DRIVPARM=/D:0 /F:2
DRIVPARM=/D:1 /F:0
```

Actually, the line for drive B isn't necessary, since 360K is assumed, but you get the idea.

Two disclaimers about this. First — it only works on MS–DOS or PC–DOS 3.2 or above! None of this neat stuff was around on v3.1 even. Second — as I said before, you can't do double stepping on an XT or an XT clone (although you can format and read 720K disks with no problem). If you want to read 360K disks on a 720K drive, you'd better spring for an X–16 or AT clone, or be prepared to spend several eons poring over the uPD765 spec sheets.

(Editor's note: MS–DOS 3.2's bug list has reached 75 pages. If you don't have 3.2 now, don't get it. Wait for 3.21 which is just about out.)

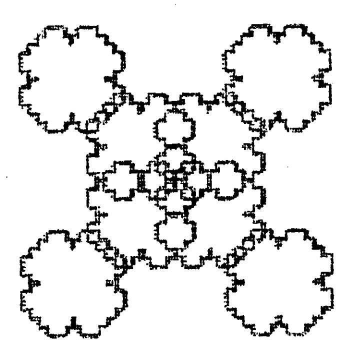
3.5 Inch

While I'm talking about this, I guess I should mention something I've never told anyone about before. I discovered last summer that it's entirely possible to read and write Atari ST diskettes on my Toshiba. I remember seeing something in the "Best of BIX" part of BYTE that somebody had tried it and it wasn't possible. Don't believe everything you hear from the peanut gallery.

I do seem to recall that I had a slight problem every once in a while reading one or two odd sectors from a Toshiba formatted diskette on the ST (or maybe it was vice–versa — I always get mixed up...), but for the most part everything went just fine. I can say that if you have a PC and an ST, you should buy yourself a cute little 3.5 inch double sided 80 track drive and plug it into your PC (they're electrically identical to a 5 inch 80 track drive). It can make transferring source code and data much easier on your blood pressure. Don't forget that you'll need DOS 3.2.

■ ■ ■

Fractals By Fogg



Want To Learn Assembly Language?

By Russ Eberhart
Applied Physics Laboratory
Johns Hopkins University
Laurel, Maryland

Books For The Beginner & The Hacker

I have a certain fascination with assembly language. I guess it's the idea that I'm really right down there talking one on one with the machine. No interpreters, no translators, no one else deciding what I can or can't do.

But learning assembly language isn't trivial, especially on the 8088 (I'd like to segment a few of its registers), so learning materials are especially important. Stay tuned as Russ digs out the good and the not–so–good primers and reference works.

Each person who takes the plunge into assembly language does so for a slightly different reason. Except for the occasional computer language junkie, who adds languages to his repertoire the way a gunfighter adds notches to his gun handle, most folks are drawn to assembly language because they want to do things faster, with more power, or more efficiently (or a combination of the three).

Many assembly language programs run at least twice as fast as comparable Pascal or C programs, two to four times as fast as compiled BASIC programs, and five to ten times as fast as interpreted BASIC programs. And there are things you can do in assembly language that would be impossible in any other environment. Examples include special graphics procedures and high speed serial communications.

My Start

My introduction to 8088 assembly language came about in late 1985. I wanted to compile one of my BASIC programs. But it contained MOTOR 1 and MOTOR 0 commands to close and open the the cassette port relay (on my PC yet).

I was using the cassette port relay to control an alarm for a security system, but my BASIC compilers (QuickBASIC and IBM's BASIC Compiler Version 2.0) didn't support the MOTOR commands.

So I figured I'd learn just enough 8088 assembler to poke the appropriate data statements into a BASIC program. Little did I know that I couldn't learn just a little assembler.

As I learned assembler I became aware of more ways to use it, which meant learning a little more assembler, which made me aware of more ways to use it... You get the picture.

High Level

I've been programming since about 1963, when I wrote my first programs in FORGO on an IBM 1620. Since then I've had brushes with assembler on the Z80 and the 6502, but for the most part I've used higher level languages like FORTRAN and BASIC. My hardware interests have included robotics and speech recognition, but I've generally been able to leave the assembly language programming to someone else.

It was with that dubious background that I went searching for a tome on 8088 assembler. Since finishing that first book, I've worked my way through two others, and have skimmed several more, primarily at the request of friends who wanted to get started in assembly language programming.

In this article, I'll discuss several training texts and I'll describe reference texts, such as the *DOS Technical Reference Manual*, and how they can help the process along. Of course it's particularly important to have a resource to help you dig out of the messes you'll make.

What's Your Experience?

Before I get started, however, I'd like to issue a request for feedback. If you've had experiences, good or bad, with learning to program in assembly language for the 8086/8088, let me know about it on the Technical Help Conference area of the Micro C RBBS.

I'll be curious about your background, and why you decided to learn assembler, in addition to any specific observations you have on assembly language primers, reference books, etc.

I'll do my best to reply when asked. I hope to do a follow–up to this article based on your comments.

Assembly Language Primers

In the early days of the IBM PC (roughly from 1981 until 1983) there weren't, to the best of my knowledge, any primers on assembly language that really taught you how to program. The typical book began with an annotated list of op codes for the 8086/8088, and, occasionally, finished up with a few examples of assembly language code.

You needed a resident guru to guide your quest because the books were only good for reference.

Bradley's Book

Two honest–to–goodness primers on assembly language appeared in 1984. One of them, entitled *Assembly Language Programming for the IBM Personal Computer*, was written by David J. Bradley, a member of the team that designed and built the IBM PC. Bradley's book includes a complete treatment of the 8087 coprocessor, something that's missing in every other book I've seen. He also covers the serial and parallel ports, with examples.

Bradley's book has two drawbacks. The first is style. I wanted an interactive tutorial, which means having relatively short routines which I can enter and run.

The second drawback is the content, or rather, the lack of it. The book was written prior to DOS 2.0, so all disk

I/O is done with the old File Control Block techniques, rather than via File Handles. (If you haven't programmed 8088 assembly language, take my word for it, you're better off learning File Handle manipulation.)

Lafore's Book

Because of these two drawbacks to Bradley's book, I wound up using another 1984 book, *Assembly Language Primer for the IBM PC & XT*, by Robert Lafore, as my assembly language primer.

(However, now that I'm comfortable with assembler, I use Bradley's book as a reference more often than Lafore's.)

Lafore's book weighs in at a hefty 501 pages. But before you're intimidated by the size, I'd like to point out that it takes that many pages to carefully, clearly, and completely explain each concept.

He takes you step by step, first using DEBUG as you write and trace through simple routines, learning what the registers and flags are all about. Then he takes you inside DOS before finishing with an introduction to a MACRO Assembler.

He has a chapter on sound, another on graphics, and one on disk I/O. He introduces new op codes as they're needed and each is covered in enough detail that you really understand it.

Another nice thing about Lafore's book is that it's relatively error free. I was able to concentrate on learning, rather than figuring out what it was he MEANT to say, or what the listing SHOULD have looked like.

I only have one major beef with the book. In Chapter 13, "Interfacing to BASIC and Pascal," it spends 21 pages talking about BASIC's USR statement before mentioning BASIC's CALL (in only 7 pages).

I know very few people who use USR. It's too involved, too finicky, and you can only pass and return one argument with USR.

Meanwhile, he didn't cover CALL thoroughly enough for a beginner to really use it.

In summary, I recommend Lafore's book. It's particularly appropriate if you want to take the time and effort to learn assembler thoroughly.

Metcalf And Sugiyama

In mid–1986, I noticed a new assembly language book in the bookstores, the *Beginner's Guide to Machine Language on the IBM PC and PCjr*, written by Metcalf and Sugiyama, and published by Compute! Publications. It was the second book I read thoroughly.

And no, it isn't really about machine language; it's about assembly language. I'm a little surprised the editors at Compute! Publications let the title go to press.

Despite the error in the title, this is a very good book. Metcalf and Sugiyama cover roughly the same material as Lafore in two thirds as many pages. The information is clearly presented but a few items have been left out.

For example, in the chapter on BIOS interrupts, they refer you to the *Technical Reference Manual* for information on RS–232 I/O (interrupt 14h).

The example routines which appear at the end of several of the chapters are generally quite good. I have only two bones to pick in this department.

First, there are no examples at the end of the chapter on the DOS Function Interrupt. (This is one of the most important concepts in assembly language programming.)

Second, the program at the end of Chapter 12 won't run as written. Several "mov" instructions with segment overrides have to be replaced with "mov byte ptr" instructions.

I recommend this book. It's a more condensed way to learn assembly language than Lafore's book, and it may be appropriate if you're in a hurry to become productive.

And, this is the book that I pick up when I'm curious about an obscure op code or pseudo–op. (There's one exception: For MS–DOS interrupt information I reach for *Advanced MSDOS* by Ray Duncan.)

Peter Norton

The last assembly language primer that I worked through page by page is *Peter Norton's Assembly Language Book for the IBM PC*, co–authored by John Socha and published by Brady Books. I first noticed it in a bookstore in Kansas City in early December 1986 while I was on a trip. After sowing a few not–too–subtle hints, I found it under the Christmas tree.

I almost don't know how to begin giving you my reaction to the book. Let me first say that, to me, Peter Norton has been on a pedestal ever since his *Inside the IBM PC*. That book, and the Norton Utilities, probably had more to do with my PC learning process than anything else. And his *Programmer's Guide to the IBM PC* seems to me to be another fine and unique accomplishment.

That said, I now have to say, I think his new book is a dog. There, I said it and lightning didn't strike me. *(Editor's note: Yet.)*

Norton's assembly language book is full of mistakes. Sometimes the mistakes are amusing, often they're annoying, occasionally they're really frustrating.

The book seems to have been rushed to the printer unedited. For instance, he spends the whole book building the program "DSKPATCH." Finally, he offers a finished version of the program on disk for $24.95. It does all those things the program you've just slaved over for several weeks doesn't do — it works.

I really felt ripped off. He could at least have provided program listings in the appendices so that hardy, thrifty souls could enter the program themselves.

And the mistakes. Mama mia! I hardly know where to start. I'll just give you a flavor.

In two figures early in the book, he labels 1100 binary with the hex value A. (It's C, Peter.) And several of the program listings are so bad, you have to stop learning and debug. Perhaps the worst is Listing 17–7, in the chapter on the ROM BIOS routines. Correcting this routine took me an entire evening.

Skip this book.

Leo Scanlon

Before I leave the subject of assembly language primers, I want to mention one more book, *IBM PC & XT Assembly Language*, by Leo Scanlon, published by Brady in 1985. Although I haven't worked through it page by page, several people I know said they like the book, so I spent one evening poking through it.

This book covers basic assembly language, macro assemblers, and it covers advanced tools and techniques.

Scanlon covers some subjects either not mentioned or only mentioned briefly in other books. For example, he has a complete chapter devoted to operat-

(continued next page)

(continued from page 61)

ing on data structures. In that he includes sections on unordered lists, ordered lists, look-up tables, and text files.

He also has chapters devoted to macros and structured programming.

In addition to the code examples sprinkled liberally throughout the book, Scanlon includes exercises at the end of each chapter so you can check your progress. (The answers are in the back of the book.)

Most of what he covers, he covers pretty thoroughly. For example, in the section on look-up tables, he spends four pages calculating the sine of an angle.

Unfortunately, Scanlon doesn't cover all subjects equally. For instance, he devotes fewer than two pages to file management using file handles and the extended DOS functions. And, he doesn't seem to cover .COM versus .EXE files at all. Oh well, from what I hear, .COM files will go the way of the dodo bird when DOS 5.0 arrives.

Scanlon's book looks pretty good. I didn't spend enough time with it to evaluate it thoroughly, but it looks like it could be a good book to use in a classroom situation.

Three things prompt me to say that.

First, it's an overview of more than just assembly language programming.

Second, it has exercises.

Third, it would be easy for an instructor to amplify on Scanlon's material.

Reference Books

After you've learned the basics you'll have questions that go beyond the scope of the previous books or you'll need a quick memory refresh. (For example, you might want to know exactly how a particular op code affects the registers.) For this kind of information you'll need a reference book or two.

Three of the most useful reference books are sometimes overlooked.

The first is the *DOS Reference Manual* you get from IBM, or whoever provided the version of DOS on your machine.

The second is the *Technical Reference Manual* for your machine. This can be very helpful, particularly if you're dealing with ports or other hardware.

A close pass through the ROM BIOS listing can be a particularly good procedure for learning both the machine and some tricks that aren't discussed in texts.

The third is the *DOS Technical Reference Manual.* To be honest with you, I didn't know there was such a thing as a *DOS Technical Reference Manual* until about two years ago! It's released by IBM for each release of DOS, although, as recently as January 1987, the latest DOS Tech Ref Manual covered version 3.1.

It's a gold mine if you're curious about screen and keyboard control, disk allocation, DOS interrupts, and DOS function calls. In my copy, the chapter on DOS interrupts and function calls is 178 pages, more than half the manual.

Unfortunately, IBM manuals are expensive. They may be the kind of book you want to borrow/beg/review instead of own.

Affordable References

Peter Norton's *Programmer's Guide to the IBM PC,* and Ray Duncan's *Advanced MSDOS* are more affordable.

Norton's book does a great job of tying together the hardware and software on the PC in ways that even I can understand. Although an elementary knowledge of BASIC is useful, all that's required is a desire to learn. And, as Norton says in his introduction, "One of the most important elements in this book is the discussion of programming philosophy."

Duncan's book focuses on software. Written for the experienced assembly language or C programmer, it's sort of a tutorial version of the DOS Tech Ref manual. Duncan gives assembly language and C code examples as he goes along.

As he states in the introduction, "It is my belief, (based upon experience) that working, well-documented programs are worth far more as learning tools than narrative exposition and tables...."

I agree.

Section II's detailed explanations of the DOS interrupts (20h through 2Fh) were particularly helpful. He also covers the functions available through Interrupt 21h (and the differences among the versions of MS-DOS).

Duncan leaves no stone unturned. This is a book you'll need after you graduate from your primer.

The Resident Guru

No matter how many books you read, no matter how many references stand in your library, you're going to be stumped. Really stumped. Enter the resident guru. And, enter one of the biggest innovations in learning.

It used to be absolutely necessary to have a resident teacher, someone to lend a hand or a towel. You generally found such folks on college campuses or in large cities.

I'm in the Baltimore-Washington area, I work for The Johns Hopkins University, and I have gurus like Scott Foerster and Bob Platte to bother.

But what do you do when you're in East Panhandle, Oklahoma? (Wait a minute, before you accuse me of being something awful like a "New Yorker," I should tell you that I was born in Liberal, Kansas. Find THAT on your world map.) *(Editor's note: He's right, there really is a Liberal in Kansas.)*

RBBS

However, now you have help as close as your computer. Because of bulletin boards, such as the Micro C RBBS you can get help no matter where you live.

Help is even available on bulletin boards from some software houses, such as Borland and Microsoft. I believe computer bulletin boards are revolutionizing the way we learn many things, not just computer software.

So, buy your favorite assembly language primer. Locate some reference books. Find a guru or dial up the Micro C RBBS. And whatever you do, enjoy.

Resources

Assembly Language Programming for the IBM Personal Computer
by David J. Bradley
published by Prentice Hall
$29.95
ISBN: 0-13-049189-6

Assembly Language Primer for the IBM PC & XT
by Robert Lafore
published by Plume/Waite
$24.95
ISBN: 0-452-25711-5

Programmer's Guide to the IBM PC
by Peter Norton
published by Microsoft Press
$19.95
ISBN: 0-914845-46-2

Beginner's Guide to Machine Language on the IBM PC and PCjr
by Metcalf and Sugiyama
published by Compute! Publications
$14.95
ISBN: 0–942386–83–3

Advanced MSDOS
by Ray Duncan
published by Microsoft Press
$22.95
ISBN: 0–914845–77–2

Peter Norton's Assembly Language Book for the IBM PC
by Peter Norton and John Socha
published by Brady Books
$19.95
ISBN: 0–13–661901–0

Inside the IBM PC
by Peter Norton
published by Brady Books
$21.95
ISBN: 0–89303–556–4

IBM PC & XT Assembly Language
by Leo Scanlon
published by Brady Books
$21.95
ISBN: 0–89303–575–0

DOS Reference Manual
Technical Reference Manual
DOS Technical Reference Manual
Contact your local IBM sales office.

Micro Cornucopia's RBBS
503–382–7643
24 hours, 300/1200/2400 baud
8 bits, 1 stop bit, no parity

■■■

New Disk From Cutsi–SIG

Disk 1313

RBBS.FBI 75K List of bulletin boards being monitored by the FBI.

RBBS.OK 3K List of bulletin boards not monitored by the FBI.

RBBS.BAD 50K All the popular RBBS numbers that were answered by incoherent groans. (Numbers were checked at 2 a.m.)

RBBS.LST 197K List of all the lists of RBBSs.

WELDING.ARC 20K Calculates length of discarded arc welding rods.

WORMER.EXE 640K Flushes worms from the system by filling up every bit of memory. (See the 4 meg version for those of you with extended memory boards.)

SQ79.EQE 0K The latest, greatest squeezer, far more effective than anything yet devised. Does not work with standard sq/usq and as yet we're having trouble with the companion UNSQ79. Before it squeezed itself, SQ79 was a 75K .EXE file.

ARC16.ARC 235K This is the ARC that everyone's been waiting, and waiting, and waiting, and waiting for. Requires ARC16 for deARCing. You should have fun with this.

SDIR270424.ARC 32K Yet another directory program.

SDIR270425.ARC 32K Yet another directory program.

SDIR270426.ARC 32K Yet another directory program.

SIDEWAYS.ARC 47K Prints spreadsheets wideways on the screen. So screen matches sideways printer output.

SDIR270427.ARC 32K Yet another directory program.

SDIR270428.ARC 32K Yet another directory program.

SDIR270429.ARC 32K Yet another directory program.

SDIR270430.ARC 32K Yet another directory program.

SDIR270431.ARC 32K Yet another directory program.

SDIR270432.ARC 32K The latest directory program.

CLEANUP.BAS 10K Adds line numbers and GOTOs to PASCAL source. (This program released to the public domain by Microsoft.)

CLEANUP.PAS 10K Removes line numbers and GOTOs from BASIC source. (This program released by Borland in response to Microsoft.)

BAGPIPEMAN.COM 33K This is the version of PIANOMAN for people who want the neighbors to know they've taken up music. Range is estimated at 12 miles.

DOGWHISTLEMAN.COM 33K Lets you practice without bothering your family. (Caution: In areas with high dog populations, neighbors will howl.)

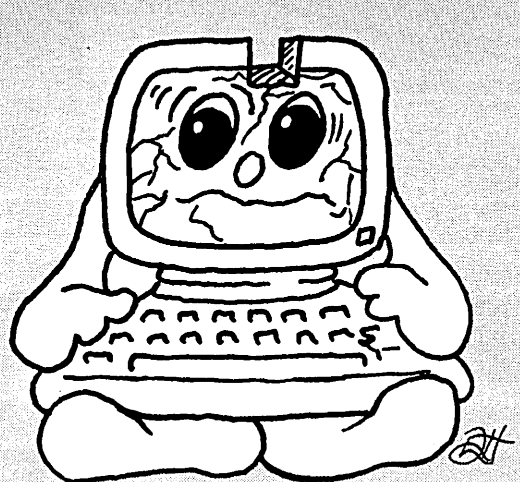
NOP34CDE.PQR 197K No one thought this would be done. Elle M. Enno combined the many functions of NOP version 34 with menu–driven user friendliness of CDE to come up with this incredible program. We know that it will be one of the hottest new public domain offerings.

3X5.ARC 12K The best full–screen notecard program we've seen. Reduces the screen work area to the size of a standard notecard. For a larger work area, see: LEGALSIZE.ARC, FILEFOLD.ARC, or BBS.ARC.

CRIPPLE.ARC 2K This is as good an example as any of the quality of software available for practically nothing. CRIPPLE is a double entry accounting system that's compact and mouse–driven. Those who want to expand the system can purchase the following modules from the author: Accts Receivable $695, Accts Payable $695, Inventory $695, Payroll $695, Printer Interface $695, Keyboard Input $695, Disk Interface $695 and Screen Display $695. There's a 5% discount for purchasing all the modules at one time, and financing is available through FHA.

DEFOOGU.ARC 24K This program encodes text files. Enter DEFOOGU and you are taken through a 27–step encoding routine that's designed confuse all but the most careful note–takers. To decode, you run the program backwards (enter UGOOFED<cr>) and enter all previous keystrokes in reverse order. This usually works.

DECODE.ARC 10K Because a few folks had trouble manually decoding DEFOOGU'd files, we created DECODE. Just run DECODE (filename). DECODE will automatically recover the original text without any operater entry. DECODE works with all files created by DEFOOGU.



Is There A Gap In Your Info?

Fill in your back issues of Micro C today

ISSUE #1 (8/81)
Power Supply
RAM Protection
Video Wiggle
½ PFM.PRN
16 pages

ISSUE #2 (10/81)
Parallel Print Driver
Drive Motor Control
Shugart Jumpers
Program Storage Above PFM
½ PFM.PRN
16 pages

ISSUE #3 (12/81)
4 MHz Mods
Configuring Modem 7
Safer Formatter
Reverse Video Cursor
FORTHwords Begins
16 pages

ISSUE #4 (2/82)
Keyboard Translation
More 4 MHz Mods
Modems, Lync, and S10s
Undoing CP/M ERASE
Keyboard Encoder
20 pages

ISSUE #5 (4/82)
Word Processing
Two Great Spells
Two Text Editors
Double Density Review
Scribble, A Formatter
20 pages

ISSUE #6 (6/82)
BBI EPROM Programmer
Customize Your Chars
Double Density Update
Terminal In FORTH
24 pages

ISSUE #7 (8/82)
6 Reviews Of C
Adding 6K of RAM
Viewing 50 Hz
On Your Own Begins
24 pages

ISSUE #8 (10/82)
SOLD OUT
Drive Maintenance
Interfacing D[illegible]
Installing [illegible] BIOS
Flip[illegible]ies
C[illegible]learly 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[illegible] BII
The Dis[illegible]ctor
JRT [illegible]
S[illegible] Keyboard Interface
Pascal Procedures Begins
36 pages

ISSUE #11 (4/83)
SOLD OUT
BBI Expansions
BBII Details
Dyna, R[illegible]sk Review
Eas[illegible]verse Video Cursor
Pla[illegible]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
Packet Radio
5MHz for Kaypro
40 pages

ISSUE #13 (8/83)
CP/M Disk Directory
More 256K for BBI
Mini Front Panel
Cheap Fast Modem
Nevada COBOL Review
BBI Printer Interface
Kaypro Reverse Video Mod
44 pages

ISSUE #14 (10/83)
BBII Installation
The Perfect Terminal
Interface to Electronic Typewriter
BBI Video Size
Video Jitter Fix
Slicer Column Begins
Kaypro Color Graphics Review
48 pages

ISSUE #15 (12/83)
Screen Dump Listing
Fixing Serial Ports
Playing Adventure
SBASIC Column Begins
Upgrading Kaypro II to 4
Upgrading Kaypro 4 to 8
48 pages

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

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

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

ISSUE #19 (8/84)
Adding Winchester To BBII
6 MHz On The BBI
Bulletin Boards
Track Buffering On Slicer
4 MHz For The 820-I
64 pages

ISSUE #20 (10/84)
HSC 68000 Co-Processor
DynaDisk For The BBII
Serial Printer On BBI Sans S10
Cheap & Dirty Talker For Kaypro
Extended 8" Single Density
72 pages

ISSUE #21 (12/84)
Analog To Digital Interface
Installing Turbo Pascal
Low Intensity BBI Video
Turbo Pascal, The Early Days
80 pages

ISSUE #22 (2/85)
Xerox 820-II To A Kaypro-8
Sound Generator For The STD Bus
Reviews Of 256K RAM Expansion
In The Public Domain Begins
88 pages

ISSUE #23 (4/85)
Automatic Disk Relogging
Interrupt Driven Serial Printer
Low Cost EPROM Eraser
Smart Video Controller
Review: MicroSphere RAM Disk
Future Tense Begins
86 pages

ISSUE #24 (6/85)
C'ing Into Turbo Pascal
8" Drives On the Kaypro
48 Lines On a BBI
68000 Versus 80x86
Soldering: The First Steps
88 pages

ISSUE #25 (8/85)
Why I Wrote A Debugger
The 32-Bit Super Chips
Programming The 32032
Modula II
RS-232C: The Interface
104 pages

ISSUE #26 (10/85)
Inside ZCPR3
Two Megabytes On DSI-32
SOG IV
The Future Of Computing
MS-DOS In The Public Domain
Graphics In Turbo Pascal
104 pages

ISSUE #27 (12/85)
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

ISSUE #34 (2/86)
Designing With The 80386
Build A Simple Oscilloscope
A Cheap 68000 Operating System
A Concurrent Operating System
Recovering Directories And FATs
On 360K Disks

BACK ISSUES OF MICRO C

U.S. Price............................$3.00
Canada/Mexico Prices....................$3.00
Foreign Price (air mail) $5.00

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

Command Processors, Taskmasters, & The PROLOG Programming Contest

Gary looks as some really interesting new additions for your system and he covers the newest, most unique Micro C contest, the PROLOG programming competition.

If you want a variation on the MS–DOS command line, you might try Command Plus from ESP Software, or Taskview from Sunny Hill. Command Plus replaces (and improves upon) COMMAND.COM, the command processor that comes with MS–DOS.

It can be executed as the boot shell (instead of COMMAND.COM) under MS–DOS version 3.0, or later, or as another program under COMMAND.COM if you're using an older version of MS–DOS.

Command Plus has an extended command file processor, command line editing, a more versatile COPY, and Browse, a file viewer.

Copy's variants include:

- Copy all files with times equal to or later than a specified date.
- Copy and create a subdirectory if it doesn't already exist.
- Copy but exclude all files specified by a specific time.
- Copy and notify, if about to copy over a file. And seven or eight more.

BROWSE is a much better TYPE, with commands for scrolling up and down lines and pages, and for forward and reverse searching.

SCRIPT, Command Plus's command file processor, adds several "structured" programming features to the Batch processing language that comes with DOS:

- IF and ELSE statements
- WHILE loops
- Boolean operators
- Case statements

It also includes a library of string and I/O routines. You can increment and decrement, move the cursor, write to files, and so on.

The command line is customizable and includes commands to search forward, backward, and undo. If the little things you screw up on the command line are driving you crazy, or if you need more power on that old DOS line, plunk down $79.95 for Command Plus or call ESP for information.

Taskview

Or check out Taskview, which lets you remove the command line entirely (when you want to), if you've got the memory and the mazuma ($69.95).

Taskview, a menu–driven shell, allows you to execute programs (or tasks) in shorthand (by letter or arrows) and easily switch between them without reexecuting.

Under Taskview command, you set up a menu of programs you want to be able to run. Then, by entering the first letter or by moving the cursor(highlight) with an arrow key, you run them. You can suspend a running program in execution, and switch back to it later or abandon it entirely.

When you suspend a program, it still takes up memory. So, Taskview manages the suspension by allocating memory and swapping programs to and from disk, when necessary. You can prioritize programs to control Taskview's swapping, and alter your specifications at any time, making this a flexible multitasking system.

I was able to suspend as many as 10 programs, which together used about a megabyte of memory, on my 640K system. I ran and suspended two expert systems which required 256K each, or 512K total (about the same memory you'd need to suspend dBASE3 while Lotus 1–2–3 was running). I then switched back and forth between them without restarting either program, using the responses from one system to determine moves on the other. This quasi–parallelism was surprisingly effective.

Since each of the systems consisted of several hundred rules, a session could become quite involved and time–consuming. Suspending execution saved me from restarting and recovering my moves in the program (a process that could take anywhere from a few seconds to more than a few minutes).

To suspend one 256K program and restart another took about 10 seconds on my X16 (an 8 MHz 80186 CPU running with a 20 Meg, 65 ms hard disk). If the expert systems had required just a little less memory, or if I'd had just a little more, I could have avoided disk swapping, and the time needed to switch between tasks would have been negligible (no more than a second). It's a quick, responsive system.

But let me emphasize that Taskview is not a concurrent system. Not even close. Tasks aren't RUNNING in the background; they're SUSPENDED in the background, WAITING. When you return from a switch to another program, all will be just as you left it, which sometimes has its advantages.

Taskview can make computing easier by helping you organize tasks, by saving you keystrokes, and by holding your place in programs. (You might think of Taskview as a much more complex SideKick: you can select the editor of your choice to suspend and use for notekeeping, for example.)

In addition, Taskview has a series of commands for cutting and pasting parts of (or entire) screens from one program running under Taskview to another. You can even write a screen to a file or to the keyboard input buffer, and use it as input for another program. Interesting.

For more info —
ESP Software Systems
11965 Venice Blvd, Suite 309
Los Angeles, CA 90066
213-390-7408

Sunny Hill Software
P.O. Box 5527A
Seattle, WA 98155
206-367-0650

The PROLOG Programming Contest

If you're planning to win fame and even a little fortune in the Micro C PROLOG programming contest, but are just getting one leg up, here are a few suggestions that might help you learn this bizarre (and very logical) language.

First, you'll need to buy an interpreter or compiler, but you don't need to spend much in order to have enough power to do some really neat things. The ADA public domain interpreter (available on Micro C MS-DOS disk #19 or from ADA) will do, and Borland's Turbo PROLOG is excellent. It's also a good buy ($65 at The Programmer's Connection, and $69 at The Programmer's Shop).

You'll also need at least one complete PROLOG programming reference. Buy or borrow one or more of these:

- *Programming In PROLOG* by Clocksin and Mellish. (Heavy going, but thorough. A good first source for standard PROLOG or Turbo.)
- The Turbo PROLOG reference manual. (Very good if you're programming in Turbo.)
- *Advanced Turbo PROLOG* by Herbert Schildt. (Good style, and useful as a second or third Turbo PROLOG reference.)

Then read and trace as many little programs as you can until you think you understand PROLOG backtracking, then study it a little more.

Make sure you understand how the cut and fail fits in.

Then write a little program and enter the contest.

Happy Trails. See you at SOG and we'll talk about it then.

For more info:
ADA
1570 Arran Way
Dresher, PA 19025
215-6466-4894

Borland International
4585 Scotts Valley Dr
Scotts Valley, CA 95066
800-255-8008

The Programmer's Shop
128-L Rockland St.
Hanover, MA 02339
800-421-8006

The Programmer's Connection
800-336-1166 ■■■

PASCAL PROCEDURES

By John P. Jones
6245 Columbia Ave.
St. Louis, MO 63139
314–645–1596

Elementary Modula–2

Whether you're learning Pascal or interested in moving from Pascal to Modula, this series will be a significant reference work. In this issue John takes an intense look at data types available in both languages.

As I mentioned in the last column, the next two or three columns will get back to the basics of Pascal as a programming language. There are two reasons for this. First, I know there are quite a few new readers of Micro C (or old readers newly interested in structured languages) who may need some review of the basics. Second, this review will also allow me to point out how Modula–2 differs from Pascal, and in the process, introduce you to programming in that language.

Basic Program Structure

- Pascal
  Header part
  Declaration part
  Statement part

- Modula–2
  Header part
  Import/Export part
  Declaration part
  Statement part

Pascal programs have three major parts, the header part, the declaration part, and the statement part. Modula–2 adds a fourth part between the header and declaration, the import part. The header part defines the program (or module) name, and for some Pascal compilers it must also define the allowed file I/O.

In the declaration part, all constants, user defined data types, and variables are defined. Unlike languages such as FORTRAN and BASIC, all variables in Pascal and Modula–2 must be declared before they can be used.

The variety and flexibility of data types provide both Pascal and Modula–2 with much of their strength. When data structures are carefully designed, the code is much easier to write.

All variable declarations for both Modula and Pascal take the form:

```
VARidentifier:type;
```

The type can be defined at the compiler level, user defined, or (in Modula–2) IMPORTed.

The Pre–Defined Data Types

There are a few data types defined at the compiler level. Most of these are common to both Pascal and Modula–2.

An INTEGER is a whole number between -32768 and 32767. Modula–2 also has an unsigned integer type, CARDINAL, with a range of 0 to 65535. These are generally used for loop control variables, counters, and simple whole numbers. Some compilers for both languages have a related type available, BYTE, which is an unsigned 8–bit INTEGER.

Numbers which can have fractional parts are REALs. REALs are a subset of the rational numbers; their range and precision depend on the compiler.

BOOLEAN variables can have the values TRUE or FALSE, and are used extensively to control program flow.

The type CHAR is used for character data, with the ASCII character set standard. Most compilers will allow any 8–bit value for a CHAR.

So far, we haven't improved much on other languages. The real power of data structuring shows up in the user defined data types.

User Defined Types

The simplest TYPE declarations define the new type as equivalent to a previously defined one. Variables of the newly declared type will be compatible with those of the old type.

```
TYPEWholeNumber  =INTEGER;
      YesNo      =BOOLEAN;
      Area       =REAL;
```

Enumerations

An enumeration type declaration defines its possible values. Each of these values is listed in the declaration.

```
TYPE  Room= (Living,Bath,Bed,
             Kitchen,Dining,Recreation);
Home= (SplitLevel,Ranch,
       TwoStory);
```

The scalar type created can be used wherever any scalar value can be used. Using an enumeration type for an array index variable can make the intent of a portion of code very clear. Compare the following:

```
VAR s  :ARRAY[0..5]OFREAL;
VAR LivingSpace: ARRAY[Living..Recreation]
                 OFArea;
```

Both arrays can be used to store the individual areas of the rooms of a house, but which will be easier to follow when you go back to the code after six months?

Subranges

When you declare a subrange type, you restrict the values to the range specified. There are syntax differences between Pascal and Modula–2 here, Modula requires brackets [] around the subrange. Subrange types are compatible with their base types.

```
(*Modula*)
TYPE  SmallRoom  = [Bath..Kitchen];
      SmallInts  = [-16..16];
```

```
UpperCase   = ['A'..'Z'];

(*Pascal*)
TYPE SmallRoom  = (Bath..Kitchen);
     SmallInts  = -16..16;
     UpperCase  = 'A'..'Z';
```

Subrange checking helps you locate many subtle problems.

Sets

A set type is declared as a combination of an enumeration or subrange type.

```
TYPE ValidInput  = SETOFCHAR;
     InThisHouse = SETOFRoom;
```

The first of these declarations would probably generate an error on most Modula–2 compilers, since sets are generally restricted to 16 elements unless extensions are IMPORTed. The predefined type BITSET is assumed if no type identifier is used in a set construct.

There are quite a few differences between Modula–2 and Pascal in the way sets are constructed and manipulated. We'll get into these as we address specific statements later on.

Sets are especially useful in situations where you need to exclude members of a subrange, or, put another way, only consider certain members of a subrange.

```
(*Modula-2*)
VAR  Included  : InThisHouse;
     ThisRoom  : Room;
     TotalArea : REAL;

Included:=InThisHouse{Living,
                      Bed,Kitchen};

FORThisRoom:=LivingTORecreationDO
  IFThisRoomINIncludedTHEN
    TotalArea:=TotalArea+LivingSpace[ThisRoom];
  END;  (*IF*)
END;  (*FOR*)

(*Pascal*)
VAR  Included  : ThisHouse;
     ThisRoom  : Room;
     TotalArea : REAL;

Included:=[Living,Bed,Kitchen];

FORThisRoom:=LivingTORecreationDO
  IFThisRoomINIncludedTHEN
    TotalArea:=TotalArea+LivingSpace[ThisRoom];
```

You will notice that Pascal and Modula use different delimiters in "set builder" constructs.

Arrays

An array is a collection of data of the same type, accessed through an index. An array can be of any type, but its index must be of a scalar type. Declarations are slightly different for Pascal and Modula–2.

```
TYPE Char20     = ARRAY[0..19]OFCHAR;
     AlphaGroup=ARRAY['a'..'z']OFSomeType;
```

The brackets are part of the array declaration in Pascal, but part of the subrange specification in Modula–2.

```
     (*Pascal*)
TYPE TwoDArray=ARRAY[0..3,0..15]OFINTEGER;
     (*Modula*)
     ThreeD=ARRAY[0..5],[1..4],[7..8]OFCHAR;
```

Records

A record declaration allows you to group data of unrelated types. Each element, or field, of a record is accessed via an identifier rather than an index. The full identifier for a field of a record is made up of the variable name, a period, and the field name. A record field can also be a record, in which case another period and field identifier will be needed.

```
TYPE Listing=RECORD
       Style:Home;
       Location:ARRAY[0..30]OFCHAR;
       LotSize:REAL;
       Rooms:SETOFRoom;
       Price:REAL;     (*orunreal*)
       Negotiable:BOOLEAN;
     END;
VAR  House:Listing;

House.Style:=Ranch;
House.Location:='Suburban';(*etc.*)
```

A record can have variants. A variant is a field which can be of an alternate data type. In Pascal, only the last field may be variant; Modula–2 is much more flexible since any field can be variant and they can even be nested.

```
TYPE BuiltOf = (Wood,Brick,Steel);
     Color   = (White,Brown,Red,Green,Blue);

     Listing=RECORD
       Style:Home;
       Location:ARRAY[0..30]OFCHAR;
       LotSize:REAL;
       Rooms:SETOFRoom;
       Price:REAL;
       Negotiable:BOOLEAN
       CASEConstruction:BuiltOfOF
         Wood:Paint:Color |
         Brick:Age:REAL |
         Steel:Rusted:BOOLEAN
       END;(*CASE*)
     END;(*RECORD*)

VAR  House:Listing;
```

Since the variant must be last in a Pascal record, a separate END for the CASE portion is not needed. Also, replace the vertical bars in the above declaration with semicolons in Pascal. Depending on the current value of House.Construction you can use the variables House.Paint, House.Age, or House.Rusted.

Record types allow you to group logically related data into a unit which can be manipulated just like any other data. Consider how much easier it would be to process an array of Listing records as compared to separate arrays for each of the fields. The time you spend designing a record for the data will be more than regained when you get to the coding.

Pointers

The value of a pointer type is the address of a value. Again, the declarations for Pascal and Modula–2 are different.

```
TYPEIntPtr  = ^INTEGER;(*Pascal*)
    IntPtr  = POINTERTOINTEGER;(*Modula*)
```

When a pointer variable is declared, only enough memory for the pointer itself is allocated; space is not assigned for the value being pointed to until it is created at run time with the procedure NEW.

Creating variables dynamically, this way, leaves open the amount of data you can process. Modula is a little strange here. Even though NEW is predefined, it can't be used unless ALLOCATE is imported from module Storage. This is because addresses are highly machine dependent, and some of the machine specific definition is in Storage.

A pointer can be a field in a record, and can be declared before the type pointed to is defined. This allows a

(continued next page)

(continued from page 69)

pointer to point to a record of which it is a field. With multiple pointer fields, some incredibly complex data structures can be generated (and sometimes even used).

The value pointed to is accessed the same way in both languages. The up arrow (^) is the dereferencing operator and is appended to the pointer identifier. (In the following, IntPtr is a pointer. IntPtr^ is the space in memory that IntPtr points to.)

```
NEW(IntPtr);    (*createadynamicvar*)
IntPtr^:=3;     (*assignavalue*)
IFIntPtr^>4THEN...
```

Variables

Before I talk a bit about the data types which are available in Modula-2 but not in Pascal, I'll review variables. Variables hold your program's data and they are declared with the general form:

```
VAR   Identifier:type;
      Identifier:TypeConstructor;
```

By TypeConstructor I mean most of what I've talked about already. The first form is preferred for three reasons. First, only variables of the same type will be strongly compatible.

```
TYPEIntArray10=ARRAY[0..10]OFINTEGER;
VAR Ar1  :  ARRAY[0..10]OFINTEGER;
    Ar2  :  ARRAY[0..10]OFINTEGER;
    Ar3  :  IntArray10;
    Ar4  :  IntArray10;
```

Ar3 will be strongly compatible with Ar4 but Ar1 and Ar2 will not be strongly compatible with each another.

I've already mentioned the second reason — well structured data leads to well constructed code. Finally, the formal parameter list of a procedure uses types, not type constructors.

Constants

Named constants are used for values which appear repeatedly in a program but which may need to be changed. Another reason for using a named constant is for clarity.

```
CONST   Pi = 3.14159;
        LinesPerPage = 66;
        TopMargin=3;
        PNPrefix='Page';
```

The type of a constant is deduced from the data. Although Modula-2 does not require constant declarations to be at the start of the declaration part, it's customary to put them there. This makes them easier to find.

Only In Modula-2

Modula-2 has some data types which are not available in most Pascal compilers. Procedures can be given a type, and thus can be declared as variables and used as parameters for procedures. Compatibility is determined by comparison of the parameter lists of the procedure types. This is structural compatibility. Very flexible programs can be written with procedure types.

A private or opaque type is a type declared in a DEFINITION module without a definition.

```
TYPEHidden;
```

The definition for the type is in the companion IMPLEMENTATION module. When imported by another module, it can be assigned, compared, and storage can be allocated. But nothing else can be done unless the procedures needed to manipulate the type are also imported.

Often, a module like a disk file handler will export one or more private types. This allows the details of the system to be hidden, while providing the use of its services.

Open arrays are declared in the parameter list of a procedure declaration.

```
PROCEDUREDemo(VAROpen:ARRAYOFListing);
```

The actual parameter passed to the procedure can be any array of a compatible type. Built in procedures for determining the actual number of array elements allow operations on varying size arrays without having to pass additional parameters.

In Closing

Now that we know what we have to work with (and it's a lot), we'll be able to look at the statements available for manipulating the data.

■■■

We can help you sell your products.

Hawthorne Technology

8836 S.E. Stark, Portland, Or 97216 (503) 254-2005

September 12, 1986

Micro Cornucopia
P. O. Box 223
Bend, Or 97709

Dear Laura;

Advertising in Micro Cornucopia has been very good for us. We have actually had response to an ad in Micro C on the same day the magazine hit the stands.

The contacts we make through our Micro C ads are of a higher quality than those we receive through ads we place in other publications. When people respond to a Micro C ad, they are really interested in our products, and not just curious.

I think the people who read Micro C read all of it. Our ads don't seem to get 'lost' among the other advertising.

You and all of the other people at Micro C are always so helpful and pleasant to work with. I expect we will have a long working relationship with Micro Cornucopia.

Sincerely,

Marla Z. Bartel
Vice President

For more information on great advertising opportunities in Micro C, contact Laura Logan or Julie Pope at (503)382-8048.

Concise Computer Electronics:

Important Considerations For Novice Project Developers

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

Bruce's "Controlling The Real World" series has been running in Micro C since last October, and if you've been following it you know he's covered some fairly complex subjects. If you missed its beginning, or still have some electronic gaps, maybe Bruce can help with some simple projects, as he talks about essential subjects like soldering, superstripping, wire-wrapping, sourcing and sinking current, calculating resistor values....

Most of the literature on electronics was written before microprocessors became cheap. It often assumes the most effective solution is one built completely in hardware.

I disagree. The best way to solve all but very trivial problems is to get the necessary information into a computer, dealing with it in software as soon as possible, then manipulating other hardware based on the results.

If you keep this notion in mind, it will serve you well.

Building One–Of–A–Kind Circuits

First, a few construction tips. I like superstrips: ICs, transistors, and other components plug right into and out of these fancy bread–boards. They're great for trying things. I use solid copper telephone wire for connections on superstrips because it's cheap and easy to find. All you have to do is push the chips, parts, and wires into the holes on the strips to make connections. Use some kind of continuity test to find out how the holes are connected.

For more permanent circuits, I use printed–circuit boards designed for prototyping. These have holes which are plated through the board. Some are even made with the same pattern as the superstrips, so you can directly transfer your circuit. Soldering to these boards is easy, and connections virtually never come loose. Some boards even plug directly into a PC bus.

When making a circuit permanent, always use sockets for chips. It's fairly easy to fry something when prototyping, and difficult to desolder a chip.

Soldering

My soldering iron is a cheap one. A small iron is good for electronic soldering. The solder should be small diameter, rosin–core, and made for electronic work. The rosin acts as a flux to clean the metal surfaces and promote a good bond. (Acid is also used as a flux when soldering pipes. Acid is corrosive and should NEVER be used on electronic circuits.)

Rosin flux tends to burn and form a dark coating on the tip of the iron. This makes soldering difficult.

To remove this layer, I take a paper towel, fold it up into a small square, soak it in water, and lay it on my work bench. I wipe the tip of the hot iron on this wet pad right before I solder a connection to give me a nice, shiny tip. (When the iron is new, it should be heated and the end should be covered with solder. (This is called "tinning," because solder is partly made of tin.)

To solder a joint, make sure there's a good mechanical bond first — the wires should be twisted together or held in place somehow. If what you're soldering is heat sensitive, clip some sort of "heat sink" on the wire between the joint and the device (this is another reason why sockets are nice — you don't have to worry about overheating the chip).

Place the shiny tip of the iron against the joint. Then touch the solder to the joint (not the iron). This way, the solder won't melt until the joint is hot enough.

You'll see the solder flow into the joint and become shiny and smooth. If you wiggle a wire as it cools or if it doesn't get hot enough, you'll see a dull crystalline pattern instead of shiny smoothness. This is called a "cold joint," and while the circuit will sometimes work, it isn't reliable.

Wire Wrapping

Another method of prototyping is called wire wrapping. This involves chip sockets with long square tails, and fine wire which is stripped and wrapped around the tails to make the connection. Entire mainframe computers have been wire wrapped. Many people swear by it for digital work, but I don't have enough patience to do it without an electric wire wrapping gun.

Basic Equipment

The tools I use most are: wire cutters, wire strippers (get very good ones — these get a lot of use), a power supply, and a multimeter. I work for Fluke so I have one of their multimeters (which are generally considered the best), but for beginning experimentation you can get almost anything.

Power supplies have become so common and cheap now that you shouldn't need to build one. You can find ads in any computer or electronics magazine, and Radio Shack often carries them. You can get away with just 5 volts DC, but it's good to get +/– 12 V also, for RS–232 and analog stuff. Get a supply which puts out at least 500 mA (mA = milliAmps = 10e–3 Amps = .001 Amps, so 500 mA = 1/2 Amp) at 5 V, since most of your power will be used for the digital circuitry. The +/– 12 V supplies can put out 50–100 mA.

It's also very helpful to have a good assortment of parts. I get resistors and capacitors in assortments from:

Digi–Key
PO Box 677
Thief River Falls, MN 56701

Ask for their catalog.

Variable resistors (potentiometers or "pots") are also useful for experimentation.

First Project

Begin with something fun and simple, like hooking up an LED. LEDs are good starters because they're one of the most useful indicators in digital circuits, letting you know visually when things are working right.

But LEDs can fool you; they aren't like light bulbs. With a light bulb, you put a voltage across it, and it takes just the right amount of current. An LED, on the other hand, doesn't know when to stop drinking — if you don't put some sort of external limiter on the current which flows through it, the LED will just draw more and more current until it self–destructs.

The magic number with LEDs is 20 mA. You shouldn't put more than this through them. They'll work with 10 mA, or sometimes less than 5 mA, and you won't see much difference in intensity.

There are two ways to assure this limit isn't exceeded: use devices which won't put out too much, or use a current–limiting resistor.

Often the device you're using (a logic gate or the outputs of an LSI chip) won't output more than a few milliamps, so you can hook the LED up to it directly without worry. You can either check the data sheets (more on that later) or, if you have a large supply of LEDs, just hook it up and see whether it works or fries itself.

Editor's note: I'd be careful about hooking LEDs up to standard ICs. The IC may self–destruct if an LED is taking every last milliamp. Special ICs have outputs which were designed to drive LEDs. Otherwise I'd suggest using a series resistor along with an IC or transistor that's rated for 20 ma or more.

Sourcing VS Sinking Current

We've grown up in an era of copious electricity. It's everywhere, and is available in almost unlimited quantities.

Integrated circuits (ICs) are different: they can only put out so much current (regardless of what the rest of the circuit will allow), and the direction of the current is important.

Some devices, for example, will "sink" (draw into themselves) several milliamps, but will "source" (put out) almost nothing. If you hook an LED to one of these chips from an output to ground, you won't see anything when the output is high because it can't provide enough current. To see something in this situation, you must connect the LED between the output and +5 V so that when the output is low, the device will sink enough current through the LED.

Data Sheets

The best way to avoid all this trial and error is to get your hands on the data sheets. These come in books, and the best two for simple digital ICs are a TTL data book and a CMOS data book. Before you run out and pay too much at an electronics store or through mail–order, try to find a used one (you probably don't need the most up–to–date) at a nearby engineering firm or electronic parts house.

Looking through these books will give you a case of information overload. They're designed to tell you absolutely everything about a chip: how fast it switches, how much power it requires, how many bad ones you'll find in a lot, how it changes with temperature — on and on.

But don't panic if you don't understand all the charts and graphs. You don't need to. Look for:

- A pin–out of the package so you know how to hook it up
- Words on one of the charts which say "high–level input current" and "low–level input current," which will tell you whether it will drive your LED.

Ignore the "–" signs on the numbers; they refer to a convention for current flow. If the high–level input is in microamps (10e–6 amps) and the low–level input is in milliamps, you'll have to connect your LED from +5 V to the output; the LED will turn on when the output is low. If the output is less than 2 mA, you may have trouble turning on your LED, and if it's more than 20 mA (rare, except for devices which are specifically designed to drive a lot), you may burn it out.

If you discover your outputs won't drive an LED, you'll need to send the signal to something that will — usually an inverter (more about those later). Hunt through the book until you find inverters which will source or sink enough current.

An additional advantage to data books is that they let you browse through and see what chips exist. Often they have examples of how to use them (application notes).

Current–Limiting Resistors

If your chip puts out too much current, or you're just connecting an LED directly to 5 V as a power on indicator, you need to use a resistor so the LED doesn't drink too much current and die.

To calculate the value of the resistor, first figure out the voltage difference. To do this, take the voltage coming out

(continued next page)

(continued from page 73)

of the device and subtract the voltage at the other end (usually power or ground). If you're getting current from the output of a TTL device, its high value is between 3.5 and 4 volts.

If you're getting current from the supply at 5 V and pulling it through the LED into a TTL device with a low output (slightly less than 1 V), you'll have a voltage difference of about 4 V.

CMOS devices are more what you'd expect: almost 5 V for a high, and almost 0 V for a low, so the voltage difference between the output of a CMOS device to power or ground is about 5 V.

When you know the voltage across your LED, you can calculate the resistor value using Ohm's law (V = IR).

Editor's Note: For an introduction to Mr. Ohm and his Law, see "Slowing Your Fan," in Micro C issue #34.

We know we want around 10 mA through the LED. So, from 5 V to a TTL "low," R = V/I = 4 V/10 mA = 4 V/.010 = 4 V X 100 = 400 ohms. Grab the value closest to this — it doesn't need to be exact. As a second example, from a TTL high to ground, we have R = 4 V/10 mA = 400 ohms. From 5 V to ground, R = 5 V/10 mA = 500 ohms.

The Resistor Color Code

Resistors are marked with bands of color. The band closest to the edge is the first digit, the next one in is the second digit, and the third is the multiplier (ten to the power of the third band). The fourth one is the tolerance, indicating how close you can expect the actual and marked values to be (don't worry about tolerance — digital is quite forgiving). High-accuracy resistors have more bands.

I use a rather risque ditty to remember the color code but when I mentioned it to my girlfriend, she was shocked. So I'll give you a cleaner, but significantly less memorable variation.

The easiest way to remember something is to associate it with a strongly impressive metaphor (they did this a lot in the middle ages, before books). You probably won't remember: Black (0), Brown (1), Red (2), Orange (3), Yellow (4), Green (5), Blue (6), Violet (7), Gray (8), White (9), but you will remember something like Big Bombs Rip Off Yellow Garages, Bring Very Great Winds (I sometimes count on my fingers while I'm reciting this).

An example: a 1K-ohm resistor: 1000 ohms. The first digit is "1" (Brown), the second is "0" (Black) and there are two more decimal places to account for, so the multiplier is 2 (Red). 10K would be Brown, Black, Orange. 10 ohms would be Brown, Black, Black. The fourth band indicates tolerance, and is usually silver (10%) or gold (5%).

About Diodes

Diodes let current flow one way, but not the other. They can be used to turn an AC (alternating current — periodically flowing one way, then the other) signal into DC (direct current — only flows one way). The diode symbol is an arrow pointing into a bar. Our LED (Light Emitting Diode) has this symbol, plus a circle around it, and sometimes little light rays coming out.

The arrow indicates the direction of conventional current flow (from positive to negative). If the arrow points towards the minus, current will flow. Current won't flow the other way.

I imagine the bar as a wall — the arrow helps the current over the wall, but it can't go back over the other way. It just spreads out at the bottom of the wall when it tries.

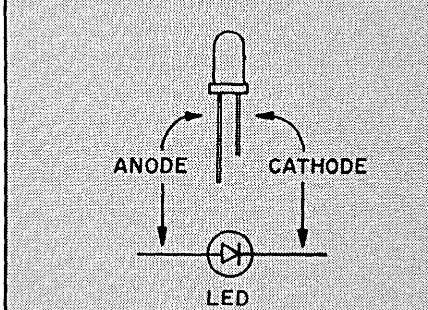
The arrow side is called the anode (often marked A), and the wall side is called the cathode (often marked K). I remember this because current flows from the arrow (anode and arrow start with A) over the wall.

An LED may have a flat spot on the plastic part to indicate the cathode. This is easy to remember because the wall and the flat spot look the same. Nearly all LEDs have a long leg and a shorter one. The longer one is the anode. See Figure 1.

A Simple Flasher Circuit

For a simple project, let's build an oscillator. Digital circuits are usually driven by some kind of clock (to move the circuit through its states), so you won't go very far before you build one.

Figure 1 - Identifying LED Leads



The circuit in Figure 2 is one of the simplest oscillators you'll ever find, yet it's quite commonly used. The chip used is a CMOS 4049 hex inverter ("hex" indicates there are six inverters on the chip — we're only using three).

An inverter is just about the simplest logic gate: it takes a logical 1 and turns it into a logical 0, and vice-versa. The triangle means the output is the same as the input, and the bubble means take whatever comes out of the triangle and invert it.

Logic Families

The concept of a logic gate (ANDs, ORs, NOTs, etc.) can be physically implemented by hooking up a circuit (inside a chip) in a number of different ways. Each of these connection schemes is referred to as a logic family, and has different voltage characteristics (i.e.: the voltages which mean logical 1s and 0s are different from family to family). These defining voltage characteristics are called logic levels.

There are many different logic families, but most of them exist only in textbooks. The most commonly used families are TTL and CMOS.

CMOS

A little about CMOS: you may be used to thinking about everything digital running from 5 V, like your computer. CMOS works at 5 V, but it also works down to 3 V and up to 15 V. It doesn't use much power. Unlike TTL (which, for a 5 V supply has logic levels of 0.8 V for zero and 4.0 V for one), its logic levels are quite sensible — the transition is in the middle of the voltage range (for example, with a 9 V power supply, devices switch from 1 to 0 or 0 to 1 at 4.5 V) and the 1 and 0 values are quite close to the supply voltage and ground, respectively.

CMOS does have a drawback or two: older designs (the "4000-series") tend to be static sensitive. i.e. you can destroy a chip by shuffling across a carpet and picking it up. Keep CMOS in conductive foam and ground yourself before moving them.

If you aren't using an input to a gate, you have to tie it to the supply voltage or ground (no uncommitted inputs). Left alone, the inputs tend to float to the transition voltage right in

Figure 2 - Simple Oscillator

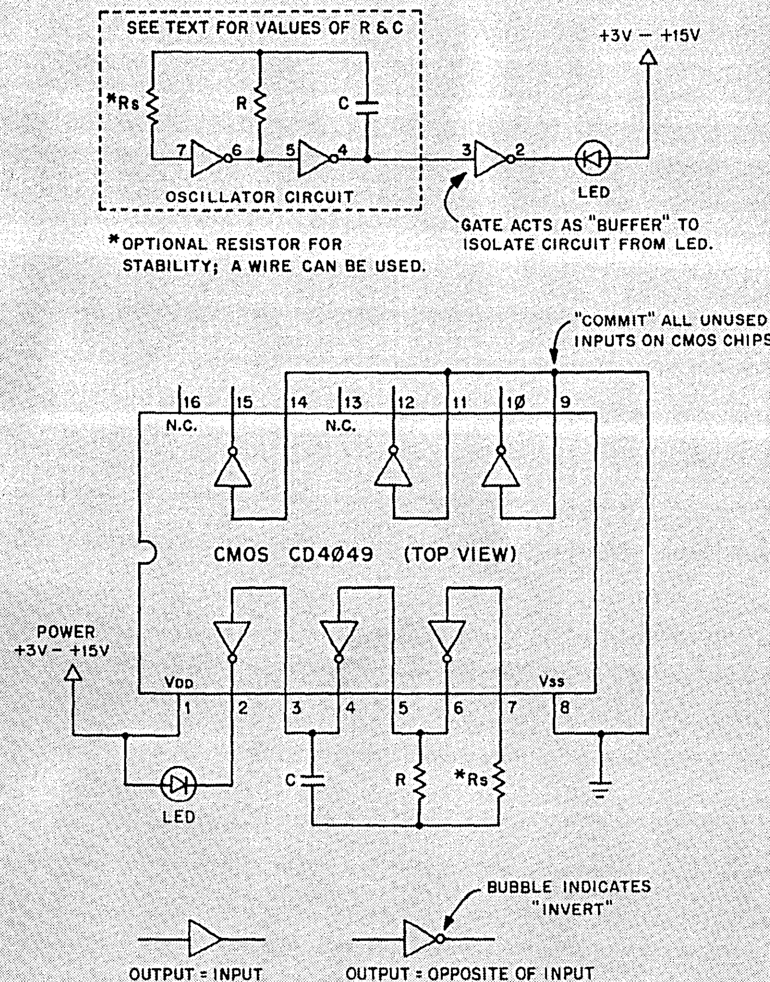
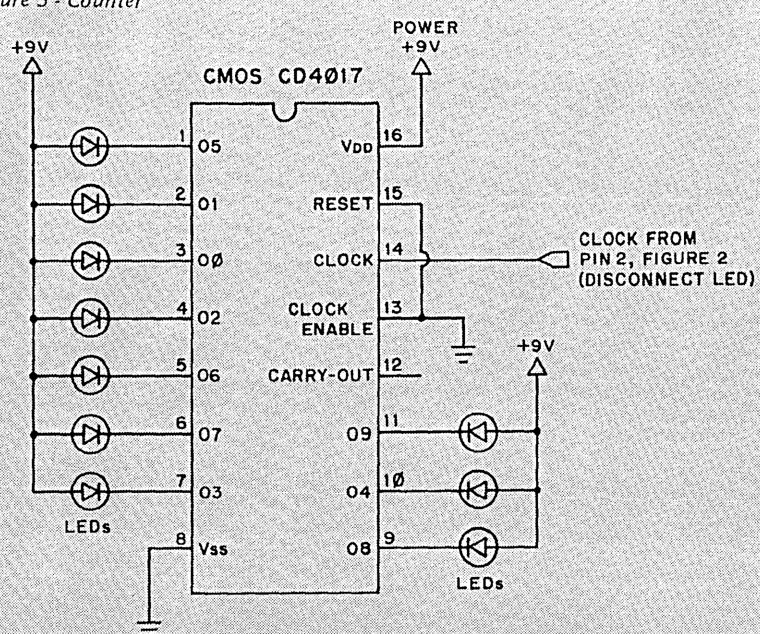


Figure 3 - Counter



the middle. This causes the chip to draw lots of current and burn itself out.

I've always had a bit of trouble with the labeling. I like "+5 V" and "GND." CMOS is so versatile they say instead "Vdd" (drain) and Vss (the chip's substrate). Hook power to Vdd and ground to Vss.

Using CMOS in the clock circuit is nice since you can use a 9 V battery instead of waiting for a power supply to arrive in the mail. (Or you could tap a line off your computer's power supply.)

The Clock

The clock circuit works by charging the capacitor until the voltage at the input inverts, which causes the capacitor to be discharged until the cycle repeats. The frequency of oscillation can be roughly estimated with the RC time constant.

This is a measure of how long it takes current through a resistor to create a certain voltage across a capacitor. It takes about 3 RCs to get within 10% of the supply voltage, but the CMOS threshold is passed during the first RC, so one RC is a good estimate of the period. (Just take the value of R and multiply by the value of C. For example, R = 1 megohm and C = 1 microfarad gives an RC of 1 second. Be careful with the multipliers.)

Connecting A Counter

As a final example, connect the output of the oscillator to the 4017 counter as shown in Figure 3. This is called a decade counter/divider with 10 decoded outputs. This means it counts to ten and then starts over, and only one of its ten outputs is high at any one time. In the circuit shown, only one LED will be off at a time.

A circuit like this is useful for sequencing events. If, for example, you were using a one-second clock and wanted to: start the VCR, wait 4 seconds for it to come up to speed, start the camera panning motor, then turn on three lights in sequence, a circuit like this could be your controller (although in many applications it'd be more sensible just to throw a whole computer at the problem).

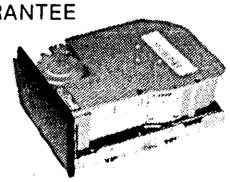
Finale

I hope these thoughts have filled a few gaps or scattered a few cobwebs. Next issue, we'll push further into the depths of computer electronics. ■■■

TI 745 TERMINAL....... $\$149^{00}$
—THERMAL PRINTER
—USED, TESTED, GOOD
—BUILT-IN MODEM (300 BAUD)

—HARD DISKS—

SEAGATE ST225............. $\$399^{00}$
—WITH CONTROLLER
—20 MB ½ HEIGHT
—1 YEAR GUARANTEE



CMI 5619............... $\$199^{00}$
15 MB FULL HEIGHT
W/CONTROLLER .... **$\$249^{00}$**

—HARD DISK CONTROLLERS—

XEBEC 1210A................. $\$79^{95}$
—FULL LENGTH
—CONTROLS 10 & 20 MB
—CUSTOM CONFIGURATIONS AVAILABLE

DTC 5150CX............. $\$99^{00}$

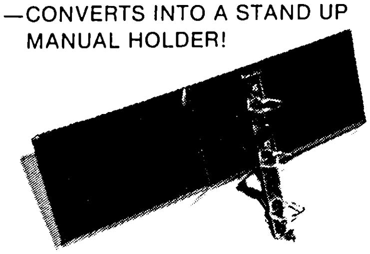
OMTI 5520.................. $\$124^{95}$
—WITH CABLES
—SHORT SLOT
—CONFIGURABLE BIOS
—RUNS ANY 3" OR 5" HARD DISK
—SOFTWARE
—SUPPORTS DRIVES UP TO 320 MB!!

HAZELTINE 1500 TERMINAL
USED, TESTED, GOOD
$\$89^{95}$

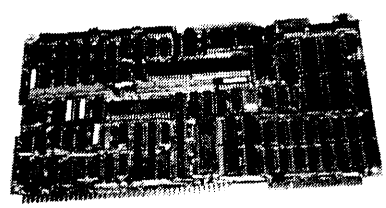
VARACTOR TUNER
$\$7^{95}$
—UHF/VHF
—133 CHANNEL CABLE READY
—VOLTAGE CONTROLLED CHANNEL SELECT
—DOCUMENTATION INCLUDED!

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!



DAISY WHEEL PRINTER. $\$179^{00}$
—PARALLEL INTERFACE
—18 CPS
—LETTER QUALITY



INTEL 86/30............ $\$395^{00}$
—8086 MONOBOARD COMPUTER
—MULTIBUS
—NEW, SCHEMATICS INCLUDED

XEBEC 1210A
HARD DISK CONTROLLER
FOR IBM PC/XT

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

$\$79^{95}$

— 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.. $\$4^{95}$

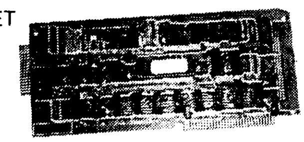
5¼" MPI FULL HEIGHT........ $\$49^{95}$
—DOUBLE SIDED
—IBM COMPATIBLE

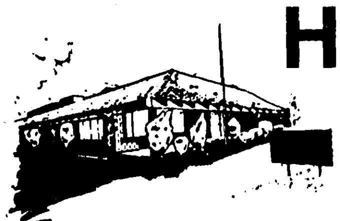
TOSHIBA 1.2M............... $\$139^{00}$
—'AT' DRIVE
—1.2M CAPACITY

TEAC 55B .... $\$109^{95}$
—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.

C'ING CLEARLY

By Ron Miller

1157 Ellison Dr.
Pensacola, FL 32503

Resident Programs

Ron has outdone (overdid?) himself with this column. I couldn't even sneak onto the Micro C RBBS for a month after we put up the TSR (terminate but stay resident) files. It's a hot topic, and Ron has discovered the secret MS-DOS internals that make it all possible. Don't miss this.

In my last column we used C to gain access to CGA and Hercules graphics. In the process we changed pixels without going through that slow, slow, slow operating system.

And I pledged, this time, to illustrate graphics screen capture and editing in C. I'd like to keep half the bargain, since the tricks involved in making resident .EXE files and interrupting DOS are more than enough for one session of C and Tell.

Why Programmers Get Gray

There are three situations where programming gets nasty.

The first and most common appears on those days when you just can't do anything right: you can't find missing semicolons, you habitually misspell the names of variables, you can't keep it in your thick head that equality tests in C require not one but two equal signs.

The second situation arises when coding is inherently complicated. I know it's going to be Excedrin Time whenever I start manipulating array indices — when modifying argv, say. Whenever a routine begins with a "char ***argvptr" declaration, I expect to scurry back to the compiler at least five times.

Even this second brand of headache can be dealt with, though, because the solution is built into the language. The game is difficult, but fair. A break, a cup of coffee, and a glance back at Kernighan and Ritchie are usually enough.

The third situation is the most frustrating — mucking about in areas where the rules aren't very clear.

Resident Joker

For instance, how do you write to disk from inside a resident program?

I'll bet I've read two dozen articles patiently explaining in grammar-school tones that DOS cannot be reentered — i.e, if an interrupt 21h is in progress when a resident program grabs control (and it often is), any call from within the resident program to send a letter to the screen, or a line to the printer, or a sector to a file, is guaranteed to crash the system. Yet anyone who's used SideKick knows it can be done.

Maybe SideKick doesn't use DOS to write to the screen, and maybe it even runs off with the keyboard without DOS finding out; but when Borland and Company write notepad files to disk, there just ain't no way to avoid a big, fat interrupt 21h. (Don't tell me now difficult the game is, just tell me how to play it.)

Well, I've found a way to make a C program resident and capable of using all the printf's, fprintf's, scanf's, and write's your heart could desire. Using one of MS-DOS's undocumented functions (See? They don't want us to succeed!) and about a zillion hours of rebooting the computer, I've waded through the most frustrating programming of my life to come out grinning on the other side.

Forbidden Knowledge

A year or so ago a friend mailed me a handy list of DOS interrupt services he'd downloaded from some bulletin board. In a moment of idle reading (probably as a break while trying to find an unmatched curly bracket) I noticed that interrupt 21h, function 34h, returns (in ES:BX) an address to a two-byte flag which is zero when it's safe to interrupt DOS. A dash to the bookshelf revealed that the folks from Microsoft Press always say this service is internal or reserved.

"Reserved for whom?" I wondered, quite to myself.

I began to imagine an "Index Of Forbidden Interrupts," with Bill Gates as the Grand Inquisitor and Peter Norton as the obedient layman who gains his living from the church.

Unfortunately, the forbidden knowledge seemed useless at first. (Like an Apple?)

I set up a tiny resident program, hit the hot key inside another transient C program, and always got a non-zero response. Pause...cup of coffee...load the dishwasher... OF COURSE, idiot: the keystroke was itself called by DOS! Why not set an internal flag at the keystroke — one that doesn't involve DOS — and let the clock tick poll the DOS-is-willing flag 18.2 times a second until it goes to zero? Then, by George, I should be able to enter DOS to read or write files.

Well, it worked, after lots of tries and a few other discoveries which no one ever bothered to tell me. Polling the flag was the easy part. The initialization program needs only to contain code like:

```
rr.ax=0x3400;/*orhoweveryourChandlesintrpts*/
interrupt(0x21,&rr)
safeseg=rr.es;
safeofs=rr.bx;
```

with "safeseg" and "safeofs" as globals that will stay initialized in the resident code. Then just poll the flag with a

```
peek(safeseg,safeofs,&flag,2);
```

whenever the clock ticks and interrupt 0x1c is called. With a "long" C, a static pointer to ES:BX will suffice.

Complications And More Mysteries

Unfortunately, there's no way to write a resident program without using assembly language, and thus you have machine and compiler–dependent routines.

Initialization code in C can diddle with the interrupt table, changing addresses and relocating the old vectors (for chaining). But to save stack pointers and registers in the code segment, you need to talk to the CPU one–on–one.

After much hair–pulling (why is it that push and pop look the same on a CRT?), the machine quit locking up. My resident C code could write "Hello world" (what else?) with a puts(), or beep rudely with a putchar('\7') with nary a glitch.

Opening or closing a file, however, still sent it north–north–west. After about six weeks of code staring it occurred to me that my own code, (like DOS) was non–reentrant.

As long as the clock didn't tick twice, I popped in and out of my handler safely. But when a second stack pointer and stack segment were moved into the code–segment storage area containing the first, the initial return was seriously inconvenienced. I immediately set another flag to keep the snake from swallowing its own tail.

With that I could read and write lengthy files and store video screens at will. For two giddy hours I thought I had solved the Riddle of the Sphinx. Then, getting fancy, I tried to use scanf() during the interrupt — which generated another long reach for the switch at the back of the computer. The Sphinx had a fallback position.

Priorities

This one Microsoft Press helped me on. Ray Duncan notes in "Advanced MS–DOS" that after you use hardware interrupts you must reenable things with an "EOI" to the timer chip. If you don't, interrupts of "lower priority" will remain blocked. That means sending a 20h to port 20h.

So I tried an outb(0x20,0x20), and, lo and behold, the keyboard returned. Though I've never located a priority table ("The Protocols Of The Elders Of Intel"?), I surmise that somehow file writing is high enough up on the Great Chain of Being to beat out the keyboard. Certainly the PIC (programmable interrupt controller) had been waiting for an end of interrupt (EOI) while I fumbled with the clock interrupt. Come to think of it, with access to my handler blocked, the clock could go on ticking at will.

No doubt SideKick does it with more finesse, because there is often a slight delay in my programs until the DOS–is–willing flag goes to zero. When the program is just sitting there waiting for a key entry — on a command line, say — the interrupt may not occur until something is whacked. However, for commercial programs, which seldom use interrupt 21h for character input, the response seems instantaneous.

Generic Resident

I seem to be able to do anything in a resident program that I can in a non–resident program, except use the "exec()" function to call other programs. I'm working on that, though the stunt requires playing with DOS's allocation chain directly — which Microsoft definitely thinks is bad form.

No doubt commercial programs that play games with the interrupts below 10h (SideKick, for instance, and XyWrite) will screw things up spectacularly, but with relatively civil programs such as Lotus and WordPerfect and the Turbo Pascal compiler, things go smoothly with my resident .EXE files. You should be able to use the routines in Figure 1 (with minor modifications) to make just about any C program resident.

I've recently written a nifty little resident routine that removes subsequently loaded resident programs at the touch of a hot key. Just refresh the interrupt vector table, hop up the allocation header chain with a long C pointer, and lop things off where you want them lopped.

No doubt I could do this in assembly language if I really felt masochistic, but C code is SO much easier. My strategy is to get out of the pushes, pops, and moves as quickly as possible, and let the compiler keep the stack straight.

At Last, A Screen Capture Program!

The code in Figure 1 gives both the C and the assembly language routines which must be linked for the screen capture routine to work. I hope the comments explain the individual steps. Notice that this routine:

- Automatically checks to see whether a CGA or Hercules system is in place, adjusting the graphics screen sizes accordingly. Since I primarily use this program to capture Lotus displays, the graphics capture is from page 1 rather than page 0 for the Hercules option.
- Captures the screens immediately but waits until the coast is clear to write them from the intermediate buffer to the disk.
- Numbers the captured text screens from "text00.pik" to "text99.pik" and the graphics screens from "graf00.pik" to "graf99.pik." You can also reset the counter to zero and name of next file to anything you wish.

I chose these keys because they don't interfere with Lotus. It would, of course, be trivial to change them to something else, or to limit yourself to one. Since I was filtering interrupt 16H, I decided to switch nulls for the hot keys because a null character return isn't likely to do wild things to any commercial program.

Do It Yourself

The code offered here is probably a worst–case option, where you have to set up your own stack and use an internal storage buffer. If you use a long C, you'll be able to use the calling program's stack instead of setting up your own. That's much easier, because it demands less assembly language.

If you've followed me this far, you can surely design a simple C program which uses the graphics routines from the previous "C'ing Clearly" column and tosses the contents of these files to screen memory in order to create a slide show or demonstration. From the magazines, I see that vendors are charging fifty dollars or more for this sort of thing. You certainly learn a lot by doing it yourself.

I won't THINK about how many dollars' worth of my time I spent saving those fifty bucks. Things certainly would have been simpler and quicker if someone had just told me about DOS entry flags and interrupt priority levels.

(Listings begin on page 80)

■■■

Figure 1 - Resident Screen Capture

```
/*********************************************************************/
                /*Ccodesection*/
#defineclockint0x1c
#definekeyint0x16
#definenewclock0x68          /*newclock&keyboardinterrupts*/
#definenewkey0x69
#include<regs.h>              /*interruptregisterdeclarations*/
externunsigned  _ps,_data;/*segmentsforCcode*/
unsigned safeseg,safeofs,scrseg,extent,fnum;/*globalsinitialized*/
char*hold,*filename,name[35];           /*forresidentrunning*/
structregsrr;
/******************/
busy()        /*returns0ifDOSsafetointerrupt*/
{
   int_busy;

   peek(safeseg,safeofs,&_busy,2);  /*testDOS-safeflag*/
   return_busy;
}
/******************/
fileit()              /*calledbyclockinterrupt*/
{
   if(!busy()&&extent){     /*notbusy&hotkeypressed?*/
      outb(0x20,0x20);      /*sendEOItoPIC—keepclockontime*/
      if(extent==1)nameit();elsewriteit(extent);
      extent=0;     /*enablenextkeystrokecommand*/
      }
}
/*****************/
nameit()              /*letsyounamethenextscreenfile*/
{
   rr.ax=0x300;                    /*getcursor*/
   rr.bx=0;
   interrupt(0x10,&rr);            /*cursorstoredinrr.dx*/
   peek(scrseg,0,hold,4000);    /*savescreen*/
   clr('s');
   puts("\n\n\n\nWhatFILENAMEforthesavedscreen?");
   gets(name);
   poke(scrseg,0,hold,4000);    /*restorescreen*/
   rr.ax=0x200;         /*setcursor=rr.dxfromprevious"getcursor"*/
   rr.bx=0;
   interrupt(0x10,&rr);
}
/*****************/
writeit(run)    /*writesfilefromscreen-storagebuffer*/
   unsignedrun;
{
   intfin;

   if(!filename){     /*justusedefaultsifnonamedetermined*/
      filename=(run==0x1000)?"text00.xxx":"graf00.xxx";
      filename[4]='0'+fnum/10; /*digitssettonumber*/
      filename[5]='0'+fnum++%10;
      }
   fin=creat(filename,0);
   write(fin,hold,run);
   close(fin);
   filename=0;
   puts("\7\7");
}

/*****************/
chkey(scan)    /*checkskeystrokesandsetsflags:calledbyint16h*/
   unsignedscan;/*char+scancodefromint16hcall*/
{
   if(extent)returnscan;   /*topreserveunfinishedcommand*/
   switch(scan){
      case0x5e00:extent=0x1000;/*textscreen—>ctl-F1,etc.*/
         break;
    case0x5f00:extent=(scrseg==0xb000)?0x8000:0x4000;/*graphics*/
         break;  /*C/Guses4000h,Herculesuses8000h*/
      case0x6000:filename=name;/*wanttosetnameofnextfile*/
         extent=1;
         return0;/*swapanull&returnittoapplicationsprogram*/
      case0x6100:fnum=0;   /*resetnumberingtozero*/
      default:returnscan;
    }
  if(extent!=0x8000)peek(scrseg,0,hold,extent);
     elsepeek(scrseg,0x8000,hold,extent);/*Lotususespage1*/
  return0;
}
/****************/
chgint(old,new,fnct)
   intold,new,(*fnct)();
{
   intinthold[2];              /*temporarystorageforinterruptvectors
   nicerprogramswoulduseint21h,fns.25h&35h;lesscodethisway*/
   peek(0,4*old,inthold,4);  /*getoldvector*/
   poke(0,4*new,inthold,4); /*placefurtheroutintableforchaining*/
   inthold[0]=fnct;           /*substituteourhandler*/
   inthold[1]=_code;
   poke(0,4*old,inthold,4);
}

/****************/
main()
{
   intkey(),clock(),test;
   char*malloc();

   peek(0,4*newclock,&test,2);
   if(test){
      puts("\n\n\7\7Interruptsinuse.Can'tinstallresidentprogram.\n");
      exit();
      }
   interrupt(0x11,&rr);
   scrseg=((rr.ax>>4)&3==3)?0xb000:0xb800;/*whichadapter?*/
   rr.ax=0x3400;                         /*getDOS-safeaddress*/
   interrupt(0x21,&rr);
   safeseg=rr.es;
   safeofs=rr.bx;
   chgint(clockint,newclock,clock);/*swapclockinterrupt*/
   chgint(keyint,newkey,key);        /*swapkeyboardinterrupt*/
   hold=malloc(0x8000);                  /*allocatestoragespace*/
   store((unsigned)hold+0x8100);   /*setstorageincodesegment:*/
                                       /*100hstack*/
   puts("\n\nScreenCaptureResidentProgramInstalled.\n\n");
   puts("Usage:\n");
   puts("  Ctl-F1     =capturetextscreen.\n");
   puts("  Ctl-F2     =capturegraphicscreen.\n");
   puts("  Ctl-F3     =setfilenamefornextcapturedscreen.\n");
   puts("  Ctl-F4     =givenametonextfile.\n");
   rr.ax=0x3100;      /*exit,stayresident*/
   rr.dx=_data-_ps+((unsigned)hold+0x8100)/0x10;
   interrupt(0x21,&rr); /*reservetol00habovestorageforinternal
      stack:(data_seg-psp_seg)*10h+buffer_size+100hofextrastack*/
}
```

```
/*******************************************************************/
/**        Assembly language for linking **/

CODE        SEGMENT PUBLIC BYTE
            ASSUME      cs:CODE

            PUBLIC      _clock,_key,_store


newclock    equ    68h
newkey      equ    69h


c_ds        dw     ?          ;storage for ds,es,ss
c_sp        dw     ?
callss      dw     0          ;also used as entry flag for handlers
callsp      dw     ?
__hold      db     ?
__pop       dw     ?


_store:                       ;storing regs in code segment
     mov    cs:c_ds,ds        ;function called from C code
     mov    bp,sp
     mov    ax,[bp+2]         ;get argument of store(arg)
     mov    cs:c_sp,ax        ;initial stack pointer for resident code
     ret


_clock:                       ;clock interrupt handler
     pushf
     cmp    cs:callss,0       ;if interrupt routine in progress, pass by
     jnz    skip1
     call   to_res
     call   _fileit           ;call C routine that writes to file
     call   from_res
skip1:popf
     int    newclock
     iret
_key:                         ;keyboard interrupt handler
     mov    cs:__hold,ah      ;preserve calling service
     int    newkey
     pushf                    ;save flags
     cmp    cs:__hold,0       ;test calling service
     jnz    skip2             ;if not service 0, pop out
     cmp    cs:callss,0       ;if a routine in progress, pass by
     jnz    skip2
     call   to_res
     call   _chkey            ;call C routine that checks for hot keys
     pop    cx
     push   bx                ;substitute return. My C uses bx as transfer
     call   from_res          ;register. If yours uses ax, push ax
skip2:popf
     db     0cah,02,00        ;not IRET but RET FAR 02
                              ;to preserve flag return


to_res:                       ;save registers & go to resident program
     pop    cs:__pop          ;save return IP, since stack is to be changed
     mov    cs:callsp,sp      ;save calling stack
     mov    cs:callss,ss
     cli                      ;disable interrupts
     mov    ss,cs:c_ds        ;set up resident C stack. Your C may not
     mov    sp,cs:c_sp        ;set ss=ds. Adjust accordingly.
     sti                      ;enable interrupts
     push   bx
     push   cx
     push   dx
     push   di
     push   si
     push   es
     push   ds
     push   bp
     push   ax                ;AX last pushed so can be passed to the
                              ;C program with a mov XX,[bp+2] routine
     mov    ds,cs:c_ds        ;set up resident data segment
     mov    es,cs:c_ds
     push   cs:__pop          ;restore return IP to new stack
     ret

from_res:                     ;restore caller's stack & regs and return
     pop    cs:__pop          ;see above on return IP logic
     pop    ax
     pop    bp                ;restore caller's registers
     pop    ds
     pop    es
     pop    si
     pop    di
     pop    dx
     pop    cx
     pop    bx
     cli                      ;disable interrupts
     mov    sp,cs:callsp      ;caller's stack restored
     mov    ss,cs:callss
     sti                      ;enable interrupts
     mov    word ptr cs:callss,0  ;if not zero, resident code closed out
     push   cs:__pop
     ret

     EXTRN       _fileit:NEAR
     EXTRN       _chkey :NEAR

CODE    ENDS

        end
```

End of Figure 1

■■■

Nasty Software

By David Thompson

Formatters By Any Other Name

During the early days of the Greek/Trojan war, a couple of Greek programmers modified one of their battle simulation programs and named it TROJANS.WIN. Then they uploaded a copy to a bulletin board inside Troy. It wasn't long before the Trojans discovered it and ran it on their Zeus.

Its first instruction: "Steal the Greeks' horse."

I remember the time when formatters were called FORMAT.COM or HDFMT.EXE. That was also the time when computers were the treasures of a select few, and neither the FBI nor the Justice Department cared what happened to this tight little community.

That time is gone, and nothing points it out better than some of the newest public domain software — the trojan programs.

PC–Write

A bulletin board operator in Los Angeles recently discovered a version of PC–Write uploaded to his board. The program said it was a newer release of version 2.71, but instead of editing text it scrambled his hard drive's FAT (file allocation table) and then initiated a low level format.

The bad version 2.71 is reportedly 98,274 bytes long while the real version 2.71 is 98,644 bytes and the earlier version 2.7 (also real) is 98,242.

PC–Write's developer, Quicksoft, is offering a $2500 reward to the first person who identifies the trojan's creator, and $5,000 for proof which convicts the scoundrel. (The copy of PC–Write available on disk from Micro C is straight from Quicksoft.)

Dealing With Trojans

Trojans are a real pain, erasing directories, scrambling FATs, and reformatting data areas. But when you stop to think about it, that's about all they can do which the on–off switch won't correct.

Sure, losing a floppy full of files can be a pain, but a floppy disk is so easy to backup that you shouldn't do anything on your only copy. The RBBS operators I've talked to are suggesting people backup hard drives before running any programs they've downloaded.

Of course, it's a pain, not that you shouldn't backup hard drives anyway, but it's still a pain and not something I'll do every time I try a new program.

However, most of our systems have flip–top cases. It's really easy to unplug the hard drive controller card from its slot and lay it, cables still attached, on top of the hard drive. (I lay a piece of conductive foam — the stuff the board came packed in — on top of the drive and then lay the board on top of the foam.)

I don't disconnect anything else. You can disconnect the power cable from your hard drive, but I haven't bothered.

Anyway, it takes about 10 seconds to unplug the board and then I reboot on floppy and try running the new software. If it does any nasties, then so be it.

Unfortunately, there are sophisticated trojans. Some work as advertised until they detect a hard drive. Others leave something in memory, just waiting for a while or for a certain kind of interrupt. Then they strike. It's not unusual to blame an innocent program after such a crime.

Some Suspect Programs:

ARC513.EXE, ARC514.COM: Writes over track 0 on your hard drive.

BACKTALK: Used to be a good PD utility. Destroys hard drive sectors. There are good and bad copies of this around.

CDIR.COM: Supposed to display a directory in color, but scrambles FATs instead.

DANCERS.BAS: Generates a color display of moving dancers. The bad version also wipes out the hard disk's FAT.

DISKSCAN.EXE: A program from PC–Magazine which searches the hard disk for bad sectors. The nasty version creates bad sectors for the good version to find.

DMASTER: A FAT scrambler.

DOSKNOWS.EXE: A harmless system status utility, if it's 5376 bytes long. Otherwise it destroys FATs.

DPROTECT: There are a good version and a FAT–eating version of this.

EGABTR: A total trojan. The documentation says it will improve the EGA display, but when run it prints "ARF! ARF! Got you!" Deletes anything it can find.

EMMCACHE: This EMS disk cache program scrambles files and writes them to disk, then destroys the boot sector. It can damage many files before you realize what's happening.

FILER.EXE: There's one report that it wiped out a hard drive, but all other reports indicate this is an excellent file manager. Test first.

FINANCE4.ARC: Rumors say this might be a trojan. Handle with care.

FUTURE.BAS: Displays a color picture, tells you you should be using your computer for better things than graphics, and then starts erasing files from drives, starting with A:.

MAP: Definitely a trojan; I don't know what it does.

NOTROJ.COM: Supposed to ward off other trojans, but it erases the hard disk FATs, all the while complaining that another program is trying to format the disk. Supposedly does nasties only to hard drives which are over 50% full.

TIRED: A FAT scrambler.

TSRMAP: Displays a map of all TSR programs and erases the boot sector on drive C:.

PACKDIR: Is supposed to pack files on a hard drive, but it also scrambles FATs.

QUIKRBBS.COM: Claims to speed the load of RBBS–PC's message file. Doesn't work.

QUIKREF: Contains ARC513.COM.

RCKVIDEO: Displays simple animation of a rock star, then erases files.

SECRET.BAS: Disk formatter.

SIDEWAYS.COM: The good version prints text sideways; the other version trashes the hard disk's boot sector. SIDEWAYS.COM, about 3K, is the trojan. SIDEWAYS.EXE, about 30K, is legitimate.

STAR.EXE, STRIPES.EXE: Puts graphics on the screen, all the while copying RBBS–PC.DEF into another filename which a snoop can download later to get passwords.

TOPDOS: A high–level hard disk formatter.

VDIR.COM: The disk killer which Pournelle described in Byte.

There are no doubt many more. Anyone who can RENAME a file can RENAME a new trojan.

Buggy Programs

Of course, there are lots of buggy programs out there and it's nice to know which ones can fail in destructive ways, but don't assume every bad ending is the result of a trojan. (Hardware can act strangely, too.) Trojan stories spread faster than trojan programs, so be a little leery of undocumented reports (including the ones here).

Also, be aware that stacked TSR (terminate and stay resident) programs can make a system overly sensitive to program strangeness. Be sure your system is clean before you write off a new utility.

The above list of trojans was taken from the file DIRTYDZ7.ARC, put together by Eric Newhouse. (He keeps the file updated so watch for DIRTYDZ8.ARC.)

If you find an honest to gosh trojan, send it in. If you know who wrote it and why, tell us. Also, be sure to contact Eric Newhouse on his bulletin board at 213–471–2518 (1200–2400), so he can add your findings to DIRTYDZ.

And, if you see a very large horse standing outside your city's wall....

■■■

TECHNICAL TIPS

This issue, Tech Tips comes from the Tech Help conference on Micro C's RBBS. Read on for discussions on how to clobber call waiting and the perils of screen-save utilities.

To Save Or Not To Save
To: ALL
From: JOE COBB
Subj: MONITOR PROBLEMS

I need some information about monitor problems.

First, why is there a soft buzz from my old Kaypro II monitor? Does it suggest the screen is about to burn out on me, or is there some simple adjustment that can be made to stop whatever is causing the buzz?

Second, on my DOS machine (Kaypro 16XT) there seems to be a color graphics card, in spite of the fact that the built-in screen is green phosphor on black. For the past year and a half I have used an external RGB monitor. A week ago, while I was out to lunch, the monitor went to lunch too. The little green screen works but the RGB is dark - even the "power on" light doesn't respond.

Does anyone out there have a theoretical diagnosis? I have been told that my continuous use of a screen-blank utility caused the problem, because the electrical whatnot had no place to go every time the screen went off and it has burned out my flyback transformer (whatever that is).

I am in mortal fear that I am about to be taken for hundreds of bucks by various "repair" attempts, and a straight replacement would be about the same price.

Is it plausible that the screen-blank utility caused the burn out?

To: JOE COBB
From: R. LEGGITT
Subj: MONITOR

I don't know anything about your monitor, but I do know: a) it is not possible for scrnsave to burn out a monitor, and b) if your power light isn't coming on then you are not getting any power. I would check power connections, fuses, fusible links, etc. Anyway, it should be relatively easy to fix, unless it's something REALLY bizarre.

If you think that the service man is pulling a fast one, let me know what his diagnosis is and I'll tell you if it makes sense or not. By the way, flyback occurs when the raster line reaches the rightmost edge of your screen and flies back at high speed to the left edge to begin the next line. The flyback transformer captures the energy produced and creates the high voltage necessary to move the raster to begin with.

The flyback transformer is also responsible for that high pitch whine you sometimes hear, especially from old TVs. The frequency of that whine is 15750 Hz, exactly the scanning rate of the raster.

To: JOE COBB
From: DAN POST
Subj: MONITOR PROBLEMS

Hey, I work on Kaypros and the 16 has the same video card as the older PCs before the new "half-length" video card with Hercules emulation came out. It even supports composite monitors! If you're getting a video signal on your green monitor, the video card is working. Since you say the pilot LED (indicating "power on" on the RGB monitor) is not coming on, I would suspect something in the RGB monitor's "primary" power area - the weakest point in a device of this type. Check to see if the fuse is blown.

And don't let anyone tell you the screen blanker tore up the monitor (unless it uses a hammer!). All the screen blanker does is sit resident in memory and decide whether or not to let the video signal out the back of your machine. It does nothing more dangerous than turning down the intensity of your monitor. The only difference is that it interrupts the video signal at the source rather than at the potentiometer on the front of the monitor.

Considering the cost of a new monitor, check the fuse first - not hard to do. Then take it to a competent computer repair place, or even a good TV repairman. Compared to a TV set, the RGB monitor is really a simple piece of equipment. No tuner, no audio, no demodulators - just two sweep circuits and a control for the intensity of the three electron guns (Red, Green, and Blue).

To: ALL
From: GARY OLIVER
Subj: SCREEN SAVE/ETC.

Just to add my "2 cents" worth:

It IS possible for screen-savers to damage monitors, though rather unlikely. Many monitors need to have a constant "supply" of sync signals (Horiz/Vert). If these are interrupted, the monitor could fail due to overheated components in the vert/horiz output stages. I do not know how most systems "blank" the display, but if it is simply removing the video signal, while leaving the syncs, then it should be ok.

Another point to consider is that a monitor in blanked state actually consumes more energy than one producing a display. This is caused by the fact that the little electrons want to hit the phosphor and in order to blank the screen, they are deflected from hitting it. As an example, you might examine the manual for an old (> 5 yrs) TV set (particularly B/W) that warns not to turn the brightness control down for prolonged periods.

If I've been redundant here, my apologies - I didn't read all of the discussion (rather long) and thought I would just make it a touch longer.

Phone Line Juggling

To: ALL
From: GEORGE KEYES
Subj: CALLWAITING

Does anyone have any ideas on how to deal with call waiting, without having Ma Bell shut it off? Are there any programs available, or is anyone ambitious enough to tackle the problem?

To: GEORGE KEYES
From: STEVE CROFT
Subj: CALLWAITING

You have to prefix the number you are dialing with "*70". I don't know if all comm packages will support this or not. Oh yeah, the "*70" is what I use for Pacific Telesis. I assume it would be the same elsewhere, but who knows?

To: GEORGE KEYES
From: MARGRET ROSENBERG
Subj: CALLWAITING

George,

There are 4 solutions I'm aware of:

1) In some parts of the country, Ma Bell has a method that lets you shut call waiting off selectively. This is by far the easiest solution, if it's available to you. Call the phone company to find out.

2) A SYSOP in Denver tells me that the phone company maintains a "deadend" phone number. They do not tell the public about it. (My friend found out about it from a lineman.) The phone co. uses it to busy up extra lines so they can test just one line. What happens when you dial it, is nothing. No ring. No busy signal. No beep–beep–beep your–phone–is–off–the–hook signal. Nothing. The call just goes to Ma Bell and stops. But until you hang up, your phone is busy.

To use this to disable call waiting, first call the deadend number. Then click the receiver cradle one time (just as you would to answer a second call). This will give you a second dial tone, and you can call your favorite board. Your first "line" will still be busy with the deadend number, so anyone calling you will get a busy signal. When you hang up, your phone will ring (as it would if you hung up on one line while someone else was on call waiting). Pick up the phone, hear nothing (it's the deadend line), and hang up again.

The problem is finding the deadend number in your area. If you have a friend who is a lineman, ask. Otherwise, it'll take some detective work. In Denver the number is 795–0000. Apparently, all the deadend numbers end in 0000. So you can try all the local prefixes, and have a reasonable chance of finding it. (Please do it in the daytime – you're going to get a lot of wrong numbers in the process.) Remember, if it rings, it's NOT the deadend number.

3) This one should work anywhere. First dial the board. Then click the cradle once to get your other line (and a second dial tone). Then call the board AGAIN. You will, of course, get a busy signal, since the board is online with you. DON'T hang up. Click the cradle again, and you'll be back online, with the second line busy listening to a busy signal.

Disadvantages – If you're using some service other than AT&T, you may be billed for a second call, since they tend to assume that the phone has been answered whenever a call goes over a minute or so. Also, it won't work if you're dialing a board that can handle more than one caller. I haven't tried this one – I'm going on hearsay only. But it may work.

4) If you have call forwarding, you can always forward incoming calls elsewhere while you call the boards.

Good luck, Margret

■■■

Changing The Picture

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

An Introduction To The TI TMS34010 Graphics Processor

Someone said, a long time ago, but in this galaxy: "A picture is worth a thousand words." We PC users have been a bit slower than most to catch on, seeming to prefer to type the thousand words than to picture data. Or maybe we've just been limited by our technology. If that's the case, there's hope for us.

Recently, new developments in computer graphics hardware has made it irresistibly possible to create high performance (affordable) "pictures" on a personal computer.

The TMS34010 plays a key role in this new development. I've been working with this chip, and I'd like to introduce you to it. But before I get into the details, let's go back a little, and at least mention a few of the key developments in the graphics technology past that have gotten us this far.

Graphical Odds And Ends

For a long time, computer graphics was limited to those huge, multi-million dollar corporations that could afford to use them for such expensive-sounding tasks as computer aided design.

Typically, graphic displays were controlled by a small computer which was linked to a mainframe, which handled the most computational intensive tasks.

I once saw a system at Boeing where a large DEC minicomputer was linked to a mainframe, allowing an operator to turn a knob on his graphics display, rotating an image in something close to real- time.

The graphics terminal market was dominated by Evans and Sutherland and later by Tektronix and Hewlett Packard. But no real standards existed for driving graphical devices.

In the late 70's Xerox quietly started a trend with the introduction of the nearly famous Star Workstation, a machine intended primarily for word-processing. It had, at its core, a CPU which was well-suited for graphics manipulations on its hi-res monochrome CRT. (We use a Star at PC Tech with special software to design our circuit boards.)

About this time personal computers came on the scene and defined the low end of the graphics environment. Most CP/M computers had no graphics capabilities, but it didn't matter at first. Just using a thousand words was pretty exciting. Now our standards for high resolution are 640 by 200 or 640 by 350 with four or sixty- four colors.

But in the workstation market hi-res means something else.

Companies like Sun and Apollo who dominate this arena create displays of over 1024 by 800 pixels with 64 colors even in their low end systems. In Apollo workstations, the graphics display is handled by a separate 32-bit processor, a Motorola 68000, with two or more megabytes of memory in addition to the frame buffer.

Graphics Chips

The NEC 7220 was the first popular graphics controller chip. Still in use, it handles display refresh and takes commands from the CPU. It performs complex drawing functions: such as lines, arcs, area fills: which are implemented with hardware built into the chip, thus providing a high speed but inflexible solution.

In 1982 Hitachi announced that their advanced CRT controller would implement even more graphics commands in hardware and add even more speed to the drawing process.

A year ago Intel began demonstrating the 82786 graphics controller, an impressive chip allowing such complicated tasks as windowing to be handled with relative ease. Both character and graphics data are easily displayed with a minimal amount of hardware and software.

TMS34010 Graphics System Processor

So, in a world of graphics controllers what makes the TMS34010 stand out?

Well, to put it simply, the TMS34010 is not a graphics controller.

It's a complete 32-bit CPU with an instruction set well-suited for graphics operations combined with a CRT controller. Gone is the rigid command set of the graphics controller. Now we can define our own command set which can be very high level, freeing the host CPU for other tasks.

The Part

The 34010 is housed in a 68 pin plastic leaded chip carrier, approximately one inch square. It's built in CMOS, allowing very low power consumption of only 0.5 watts. It can control displays of up to 64K by 64K pixels and can draw at a rate of up to 48 million pixels per second for single bit pixels.

The 32 bit CPU incorporates 31 general purpose registers, a 256 byte LRU cache, a barrel shifter, and a complete instruction set.

The Host Processor Interface

The 34010 accommodates both 8 and 16 bit host bus interfaces on its sixteen host data lines. Two data strobe lines, /HLDS and /HUDS, are used for host processor communications (see figure 1).

So for an eight bit bus interface, the upper 8 data lines (HD8 - HD15) can simply be connected to the lower eight data lines (HD0-HD7). The /HLDS signal can be address line A0, and the /HUDS signal can be /A0.

In addition, the 34010 has a host chip select input, read and write inputs, two address inputs as well as a ready and interrupt outputs. These pins are summarized in table 1.

Four 16 bit registers are accessed through the host interface. Through three of these registers any of the 34010's local memory can be accessed, which also allows initialization of the CRT control timing and other functions. The two Host Function Select pins access the registers in the pattern in figure 2.

To access a memory location in the 34010's local memory both address registers must first be loaded with the complete 32 bit address. Then the data may be read or written through the data register.

The Control register allows the 34010 CPU to be halted or interrupted and also allows the memory access to be programmed so that the address is incremented after every write or read. The 34010 cache may also be flushed via the control register.

Local Memory Interface

The 34010 memory interface is a unique triple-multiplexed address/data bus. Keep in mind that the 34010 is designed to work best with dynamic RAMs and the elegance of the design soon becomes apparent.

At the start of the bus cycle the 34010 first outputs the row address to the DRAMs. A short time later the /RAS signal is output, then the 34010 outputs the column address, followed by the /CAS signal.

In order to keep the column address stable to the DRAMs throughout the remainder of the bus cycle the column address is latched into transparent latches with a signal called /LAL. The 34010 now may read data or write data on the local bus.

The display memory in a 34010 system MUST be dual ported Video RAMs (VRAMs). These memory devices have built-in shift registers which load an entire 256 bit column from the RAM for shifting out to the display. Through its memory interface the 34010 performs a RAM to shift register transfer as needed to refresh the display. These transfers can occur only during horizontal blanking or vertical blanking.

The main advantage of this type of video memory is that the CPU has nearly unlimited access to the display buffer, freeing it to draw at will, reducing the memory access time requirements.

Most often the 34010 will have additional memory on its local bus. Additional DRAM can be supported as easily as the video RAMs through the multiplexed address/data bus. One difference to note is that the write enable signal to the DRAMs is too late for 120nsec access time DRAMs. This caused me more than a little grief!

The 34010 also performs the required DRAM refresh cycles. Supporting EPROMs on the 34010 bus is only slightly more difficult. The address must simply be de-multiplexed by latching the row and column address at the appropriate times during the cycle to create the address for the EPROM.

The CPU

The 34010 is a full-fledged 32 bit microprocessor. When executing instructions from its on-chip cache it runs at 6 million instructions per second maximum. The 256 byte cache always keeps the most recent instructions present, unless it's disabled by the host.

The 34010 is a bit-addressable CPU with a 32 bit internal memory address. This 32 bit address points to a single bit in the memory, which can be viewed as a continuous stream of bits. Bits may be organized into structures called fields. Each field is addressed by its bit address and length. A field length may be from 1 to 32 bits and is programmable.

The 34010 CPU has several different types of registers. The program counter is a 32 bit register which has its least significant four bits fixed at 0. Thus, all instructions will be aligned to a 16 bit word.

The CPU also has a dedicated 32 bit status register which contains the normal sign, carry, zero and overflow bits as well as interrupt enable and field size bits. There's also a 32 bit stack pointer.

Thirty general purpose registers remain. They're divided into two register groups, A and B, of fifteen each. The stack pointer shows up as the sixteenth register in each of the register files.

Register file A is general purpose, with no dedicated graphics functions associated with any register. These registers are referred to as A0 through A14.

The B register file, B0 through B14,

(continued next page)

Figure 1 - Data Strobe Lines

/HUDS	/HLDS	Operation
0	0	16bit
0	1	Lower8bit
1	0	Upper8bit
1	1	None

Figure 2 - How Host Function Select Pins 0 and 1 Select Registers

HFS1	HFS0	Register	
0	0	HSTADRL	16LSBsofaddress
0	1	HSTADRH	16MSBsofaddress
1	0	HSTDATA	Dataregister
1	1	HSTCTL	Controlregister

Table 1 - Host Bus Interface Pins

/HCS	HostChipSelect
HFS0,HFS1	HostFunctionSelect
/HREAD	HostRead
/HWRITE	HostWrite
/HLDS	HostLowerDataStrobe
/HUDS	HostUpperDataStrobe
HD0-HD15	HostDataBus
HRDY	HostInterfaceReadyOutput
/HINT	HostInterruptOutput

(continued from page 87)

is also a general purpose register file. However, each of these registers has hardware - dedicated functions associated with it during pixel operations. These functions are briefly shown in table 2.

Instruction Set

In some ways the 34010 is similar to a RISC machine. The 34010 doesn't have the 3 zillion addressing modes and data types of the more popular 32-bit processors, but does include everything needed to make it shine in the graphics environment. And it does shine (very quickly).

Move Instructions

Two basic move instructions become over two dozen types of moves when all addressing modes are considered. Data may be moved from register to register, from a constant to a register, bytes may be moved, fields may be moved, multiple registers can be moved and more.

General Instructions

Some 49 different instructions cover all the arithmetic and logical operations including signed and unsigned multiplication and division. Logical shifts, arithmetic shifts and modulus instructions also fall in this category along with dozens of other useful instructions.

Program Control Instructions

Any instruction which alters the flow of the program fall in this group. Jumps, calls, traps, returns and others add up to 20 instructions here.

Graphics Instructions

These are the really exciting ones. There are 12 basic instructions available in this group. A few of the highlights are:

FILLXY
LINEZ
PIXBLTn,n

The FILL instruction can be used for rapid fills of blocks of the display area. LINE is used to draw a line (no kidding) and PIXBLT is used for every conceivable type of pixel block transfer. Prior to the execution of any of these instructions the B register file must be initialized. These registers contain operands which are implied for these instructions.

Since I'm not a software guru I'll leave this area to be covered in a future article, after we know a bit more about programming in 34010ese.

Texas Instruments provides good and fairly priced tools for software development. These tools will run on the X16 (or compatible) as well as a VAX.

Some Real Designs

At PC Tech we've been looking at the TMS34010 chip for a while. Using our Star workstation we've developed two designs which utilize the part.

The first design is an ultra hi-res monochrome board which we're testing as I write this. It's designed to go into an IBM PC/XT/AT, X16, X32 or compatible where it emulates the monochrome adaptor or the color graphics adaptor while providing screen resolutions of up to 2048 by 1024 pixels, with four pixel intensities. Currently, this board has one megabyte of DRAM for 34010 program storage and font storage. We developed it for a German company and will be in production by summer.

The second design exists only on the Star workstation where our routing guru, Buddy, is tweaking it to completion now. This design is actually a two board set, with the video RAM and color palette on a second daughter board. This will allow the board's capabilities to be expanded as needed for higher performance applications.

The present configuration of this second design allows for a display of up to 1024 by 1024 pixels with 256 colors from a possible 256K palette. The total on-board DRAM is limited to four megabytes (but wows are appropriate!) and the 34010 also has up to 512K of local EPROM.

Maybe we'll have something to show at SOG.

Closing Fast

The distinctions between micros and dedicated workstations is closing faster and faster. Now we can put together a desktop computer with 16 Megabytes of RAM, 340 Megabytes of hard disk, a floppy drive, a tape drive, and a 640 by 350 hi-res display. Tomorrow we'll change the picture. And you and I know what a picture is worth.

References:

TMS34010 User's Guide - 1986 Texas Instruments Corp.

TMS34010 Data Sheet - 1986 Texas Instruments Corp.

■■■

Table 2 - Register Functions

Register		Function Explained
B0	SADDR	Source address for PIXBLTs
B1	SPTCH	Sourcepitch
B2	DADDR	Destinationaddress
B3	DPTCH	Destinationpitch
B4	OFFSET	
B5	WSTART	Windowstart
B6	WEND	Windowend
B7	DYDX	DeltaY/DeltaX
B8	COLOR0	Color0,PIXBLTs
B9	COLOR1	Color1,PIXBLTs,FILLs&DRAV
B10	TEMP	TemporaryRegisters used during
B11	TEMP	PIXBLTs and FILLs
B12	TEMP	
B13	TEMP	
B14	TEMP	
B15	SP	Stackpointer

(continued from page3)

I asked if they could handle data via phone.

"Just call our number in 5 minutes and we'll answer with a modem."

Bits per character? Parity? Baud rate?

"Whatever you want."

How about 8 bits, no parity, 1 stop, and 2400 baud?

"Sure, we do that all the time."

It turned out they were running a 1200 baud modem which couldn't answer the phone. They didn't have the slightest idea how the modem software worked. Oh well.

We posted files on the RBBS. They couldn't call. We shipped disks on the bus. They lost the disks, or the files didn't work, or the disks didn't fit into a MAC (yes, PC disks really are larger, but a good pair of scissors...).

We even tried Federal Express. Once. Sure enough, they call you back with your package's status within 30 minutes.

"I'm sorry sir, I don't find that waybill number, do you know where it's being sent?"

It's been five days now and it still hasn't been delivered.

We sent raw text files. They didn't have Ventura.

"I guess our Ventura hasn't come yet...."

But it really didn't matter; they didn't have a hard drive either.

If you're wondering why we didn't just output through a laser printer, we did. I mean, we tried.

But that's a whole 'nother story, and you're probably sympathetic enough already.

(By the way, if this issue really comes out of a RIP, I take it all back. At least, most of it.)

Fast

I thought you might be up for a quick look at fast.

Some systems are measurably fast. Others are immeasurably fast. Norton's fast seems to be the standard fast, probably because it's such a fast fast.

Anyway, here's how I rate the systems around the office. You'll have to understand that this is purely subjective. It's an integration of how fast a system boots, how fast it transfers files from disk to memory, how fast it scrolls the screen, and whether or not a short compile sends me to the kitchen for tea.

Kaypro 2000

Our slowest system is the Kaypro 2000. It has a 4.77 MHz 8088, a 300 baud screen (that's optimistic), and a 3 1/2" drive that's so pokey it makes a full-height Tandon look like a RAM disk. If Larry had really wanted to create the world's slowest clone he would have started with the 2000 (see his Half-Fast article in this issue).

As far as I'm concerned, the 2000 rates: 1 for processor speed, 0.5 for disk speed, 0.5 for screen writes, 9 for compatibility, and 10 for dependability. (All ratings are on a scale of 0 to 10.)

Every office needs at least one Kaypro 2000. It's portable, has a very nice keyboard, and it makes the other systems look a lot faster.

Generic Clones

We have a couple of standard clones which we've sped up to 6 or 7 MHz (with added V20s). They're OK, especially with little 20 meg Seagates. I give them 1.5 for processor speed, 1 for disk I/O (3 with the Seagate), 1.2 for screen writes, 9.0 to 9.5 for compatibility, and 8 to 9 for dependability.

Holliston 80186

We have three of the old Holliston boards (8 MHz 80186). (They've been available from Challenger Computer since Holliston went under.) They really take their time putting on cold boots (for you coffee drinkers, it's about a two-cup delay). When Trevor Marshall (Definicon) used one of our Holliston's as a base for one of his coprocessors, he was sure he'd broken it. (I wasn't concerned, his second cup was still warm.)

Once the Hollistons are running, they're quick. Oh, the disk I/O still isn't anything to write home with, but the 80186 is a decent processor.

Also, the Holliston is a good test for compatibility. It irritates the heck out of most copy protection schemes, and it seems to be particularly sensitive to problems with standard applications packages.

If any machine will have a problem running something, it'll be the Holliston. The problems are usually small and avoidable (the system might lock up when the keyboard buffer fills). Fortunately, I haven't found anything that really bombs on Hollistons (not counting copy protection).

The Hollistons don't support floating point coprocessors, but they do have a couple of AT style sockets in addition to the XT style.

I rate the Holliston: 4 for processor speed, 2 for floppy I/O, 4 for Seagate 225 hard disk I/O, 3 for screen writes, 7.5 for compatibility, and 10 for dependability.

PC-Tech's 80186

At first, PC-Tech's X-16B appears to be just another 80186, but hiding somewhere in its sparsely populated board is a very finely honed machine. Disk and screen I/O rival any of the 286s. And only one program has refused to run on it (copy protection again).

The X-16 is my main machine, and it reached that status by calmly accepting resident program piled upon resident program without gasping. It also let me use 360K (of its 1 meg RAM) for RAM disk without impacting its 640K TPA. (I recently added a Lapine 20 meg (65 ms) hard drive — it's tiny and very quiet.) Boot up is faster than any other system.

I rate the X-16B: 4 for processor speed, 5 for floppy I/O, 6 for hard disk I/O (with an XT controller, SCSI would no doubt be faster), 7 for screen writes, 10 for compatibility, and 10 for dependability.

(I know you're going to ask me when Dean and Earl will be releasing their super-charged 386 board.... Look, you're not supposed to know they're working on it. So if I were to say anything at all about their zippy new 386 board, people would think I couldn't keep a secret.)

(continued next page)

Kaypro 286i

The 286i's massive case and matching keyboard put it in the continental class. (The keyboard was incredible.) Its low–revving V8 (6 MHz 80286) cranked out a lot of power, but the impression was one of ponderous mass.

I replaced three AT style WD drive controller cards in the 286i. (I talked to Western Digital recently and they say their new cards are much better — the new version has a large square custom IC in the middle of the board.)

I rate the 286i: 5 for processor speed, 4 for floppy I/O, 5 for hard disk I/O (full–height Rodime), 7 for screen writes, 10 for compatibility, and 7 for dependability.

Generic AT

Two weeks ago we replaced our Kaypro 286i with a generic AT — after a midnight requisitioner wandered into the office and left with the Kaypro. (If you happen to see a 6 MHz Continental...)

At 10 MHz (1 wait state) the new AT clone is very fast. I'm bummed by the loss of the 286i's keyboard and color monitor, but I'm really enjoying 10 MHz.

This is now our main system for Ventura Publisher, and it's really responsive. Disk and screen I/O are the thinnest hair shy of instantaneous. (The Seagate 4038 high–speed 30 meg drive doesn't hurt.)

I rate the Generic AT: 8 for processor speed, 8 for floppy I/O, 9 for hard disk I/O, and 8 for screen writes. (When PC–Tech's secret 80386 board hits the market I'll adjust my definition of instantaneous.)

I don't have a good feel for the unit's compatibility or dependability, but it's run everything we've fed it, and it came right up even though it started out life as a pile of boards. (If you don't count 8 hours of frustration trying to figure out why the drives didn't work — there was one jumper too many on the AT–style drive controller card.) It occasionally has trouble reading or writing floppy disks, but slowing it down to 6 MHz solves the problem.

I purchased the boards through MicroSphere and they asked me whether I'd let them put it together so they could see how easily it came up and how well it performed. They got orders for three more from customers who watched mine come up.

Finally

It looks like there's going to be more and more fast around. Don (from MicroSphere) and I are teaching a "Build Your Own Computer" class at the local college, and most of the students ordered 4.77/10 MHz 8088 processor boards. (I was so surprised when I read the specs I insisted Don verify the clock speed with the distributor.)

Even with a V20 in place of the 8088, it's not as fast as the Holliston boards, but compared with 8088 systems it moves. Screen I/O, however, doesn't seem much faster than our 7 MHz clones, but I guess it wasn't supposed to be much faster.

PD32

We've had a lot of messages on the RBBS from discontented PD32 folks. Where is it? What happened to Efron? When's the board going to be available? I always grit my teeth when I log onto the PD32 conference.

Well, no longer.

Dan's back. I understand he's been corresponding with those who've written to him, and just last week got back on the Micro C RBBS. Anyway, here's the latest.

Dan Efron, Dave Rand, and George Scolaro have lined up a board manufacturer for their version of the PD32.

Dave Chen
Cybertool Systems
192 Whitney place
Fremont, CA 94539
415–659–0808

It's a 10 MHz, 2 megabyte, 4–layer board, and preliminary prices are:

- Assembled and tested with UNIX 5.2. — $949.00
- Full kit. All sockets are wave soldered in place, just plug in the chips (2 meg RAM included) and fire it up. Comes with UNIX 5.2. — $795.00
- Full kit without UNIX. — No price yet.
- A basic kit which contains only the board and special parts like PALs, ROMs and the 10 MHz 32016 chip set. — No price yet.

- A bare board (available only from Dan Efron). — No price yet.

Dan says the user group will be supplying additional software such as Micro–Emacs, a fancy text editor with windows, separate buffers, and shells.

Dan's been having trouble getting messages from the boards, so the best way to get in touch with him is to write. Here's his address:

Dan Efron
8910 Westmoreland Lane
Minneapolis, MN 55426

Mini Mouse Review

A while back we purchased one of Mouse Systems' PC mouses (meece?) for Sandy. She's been using it to referee the CAD wars. The Mouse Systems unit is optical and comes with PC Paint (it keeps the kids occupied).

When we purchased Ventura we needed another mouse.

So I ordered a Logitech mechanical mouse (robot with a tail). That way I could compare the two mice, plus, I could use Ventura without impacting Sandy's CAD work (kill two rodents with one rock).

It looked good, in theory.

Within a week of the Logitech's arrival I had compared the two mice and found I really prefer using the Logitech. In that same week I also found that everyone else on the staff, including Sandy, prefers the Logitech (shades of Reflex). So the mechanical mouse was always tied up when I was ready to use it. How underhanded.

I called Logitech to complain.

Now we have three Logitech mice (I have one of my very own) and are keeping two of them on adjacent systems so we'll soon have baby Logitech mice. I'll keep you posted as the tale grows.

There are three reasons for the Logitech's popularity.

- It moves the cursor farther per inch of mouse movement, so it feels more responsive.
- It runs over anything; you don't need a clean level spot for a special optical pad.
- At $99, the Logitech is less expensive than most.
- The serial version is easier to move from system to system than the Mouse Systems version, because the Logitech doesn't require a separate power supply.
- Logitech makes a great Modula package. (I know, I said three reasons, but I'm the editor here...)

Credit Where It's Due

We'd like to make special mention of our new advertiser, Jolly Roger Software, and its proprietor, Charles Johnsen III.

Higher Prices

"New Prices" sounds better, but it's really higher prices. We're trying to make the magazine part of Micro C self–supporting.

We could cut costs by eliminating the extras like tech support, SOG, and the RBBS, but Micro C would no longer be Micro C.

Anyway, our new prices are: $3.95 for a single copy beginning with this issue (up from $3.00) and $18.00 per year for U.S. subscribers (up from $16.00). Other prices remain the same.

User's Disks Free

I haven't mentioned it for a while, but Micro C disks are made up mostly of programs sent in by subscribers. These are programs you've written and programs (shareware and public domain software) that you've found particularly useful.

You get your choice of a Micro C User disk for each disk of public domain software you send in. If the programs are worthwhile to you (and not already on a Micro C disk) then get them in; we'd love to see them. Even if you have just one great program, send it in and tell us which disk you'd like in return. Free.

David Thompson
Editor & Publisher

■■■

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.

Affordable Engineering Software for CP/M-80, PC-DOS, MS-DOS, and TRS-DOS. Free 52-page catalog. Low cost pen plotter driver, transfer function analysis, scientific calculator, communications design spreadsheet, graph printing, circuit analysis, signal processing, mathematics, root locus, active filter design, thermal analysis, and report proofreader programs. BV Engineering, 2200 Business Way, Suite #207, Riverside, CA 92501. (714)781-0252. VISA and MasterCard accepted.

$19 Instrument Flight Simulator CP/M or MS-DOS — four aircraft types, air traffic control, realistic navigation, flight lessons, 25 page manual. Pilots or beginners. Fun! For CP/M, 8" or Kaypro II 5". BaileyTech, 304 WS College, Yellow Springs OH 45387.

Master Key opens the doors into your MS-DOS disks. It's a disk and file editor that can access any disk known to MS-DOS. Search for text! Alter file attributes! Erase, rename, and unerase files! Much more! For IBM PC/XT/AT and true compatibles. Send $20 to R. P. Gage, 1125 6th St. N. #43, Columbus, MS 39701.

Book Of Changes: computer oracle. Complete with 90K of text correspondences, newly translated. 72 page typeset book. Kaypro 4-84 video, vanilla versions, GINST included. Kaypro disk format. Professional package; published by author. Not Public Domain, but at $14 US postpaid, who cares? ZYQOTE Systems, Box 1165, Bonavista, NfLd., A0C 1B0 Canada.

Slicer Boards For Sale: Recently upgraded my Slicer to 1 Meg. Now have two Slicer 256K Expansion Boards for sale. One with all sockets, resistors and capacitors for full board installed plus 32 4164 chips, $125.00. Other with all sockets, resistors and capacitors for memory only board installed plus 32 4164 chips, $100.00. LLoyd Edwards, Rt. 1 Box 119, Lampe, MO 65681. 417-779-4076.

_____R_A_M___D_I_S_K_____ S-100, 2 MEG, PORT I/O, NEW, WARANTEED, $725. S. Lugert, 439 Peck Slip, N.Y.C, N.Y. 10272 or call 718-622-0654.

COMPUTER-AIDED MATHEMATICS_CompMath is an educational software tool for IBM-PC and compatibles. It is menu-drivew with selections of general mathematics, complex mathematics, matrix mathematics, engineering mathematics, and statistical math. CompMath will do polar/rectangular plots, complex graphs, bargraphs and scatter graphs from user data that is input from keyboard or disk file. CompMath includes tutorial style manual. $59.95 — Visa/MC — 614/491-0832. ESOFT SOFTWARE, 444 COLTON ROAD, COLUMBUS, OHIO 43207.

Magnetic Software — See those Magnetic Fields they have been telling you about! MAGPLOT scientific software computes and plots the magnetic field generated by current carrying regions in the presence of magnetic material. For CP/M - Kaypro II with MX-80 Printer. Send $15.00 for Manual and Demo-Disk to Saltek Services, P.O. Box 7847, Van Nuys, CA 91409 or call 818-708-9815 for additional information.

MAG TAPE DATA TRANSFER TO FLOPPIES: 800/1600 BPI to IBM-PC. First disk $40.00 + $18.00/disk or $60.00/hour. Other formats available, inquire. Micrologics Systems, 207 Kent Avenue #1, Kentfield, CA 94904. 415-461-8077.

Micromint MPX-16 Main Board: Keyboard/interface, complete documentation, firmware and software. $200 or best offer. 509-483-0231. Ask for Robert.

ATs, clones, PCs: RAM printer buffer — Can use extended (AT) or local memory and will cooperate with extended memory VDISKs. Many features. MS/PC-DOS 2.x/3.x. Captures all BIOS printer calls so even Shift-PrtSc goes through buffer. $25+$5 s&h/U.S. MiPS, 3920 Princeton Way, Livermore, Ca 94550, (415)455-0467.

TURBO Cheques! and TURBO Budget!: TURBO Cheques! is the most powerful personal Checkbook management program available for the IBM-PC. TURBO Budget! provides Budget management tracking capabilities. Both programs are professionally developed, easy to use, support color, hard disks and are not copy protected. User's Manual on disk. FREE DEMO DISK AVAILABLE. For the IBM-PC/XT/AT and true compatibles. Requires DOS 2.0 or higher, 2 DS/DD floppy disk drives and 128K RAM. $49.95 for both programs. CompuTech, Box 7000-309, Redondo Beach, CA 90277. 213-377-7198.

BBI System Components for Sale, from working system. 4 CDC 8" drives, new — $35 each. BBI board, make offer. Software, keyboard. Write for details. Greg McBrien, 149 Coles Road, Cromwell, CT 06416.

Wanted — SWP Co-Processor Board for '83 Kaypro 2. Landon Hall. 913-362-8791.

ELECTRONIC CIRCUIT DESIGN_CompDes is a computer-aided design software tool for IBM-PC and compatibles. It is menu-driven with selections from basic electricity to circuit design. CompDes will calculate values of resistance, reactance, db ratios, resonance, and more. It has selections for design of amplifiers, active and passive filters, attenuators, power supplies, etc. Includes educational tutorial manual. $49.95 — Visa/MC — 614/491-0832. ESOFT SOFTWARE, 444 COLTON ROAD, COLUMBUS, OHIO 43207.

Advertisers Index

Issue 35

THE LAST PAGE

By Gary Entsminger
1912 Haussler Dr.
Davis, CA 95616

On Your Own (With Lots Of Help)

Dan Bricklin was co–author of VisiCalc, the program which sold millions of Apples. You may be wondering what he's done for computers lately. Gary's description of DEMO might be a good clue.

The terse, but effective user manual for Dan Bricklin's DEMO ends on a high note, with several paragraphs of tribute to the tools, companies, and good folks who helped make the program possible.

A lot of talk, but for a good reason. I think Dan wants to remind us how a one–man operation can do it — with support and a little help from friends. DEMO is an excellent example of a programmer not reinventing the wheel, but inventing something new (and interesting).

Prototyping

DEMO is a prototyping tool which lets you create a demo of a program. You can demonstrate — a program which doesn't exist, or barely exists (a prototype), a program which does exist (a simulation), the use of a program (an interactive manual), or the use of something else (a tutorial).

The demonstration is a slide show on the PC, and DEMO helps you create and run the show. Consider these scenarios.

You're a consultant and a client wants you to create an expert system which will diagnose a microcomputer's problem and suggest a repair. Unfortunately, you're really not sure what your client has in mind. So, you pull out DEMO and create a slide show of what you think the program should look like.

Let's say, the program signs on and opens with a menu of choices. Slide one might look like Figure 1.

Slide two might demonstrate a selection, with a new window opening to show the available knowledge bases. (See Figure 2.)

Slide three might show the user's selection, and a fourth might begin a consultation which continues for 25 slides. Slide 26 might show the online help system. Slide 27 and so on, until eventually, slide by slide, the "image" of a program is created.

The point is, you can simulate the running of a program you haven't written, and probably clarify most of your or a client's expectations *before you start programming.*

Let's say you've written 75–90% of aprogram, and you want to test it (again from a user's perspective). You load CAPTURE (a RAM resident utility included with DEMO), which lets you dump a running program screen by screen into slides. You can then retrieve them from within DEMO, and edit the images. Great feature!

When your program really sings, you can capture a session, and add comments to slides, creating a demonstration of your program. A third included utility, RDEMO, lets you create unmodifiable demos, which you can run yourself or show to clients and prospective customers. The demo can even wait for specific user input.

(RDEMO runs the demo of Dan Bricklin's DEMO, an excellent interactive manual. In fact, the only way to learn to use DEMO is from the demo.)

Or you can create a program which will never have code; a tutorial on arithmetic, for example. Slide by slide, you can show how to do something, and explain it simultaneously. Slide one might present a problem along with instructions for the user. See Figure 3.

Our program (our demo) can wait for an answer, and if the answer is correct, move to another problem (another slide). When the answer is incorrect, it can display a different slide. The possibilities in this arena are virtually endless.

For more info —
The Software Garden
P.O. Box 238
West Newton, MA 02165
617–332–2240

■■■

Figure 1 - Slide 1

```
LoadRules
ConsultRules
ListRules
UpdateRules
SaveUpdates
Help
```

Figure 2 - Slide 2

```
LoadRules     Pleaseselectafile:
              PC.DBA
              KAPRO.DBA
              STAT.DBA
```

Figure 3 - Tutoring

```
Problem1
    IFX=50+Yand
    X=20
FINDY

PRESS
F1ifY=70
F2ifY=30
F3ifY=-30
F4ifY=NONEOFABOVE
```

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.

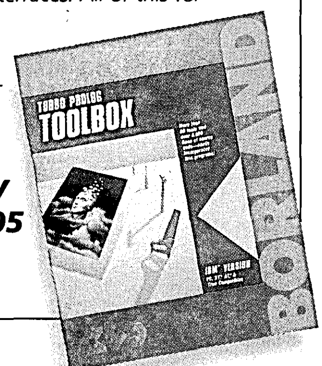
New Turbo Prolog Toolbox features include:

- *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*

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

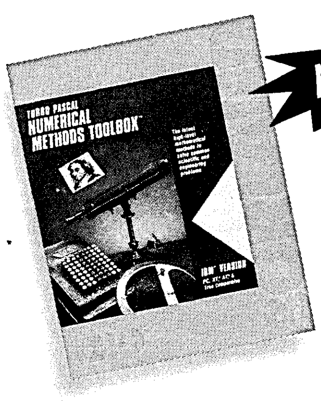
Only $99.95



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.

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

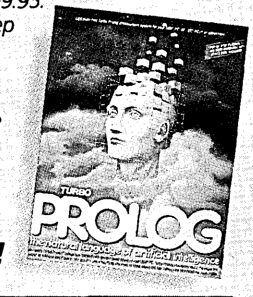
System requirements

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