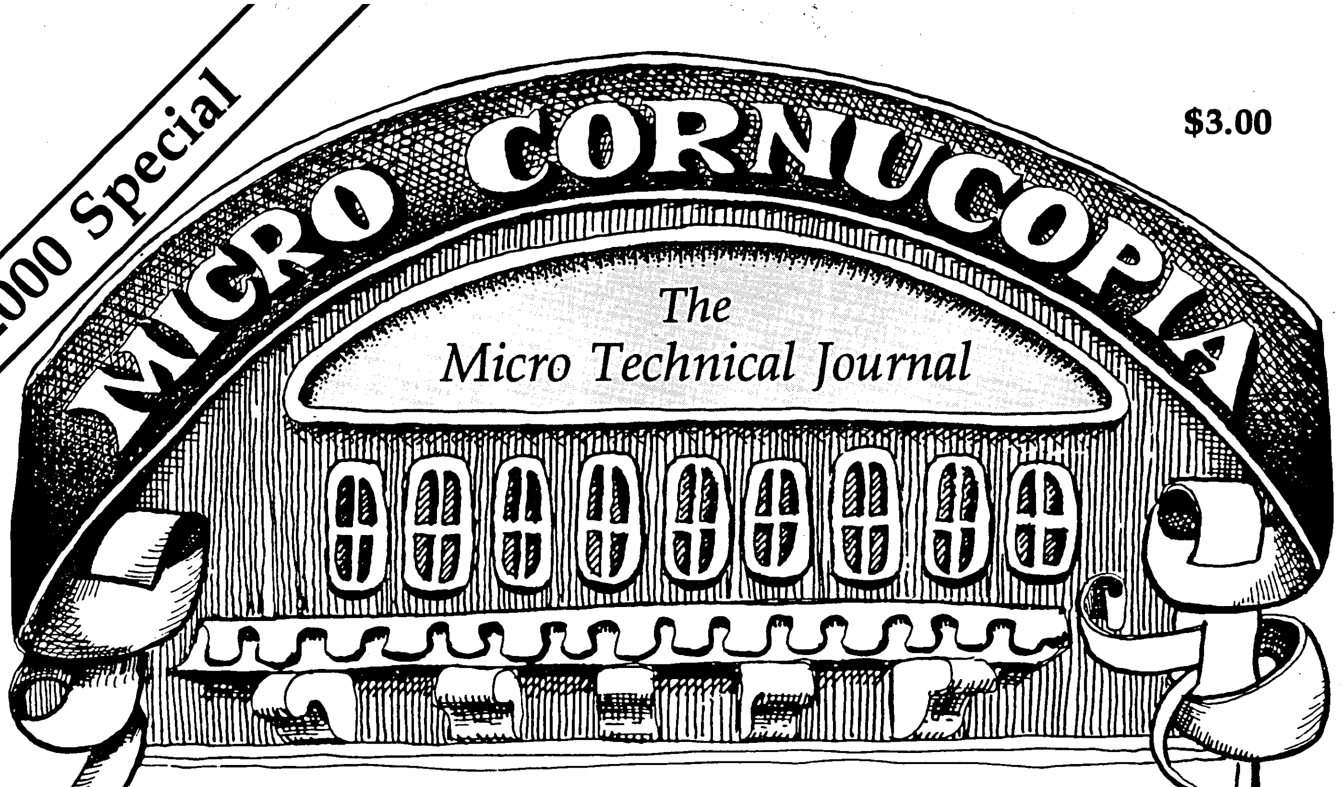
32000 Special

$3.00



MICRO CORNUCOPIA

The Micro Technical Journal

October-November 1986 No. 32

World's Cheapest Unix System

Special Features

RISK FREE
30-DAY MONEY-BACK GUARANTEE

NOW FOR YOUR IBM PC, XT, AT, AT&T, PC Jr. AND COMPATIBLES *

"THE PC HARDWARE BARGAIN OF THE YEAR"

Chuck Humble — Computer Editor

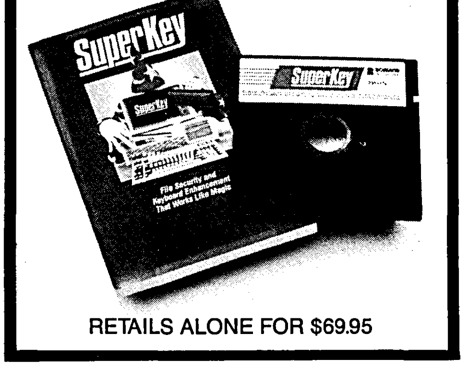
Both Hardware & Software

~~$129.95~~

Introductory Offer $99.95

OUR NEW PC 8700 AT-STYLE KEYBOARD AND BORLAND'S SUPERKEY® SOFTWARE

INCLUDES BORLAND'S *SUPERKEY* AWARDED "BEST OF 1985" BY PC MAGAZINE


SuperKey
File Security and Keyboard Enhancement That Works Like Magic
SuperKey

RETAILS ALONE FOR $69.95

THE PC 8700 *AT-STYLE KEYBOARD* INCORPORATES EVERY IMPROVEMENT IBM DESIGNED INTO THEIR ADVANCED "AT" KEYBOARD. QUALITY SO GOOD, IT'S BACKED BY AN UNPRECEDENTED TWO-YEAR WARRANTY!

Advanced design features include:

- Separate Numeric Keypad simplifies data entry
- Status Lights for Cap, Num and Scroll Lock helps eliminate typing errors
- Extra-wide Control and Return Keys for ease of use
- Return Key placed in "standard typewriter" position for touch typists
- Selectric layout for Office Standardization
- Positive Tactile Feedback
- Keyswitches rated for 50 million keystrokes
- Made in USA
- 2-Year Warranty

BORLAND'S AWARD-WINNING *SUPERKEY* INCLUDED! GET THIS BEST-SELLING KEYBOARD ENHANCER SOFTWARE — VALUED AT $69.95 — AS PART OF THIS AMAZING OFFER AND CAPTURE THE POWER OF MACROS!

SuperKey's macro capability is remarkable. It literally turns a thousand keystrokes into one.

- Exclusive screen data Cut & Paste
- Screen Privacy and Protection
- Time & Data Stamping
- ProKey Compatibility
- Data Encryption
- Keyboard Lock
- Keyboard layout customization
- Integration with SideKick
- Pull-Down Menus
- On-Line Help and much, much more!

ORDER NOW — CALL TOLL FREE 800-826-5398; IN CA 800-592-9602

PLEASE RUSH ME __________ KEYBOARD/SUPERKEY BUNDLES AT $99.95 EACH. PLUS $10 SHIPPING AND HANDLING IN THE U.S.A.*

□ ENCLOSED IS MY CHECK FOR $109.99 FOR EACH BUNDLE ($116.45 FOR CA RES. — SALES TAX INCL.)

□ VISA □ MC □ NUMBER ____________________

EXP. DATE ________ SIGNATURE ____________________

NAME ____________________

ADDRESS ____________________

CITY ____________ STATE ____________ ZIP ____________

TELEPHONE ____________ COMPUTER TYPE & MODEL ____________

* PC Jr., AT&T and Corona computers require adaptors available at a nominal fee. Call for pricing.

CRITICS' CHOICE

"Who Can Pass Up a Deal Dept?"
John C. Dvorak, Infoworld, 3-10-86

"The best part of the keyboard is the feel. It is ideal!"
PC Productivity Digest, 5-86

"The keyboard's construction is solid . . . also quick and responsive to the touch. It is ideal for anyone who wants to or has to replace their PC keyboard . . . At $129.95 DataDesk's keyboard is a bargain. The PC 8700 keyboard with SuperKey for that price is a steal."
Computer Shop Talk, 3-3-86

"DataDesk International has designed a sturdy and handsome keyboard that has a tactile response . . . and bundled with SuperKey is the PC hardware bargain of the year."
Charles Humble—Computer Editor, The Oregonian

"DataDesk's is genuinely a nice keyboard . . . It feels nice . . . I can type much faster and much more accurately on it than most others . . . and it's inexpensive. It costs only $129, compared to around $300 for the Keytronics 5151."
Harry Newton, Publisher, Teleconnect, 5-86

TELECONNECT
TEST DRIVE SCORECARD

The DataDesk PC8700 Keyboard	
TECHNICAL PERFORMANCE	10
EASE OF USE	10
DOCUMENTATION	10
OVERALL RATING	10

NECT
RIVE
CARD

The KeyTronics 5151 Keyboard	
TECHNICAL PERFORMANCE	9
EASE OF USE	9
DOCUMENTATION	9
OVERALL RATING	9

TELECONNECT'S CHOICE

1-800-826-5398
1-800-592-9602
inside Calif.

datadesk
INTERNATIONAL

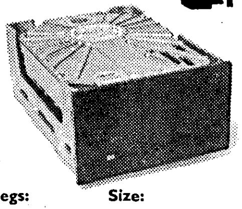
MasterCard

7650 Haskell Ave.
Van Nuys, CA 91406
818-780-1673

VISA

PLUG INTO SAVINGS!

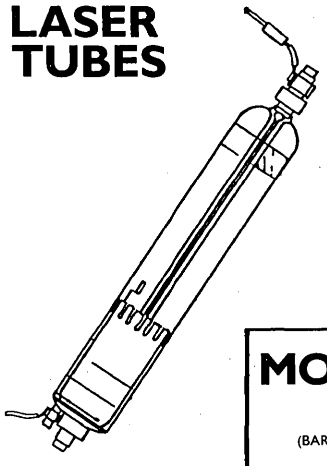
HARD DRIVES



LIMITED SUPPLY ON HAND

Name:	Model:	Megs:	Size:	Price:
Tandon (New)	262	26	5¼ or 3½ ½ Height	$ 399.95
Rodime (New)	202E	26	5.25 Full	399.95
CMI (Used Tested)	6426	26	5.25 Full	299.95
CMI (Used Tested)	6640	40	5.25 Full	489.95
Quantum (New)	Q540	45	5.25 Full	699.95
Atasi (Used Tested)	3046	46	5.25 Full	649.95
Maxtor (Used Tested)	1050	50	5.25 Full	649.95
Micropolis (New)	1325	85	5.25 Full	1,194.95
Seagate	ST419	15	5.25 Full	359.95

LASER TUBES



3 mw (5 mw Fallouts) with instructions
may have cosmetic defects
but are fully operational

$\$89^{95}$

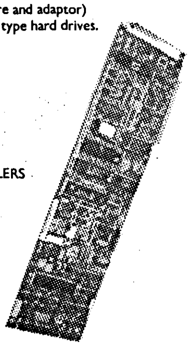
POWER SUPPLY KITS

(suitable for above)

$\$69^{95}$ (includes P.C. Board)

WARNING: Improper use of these items can be dangerous

CONTROLLER CARDS



D.T.C. 1403-D (shown with optional software and adaptor) control two 8" 851, DSDD and two 8" 1000 type hard drives.

$\$249^{95}$

D.T.C. 520-B control four 5.25" DSDD and two ST-506 type hard disks

$\$69^{95}$

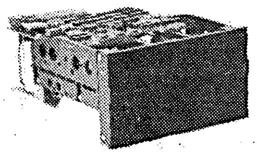
HOST ADAPTER FOR ABOVE CONTROLLERS
(for most popular Z-80 computers)
The Winif Adapter and software

$\$59^{95}$

Xebec Controller, auto boot with your software (advanced diagnostic disk) for XT PC clones with up to 33 meg.

$\$119^{95}$

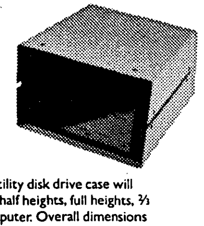
SA 450 FLOPPY DRIVES



DSDD (Brand new in boxes)
48 T.P.I.
Full Height; 5.25

$\$49^{95}$

THE SENSIBLE 5¼" DISK DRIVE CABINET KIT



☑E.M.I. Filter, 6 AMP
☑Surge Protection
☑Power Switch
☑Power Supply Shield
☑Connector and Fan Cutouts

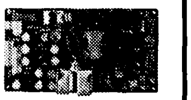
Constructed of heavy gauge aluminum, this utility disk drive case will accommodate: 5¼" hard disks, floppy drives, half heights, full heights, ⅔ heights, controller card and single board computer. Overall dimensions 10¼" x 10" x 7"
Color is TAN

ONLY ~~$\$119^{95}$~~ SALE $\$69^{95}$
Without Power Supply

GO ON LINE WITH UNITED PRODUCTS

Take advantage of the many items that space does not allow us to advertise. Use your computer to down load our weekly specials and hard to find items.
On line 24 hours a day, 7 days a week.
300 or 1200 baud, even parity, 8 bits with one stop bit.
The phone number is: (206) 682-5593
New items are added weekly. Don't miss out, call each week.

SWITCHING POWER SUPPLY



Real Power From a Famous Manufacturer delivers: 10 amps at 5V, 8 amps at 12V, measures only 8.5" by 4.5" and is only 2.25" thick

with pinouts $\$39^{95}$ (limited supply on hand)

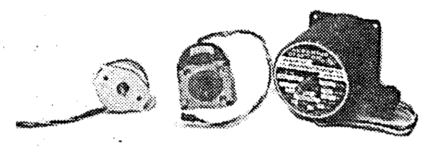
TDK model #EFX-100T-1

+ 5V @ 8 amp
− 24V @ .5 amp
+ 12V @ 2 amp
− 12V @ 2 amp
+ 24V @ 5 amp

$\$39^{95}$

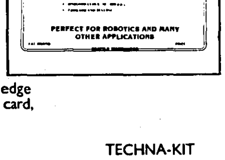
BUILDING A ROBOT? WE CAN HELP!

STEPPER MOTORS



Superior model #M061-FD02 ¼" shaft, 200 steps per rev., 35" oz. of torque 5V @ 1 amp $\$9^{95}$	Superior model #M092-FT-402 ⅜" shaft, 200 steps per rev., 200" oz. of torque, 3.0 V. @ 4.0 amps. $\$19^{95}$
Copal model #SP-57 ¼" shaft, 7.5° steps 12V, 36 OHM $\$6^{95}$	Airpax model #LA82120-MI 1/16" Shaft, 15° steps, 5V, 65 OHM $\$4^{95}$

UNIVERSAL STEPPER MOTOR DRIVER KITS


TECHNA KIT
UNIVERSAL STEPPER MOTOR DRIVER KIT
PERFECT FOR ROBOTICS AND MANY OTHER APPLICATIONS

FOR DIRECT CONTROL
FROM YOUR COMPUTER

- Control almost any stepper motor, forward/REVERSE and step sequence.
- For stepper motors rated 1.7 VDC — 12.0 VDC
- Uses a power supply with an output between 3.0 VDC and 12.0 VDC
- Modifications included for motors rated above 12.0 VDC
- Inputs/outputs are on a standard 22 pin edge card connector (.156 spacing) 4½ x 6½ card, G-10 epox.

TECHNA-KIT
cat# U.S.M.D.-C

$\$29^{95}$

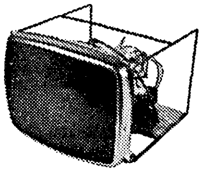
SAME SPECS. AS U.S.M.D.-C
except a stand-alone driver
- On board clock (user optional)
- No edge connector

TECHNA-KIT
cat.# U.S.M.D.

$\$19^{95}$

MONITORS

(BARE FRAME)

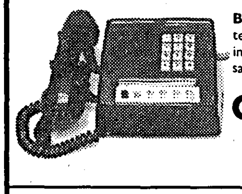


Ball Brothers TV-120 (shown)
16mhz, Xerox 820 compatible,
12" B/W, P-4 phosphor.

$\$49^{95}$

Audiotronic 964-04 TTL/XY Xerox 820 compatible 12 VDC .79 amp 9" Green $\$39^{95}$	Hitachi MD 1431 TTL/XY Xerox 820 compatible 12 VDC 1.5 amp 13" Green $\$49^{95}$
NEC 140 ANB4 TTL/XY Xerox 820 compatible 12 VDC .8 amp 5" B/W $\$49^{95}$	Audiotronic 938-34 Composite video or TTL/XY 12 VDC 9" B/W $\$49^{95}$

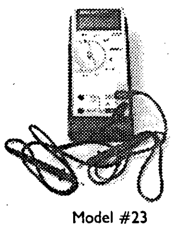
BUSINESS TELEPHONES



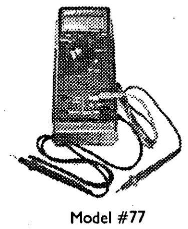
BRAND NEW 6 button business telephones all the standard features including buzzers and ringers looks the same as picture. Color, choc. brown.

CLOSE OUT
$\$49^{95}$

FLUKE MULTIMETERS



Model #23
$\$135^{99}$



Model #77
$\$129^{95}$

DC FANS

12 Volts @ 6W, 4.5"

$\$19^{95}$

united products, inc.
since 1968

1123 Valley
Seattle, WA 98109

Hours: 9-6
Sat. 9-5

PHONE (206) 682-5025

Terms: All orders are shipped F.O.B. Seattle, Wash.
C.O.D.'s are OK with cash or certified check
MasterCard and Visa accepted
On orders over $50.00 in the continental U.S. we pay the freight.
All advertised items are subject to prior sale

MICRO CORNUCOPIA

The Micro Technical Journal

Editor & Publisher
David J. Thompson

Associate Editors
Rebecca Ozrelic 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 Renee Katter

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
503-382-7643

ISSN 0747-587X

By David Thompson

Hi Pops!

I'm not really "Dad"

"I'd like to introduce David Thompson, father of us all."

Boy, that introduction got my attention. Not that I don't like being a father, but when "all" means the more than 400 computer fanatics attending SOG V, it's more than your classic scene where the little girl runs up, flings her arms around you, and wails, "Daddy."

In fact, I've sometimes wondered what I'd do if I were waifed that way. If the child were absolutely adorable, I'd probably confess my lack of self-control and start adoption proceedings. (Erin would love a younger sibling.)

But I'm not sure how one goes about adopting some 400-odd people (some of them old enough, and odd enough, to be my parents), and I'm not sure how I'd explain that many indiscretions to Sandy.

"You see, sweety, it was a dark and stormy night. The moon was full, the bats making their rounds. I was alone in the computer room when a harried looking little man in a musty white lab coat scurried by looking for raw material to use in a cloning experiment and..."

Sandy's smarter than that.

Gary set up the introductions this year, so I'm betting he's behind this. There's no question in my mind that he needs to speak next year, and I need to introduce him.

Speaking Of Gary

Gary Entsminger got married the end of August (so by the time you read this it'll be too late to save him). He's now living in Davis, California where Alison, his new bride, is attending UC Davis. Gary will still be editing and writing for Micro C (hooray, hooray).

Hard Drives

I've gotten some very good feedback on my hard drive article in issue #31. One person has written to say that his 20 Meg CMI drive has been working fine, for a whole month. Another noted that Siemens has been putting out shoddy drives so long, they couldn't be Japanese.

One thing I forgot to mention in that article. Many drives, including the Seagate 225s (20 Meg) run quite warm. If you put anything directly above them, they get hot enough to generate read errors (the platter expands as it heats). Put your winnies on top of the drive stack so they'll stay cooler; the floppies won't care.

Mike Maurice, member of a large Portland, Oregon PC group, called me the other day with the following winnie info:

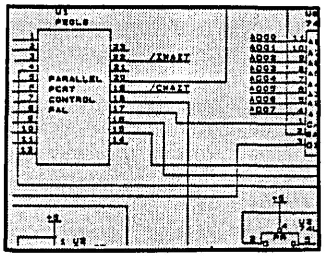
About six months ago, the Portland market was flooded with Micropolis 20 Meg drives (Tektronix had shut down a product line). Some of those drives have

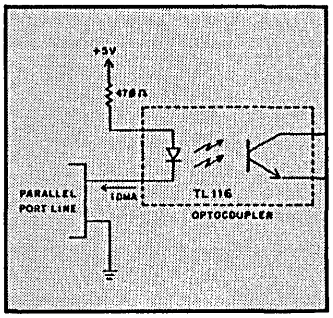
(continued on page 78)

October-November 1986
Issue No. 32

CONTENTS

Features





Columns

Future Tense

By Gary Entsminger

Free Software Catalog

At Dave Rand's public domain forum at SOG V, I promised to find you the name and publisher of a book that one of the attendees was asking about. The book is called "The Free Software Catalog and Directory." Written by Robert A. Froehlich, it's published by Crown Publishers, Inc., One Park Ave., New York NY 10016.

It's a cross-referenced listing of public domain files available for the CP/M operating system. I've found it very useful for quickly finding stuff in the SIG/M and CPMUG collections. It also provides a brief description of each file or collection. I don't hesitate to recommend it to other SYSOPs or anyone needing this kind of information.

I had a ball at SOG and enjoyed very much meeting Dave Thompson and the others there. I especially enjoyed those late night sessions at the dorm. The trip was worth making for those alone!

Dave McCrady
13324-138 Street
Edmonton Alberta Canada T5L 2B4

PC Speedup

I just finished Larry Fogg's PC speedup described in issue #31. Overall, it went well. Only one call to your tech support line with questions. My machine is a K16. It ain't fun to work on.

I had to remove some slow memory. There was one solder bridge and one wire soldered to the wrong place. Since the Speedset program didn't work with this machine, I installed a switch at your suggestion. Now everything goes like hell.

The graphics routines are much faster too. I'd be glad to help out anyone in the Spokane area who's having problems with this conversion.

John van Veen
E. 2230 11th
Spokane WA 99202
(509) 535-5131

More On Hard Disks

I have been building 8MHz V20 XT clones lately, and having researched boards, suppliers, and prices, I have a few words about hard disks.

The Tandon 3 1/2" drives are OK. They are shock mounted, which makes them particularly good for portables, but they seem very slow compared to others I've used. The rumors that these drives are self-parking are false (the same is true with the Rodimes that came with the Compacs). My understanding is that only the more expensive voice-coil drives that use much less juice to position the heads can be made to sense when the power goes off and quickly park the heads on the last track while there is still a bit of juice left in the circuits.

The Seagate ST-225 is still the old standby, but it's not shock mounted, and must be carefully bolted in from underneath. (Four #8 very short bolts hold them to the chassis, accessed by holes in the bottom of the clone cases. You must place washers in between the drives and the chassis and torque them down evenly and carefully. I usually cut an edge off the washer with wire cutters so it fits flat against the side of the chassis.) Otherwise, it won't boot up all the time, or will get senile when looking for data.

The newest technology is the Adaptek controller card which compresses data before it is stored on the drive, and expands it after it is read. Thus, a 20 Meg drive can store 30 Meg and I/O speeds up about 33 percent.

Seagate has come out with their ST-238 for use with this card, and Western Digital is about to release their own version of this controller technology.

However, I've had problems using Concurrent PC-DOS with this system. It works on the CP/M partition, but you must format the PC-DOS partition under Concurrent, and then files written by Concurrent cannot be accessed under PC-DOS and vice versa.

All of which brings up a few questions: Why is Digital Research Concurrent never fully compatible with the standards set by Microsoft and IBM, why don't they seem to care, why are they such a pain in the ass to deal with, and why doesn't anyone else come out with a fully compatible multi-tasking DOS so I wouldn't have to use Digital Research's program and put up with their lousy attitude?

Everex/Western Digital cards come with a disk containing Fmtdrive.Exe which does the physical format (rather than using DEBUG and having the controller do it). This program seems to work fine on all of the drives and controllers I've used. I've been warned that the shipdisk program on the same disk (which allows you to park more than one hard disk, one at a time, and gives you an OK message so you know the heads are really safely parked) will not work correctly with other cards.

If you haven't mapped bad tracks during Fmtdrive (drives usually come with a list of bad tracks), use Norton Utilities disktest (DT) to lock out bad tracks before entering any data. It's a good idea to run this program regularly as well as Chkdsk/f from your MS-DOS disk.

Finally, for clones or parts, check out Ishvar Technology (formerly Avatar Systems). They have the best quality and prices that I've found, and Jim is very knowledgeable and honest. Look for his add in Computer Shopper, or call or write at 43571 Sierra Drive, Three Rivers CA 93271, (209) 561-3139.

Harv Koplo
AVROM Systems
(formerly Slipped Disk Computers)
RR 11 Box 217
Springfield IL 62707

More On Mix C

In response to R.W. Odlin's letter about Mix C that appeared in Micro C Issue 29:

Mr. Odlin wrote to me after reading my review of Mix C in the MAXIMUL newsletter (the Lobo Max-80 newsletter edited by Glen Guyer, 120-C3 North Pacific, San Marcos, CA 92069) concerning a discrepancy in our timings of the string.c benchmark. (He reported 54 seconds, while I measured 880 seconds without running the object code through a Mix-supplied speedup utility and 465 seconds after speedup.)

Mr. Odlin discovered that instead of coding the for-loop in the string-length function:

```
for (p = s; *s != '\0'; s++)
  ;
return (s - p);
```

he wrote the following:

```
for (p = s; *s != '\0'; s++)
return (s - p);
```

This will definitely have an effect on the execution of the loop! (It would execute just once.)

In his letter, Mr. Odlin also noted that he could not get the register version of this function to compile. I have found a work-around for this problem. Mix C will not let you code the following sequence:

```
strlng(s)
register char *s;
```

but will allow you to code it if you create a dummy function argument and declare that first:

```
strlng(s, j)
int j;
register char *s;
```

However, all of this is academic because Mix C doesn't actually use a register for this variable. The timings are the same with or without the register declaration,

All in all, I am quite happy with Mix C. Its complete implementation allows me to try just about anything appearing in the magazines and numerous C books. Although its execution speed is quite slow, its compile and link times are fast, and for beginners like me, that's important.

Spencer R. Lepley
1655 Capital Circle SE, Lot 12
Tallahassee FL 32301

Editor's note:

Thanks for all the great information, John, Dave, Harv, and Spencer. This is one of the top rated columns in Micro C because of you.

However, I must complain:

You guys are falling down a bit. In all these letters, there's not a single reason for me to drag out my apple box and expound. What am I going to do, write a longer editorial?

I spent one whole afternoon trying to contact other editors to see how they were faring: three had disappeared, six had taken the afternoon for lunch, and the last had fallen asleep reading old listings.

Wake up! You guys can do better than this! So, I'm declaring November "Harass An Editor Month."

Get out there and harass an editor. (Editors need to make a living too.) Take your choice: a city desk editor, a newsletter editor, a political editor, a full-screen editor, a serious editor, even me.

(And if your letters are particularly good (bad?) and particularly short they'll get printed.)

Write: Harass An Editor
Micro Cornucopia
PO Box 223
Bend, OR 97709

Clone Report

I recently bought an AT clone from a company in Austin TX called Compuadd. The machine's an 8MHz with 640K RAM, a 20 Meg. CMI disk, a 1.2 Meg. floppy, an Amdek monitor, and a Hercules clone monographics board.

(continued on page 95)

Public Domain 32000: The Hardware

By Dave Rand and George Scolaro

72 Longfellow St.
Thousand Oaks CA 91360
(805) 493-1987

This is for all of you who want to put a computer in your clone or CP/M machine. (Sorry, that was a cheap shot.) Actually, this is the cheapest way known to hacker or human to build a truly powerful system (UNIX System V if you wish).

With hardware floating point that outruns DEC 11-70s, 2 Meg of DRAM, DMA, and a memory management chip, this $400 system is an incredible hardware bargain. See the SOG V article in this issue for information on a public domain operating system for this machine.

Our primary goal in designing this 32000 based system was to come up with a low cost system without compromising performance. One of the unique aspects of the design (which was first presented at SOG V) is that it has been released to the public; schematics, PAL equations, interface software, and the PCB artwork. For this reason, we named our design PD32. (We are, however, restricting commercial copying of the system, see the note at the end.)

We chose the National Semiconductor 32000 chip set (second-sourced by Texas Instruments) because it's inexpensive, is a mature product, and was designed to support high level languages. The total cost of both the hardware and UNIX System V, with which the system runs, should be under $1000.

The version of UNIX available for this board is the full AT&T UNIX System V release 2. The port was performed by ZAIAZ Communications and is AT&T and National Semiconductor approved.

About The 32000

The 32000 series includes the timing control unit (TCU), the central processing unit (CPU), the memory management unit (MMU), the floating point unit (FPU), and the interrupt control unit (ICU). Currently four CPUs are available in the 32000 series; they are the 32008, 32016, 32032, and 32332.

The 32000 series is unique in the microprocessor world in that it is the only family that maintains 100 percent upward and downward software compatibility at the binary level. This compatiblity means the end user doesn't need to change software when changing CPUs.

Of the four CPUs, we chose the 32016 for its price and ease of board routing (the board has only two layers). The 32016 still has a full 32-bit internal architecture but has a 16-bit external data path. To reduce pin count both the CPU and MMU multiplex their data and address buses. In other words, data and address information are put on the same set of pins.

Hardware Design

Now that we've covered some of the background, let's take a close look at the design. From Figure 1 we can see that the TCU, CPU, MMU, and FPU literally bolt together. To understand how the rest of the system interfaces to these four devices, we need to understand the timing of a bus cycle.

Addressing

The CPU places the logical address onto the data/address bus during T1 and pulses its address strobe (ADS) pin. If an MMU is in the system (as in this design) ADS from the CPU connects to ADS of the MMU. The MMU then performs any necessary page table lookups and translations, places the physical address onto the data/address bus, and pulses the physical address valid (PAV) pin. This is performed in the TMMU state.

This pin becomes the system-wide address strobe pin and also connects into the TCU. The TCU, on receiving the ADS signal, generates either a read (RD) or write (WR) signal during T2 and T3. During T2 and T3 the bus becomes a data bus with data either being read or written by the CPU or MMU. The final T4 state is used internally by the CPU to prepare for entering an idle bus condition (if the 8-byte prefetch queue is full) or for starting a new T1 state.

Wait States

Since the TCU generates the RD and WR signals, wait states may be requested by asserting either the CWAIT, PER, or WAITn signals. The TCU generates READY, which the CPU and MMU sample and wait on as necessary. The CWAIT is a continuous wait signal. WAIT1, WAIT2, WAIT4, WAIT8... allow from 1 to 15 wait states to be digitally selected.

We did not use the PER signal in this design. It modifies the bus cycle so the 32000 can talk to slow peripherals such as UARTs.

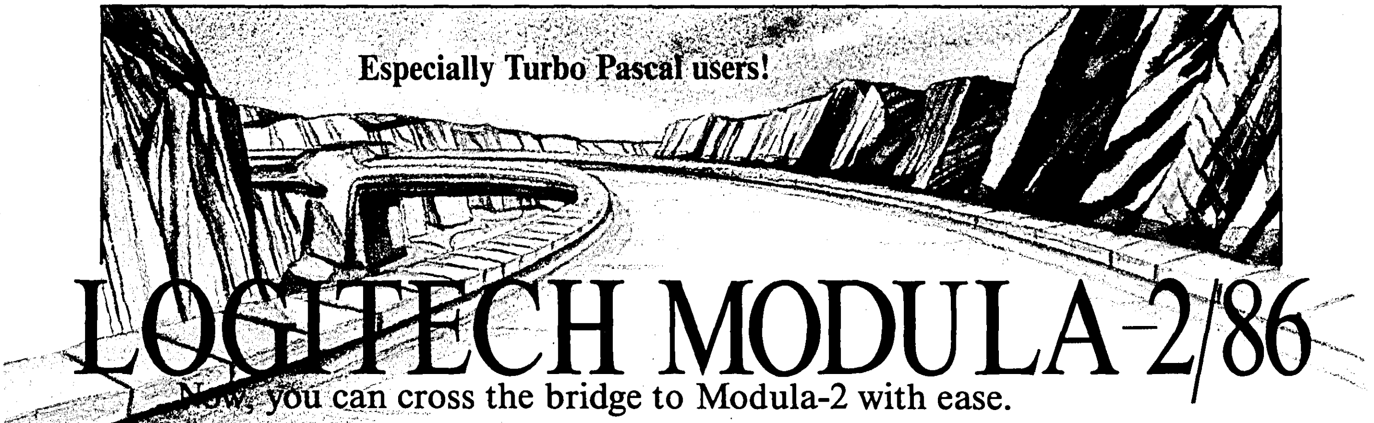
The TCU generates several other useful signals: data bus enable (DBE) which is used to enable data bus transceivers (which don't exist in this design), timing state output (TSO) which is a signal that comes out very early in T2 and lasts through T3 (used by the DRAM controller), CTTL (TTL buffered PHI1), and FCLK (crystal frequency).

Latching And Decoding

Since the data and address buses are multiplexed together we need to separate them into two buses for the EPROMS, the ICU, and address decoder PAL (U3 DEC32). Figure 2 shows U14 and U15 which latch the 16 least significant address lines using the inverted system address strobe signal (ADS). The high order address lines A16 to A23 are latched by the CPU/

(continued on page 8)

This is the Modula-2 compiler *everybody's* been waiting for...



Especially Turbo Pascal users!

LOGITECH MODULA-2/86

Now, you can cross the bridge to Modula-2 with ease.

This is Modula-2 at its *absolute* best. It's a fully integrated development environment that takes into account what you need as a programmer. Without leaving the Editor, you can call the compiler, linker and utilities.

With Logitech's Modula-2, you'll have the ability to edit several files at once, comparing, window to window, various code modules. You can even move from window to window compiling, linking, debugging and running.

The compiler has the kind of power and room to breathe that you really need in today's complex applications. It is as *easy* to use as Turbo Pascal, without your programs being limited to 64K of code.

At your command will be the libraries of modules that make Modula-2 a programmer's dream. It has essentially the same structure as Pascal with the major addition of a library organization of code modules that allow you to put together programs on a solid, block-by-block, foundation of proven code.

Whether you're working with a module of your own making, or one of the many in our library, you'll find the system by which each module is identified, described and stored an organizational masterpiece. And that's at the heart of Modula-2.

Underneath the sophisticated system is a Modula-2 compiler that is the result of years of development and proven use in industry. We run on the Vax*, and we run on the IBM PC. And the code is portable–from one to the other.

Best of all . . . you can have it right now!

Logitech Modula-2/86 Complete with Editor, $89 Run Time System, Linker, Cursor-positioning debugger, 8087 Software Emulation, BCD module, Logitech's extended library, Utility to generate standard .EXE files, Turbo Pascal (and standard Pascal, too) to Modula-2 translator *(included without charge until 8/1/86)*, and much, much more!

Logitech Modula-2/86 with 8087 support Even if $129 you haven't yet gotten an 8087 co-processor, you can still use this version.

Logitech Modula-2/86 Plus For machines with $189 512K or more. Takes advantage of the larger memory to increase compilation speed by 50%! Supports 80186 and 80286 as well as 8086 and 8088. Includes 8087 and 80287 support, too.

Window Package Now you can build true windowing into your Modula-2/86 code with $49 ease, too. Very powerful and very full, yet only 15K in size. Features virtual screens, color support, overlapping windows and a variety of borders.

Run Time Debugger (source level) Much more $69 powerful than just a symbolic RTD. Display source code, data, procedure call chain and raw memory. Set break points, assign values to variables, pinpoint and identify bugs in your source. The ultimate professional's tool!

Utilities Package Features a post-mortem de- $49 bugger for static debugging. If a program you've written crashes at run time, the situation is frozen, and you can pinpoint, in source, the cause of the error and the data at that moment. Also includes a disassembler, a cross reference utility and a "version" utility that allows conditional compilation.

Make Utility Automatically selects modules af- $29 fected by code changes for quick and minimal re-compilation and relinking. Even figures out dependencies for you.

Library Sources Source code for our major library $99 modules is now available–for customization or exemplification.

ROM Package If you need to produce rommable code, call our 800 number for further information on this package.

To place an order call our special toll free number

800-231-7717

in California

800-552-8885

Special offer until 8/1/86! $89
includes
Free! $49.95 value Turbo Pascal translator!
Now, you can take your library with you!

Yes, I'd like to take the next logical step in programming.

Please send my copy of Logitech Modula-2/86 to the following address:

□ VISA □ MasterCard □ Check Enclosed

Card Number ______________ Expiration Date ______________

Signature ______________

Name ______________

Address ______________

City ______________

State ______ Zip ______ Phone (______) ______________

Here's the configuration I'd like:

□ Logitech Modula-2/86 $89

□ Logitech Modula-2/86 $129 with 8087 support

□ Logitech Modula-2/86 Plus $189

Please add $6.50 for shipping and handling.

And include the indicated items:

□ Window Package $49

□ Run Time Debugger $69 (source level)

□ Utilities Package $49

□ Make Utility $29

□ Library Sources $99

Total enclosed $______________

(California residents, please add applicable sales tax)

LOGITECH

LOGITECH, Inc.
805 Veterans Boulevard
Redwood City, California 94063
Telephone (415) 365-9852
For European pricing, please contact:
LOGITECH SA
Box 32, CH-1143 Apples, Switzerland
Telephone 41 (21) 774545

Please call our 800 line for: □ Information on our *VAX version □ Site License and University Discounts □ Dealer and Distributor information

*Turbo Pascal is a registered trademark of Borland International

(continued from page 6)

MMU for the full bus cycle, so at this point we have a separate 16-bit wide data bus and 24-bit wide address bus.

The DEC32 PAL takes several of the address lines and generates the PPORT, INT86, EPROM, ICU, and RAMS signals. PPORT is the select for the parallel port which forms the PD32-to-host data interface. INT86 is a signal used to interrupt the host or set a flag that the host can sample to see if the PD32 has a request waiting for service. EPROM is the select line for the two EPROMS (remember the 32016 has a 16-bit wide data bus). ICU is the chip select for the ICU, and RAMS is the chip select for the dynamic memory.

DEC32 also generates a software reset signal that is combined with IRESET (the power-on reset signal from the host) which connects to the input reset pin of the TCU. DEC32 also takes several of the host signals (ISEL, IA1, IA2, IOR, and IOW) and generates a signal to clear the flip flop U2 (via pin 13). This half of the flip flop is set whenever the PD32 attempts to interrupt the host. Thus the host may honor the interrupt (the Z80 host design does not use interrupts), or it can poll the interrupt status via a status read.

Parallel control PAL U1 takes the output of the interrupt status flip flop and makes it available on ID0. U1 also generates the signals to clock data into the data latch when either the PD32 or the host performs a data write cycle, and the data enable output signal when either the PD32 or host performs a data read cycle.

The data latch is a 74AS646, a very handy chip. It has two back-to-back transparent latches (in a glass case) with tristate capability. Thus it acts somewhat like a one-byte FIFO.

The other half of U2 is used to keep track of whether the data latch, U5, is full or empty. The PAL U1 and the flip flop form the control circuit for managing the data latch, U5. This design forces wait states on both the PD32 and the host to synchronize data transactions. Wait states are asserted if either the PD32 or host attempts to write to the data latch if it is already full, or if either attempts to read the data latch if it is empty.

We designed the software protocol so that in all situations the PD32 rather than the host will be placed in wait states. Plus, we designed the PD32 so that the DRAM continues to be refreshed even if the CPU is forced to indefinitely wait (this cannot be assumed of the host and certainly not the IBM PC/AT).

Parallel Port Vs. Dual Port

In case you're wondering why we chose a parallel port interface rather than a dual port approach, the parallel port approach has many advantages in this design, some of which are listed below:

— very simple interface requirements on the host. The IBM PC/AT or Z80 or Slicer make ideal hosts for the PD32.

— very fast data transfer rate. 250Kbytes/sec on a 4MHz Z80, and up to 500Kbytes/second on an IBM AT or PC with a V20 or V30.

— easy to debug the interface via the host by output commands with a debugger while monitoring the signals on the PD32.

— data from the host is read by a software loop on the PD32. This means that the CPU can read the data and transfer it to the logical address with the MMU actually mapping the address to the physical address. Thus, data from the host does not have to be double buffered.

The last point is very important since UNIX requires that the data be transferred to the logical address. In a dual port system the host would move the data directly to the physical address. Since the UNIX System V on the PD32 supports demand paged virtual memory, logical pages are not necessarily stored contiguously in physical memory. This complicates the software in a dual port system since the host would have to read the MMU translation table entries for every 512 byte block of data it was transferring.

The ICU's Function

The ICU performs several tasks in this design. First, it generates both real time interrupts (60Hz time slicing for UNIX) and software interrupts inside UNIX, and also accepts interrupts from the host on IR13. Second, since in this design the ICU is programmed to have eight general purpose outputs, G0/IR0 to G7/IR14, we use these to perform the following:

— When the G1/IR2 output goes low, U7's /1Q output latches low causing the DEC32 PAL to switch out the EPROMs in low memory and select the DRAM. (On reset, the 32000 starts execution at memory location 0. Thus we want the EPROMs at 0 initially, but once the PD32 is up and running, we want RAM at location 0.)

— G2/IR4 if low causes the EPROM software to perform a full memory test every time the host resets the PD32. This helps you debug the memory circuit.

— G3/IR6 may be monitored with a meter, and if low indicates that the self-test performed by the EPROM has failed. This information is also output to the data latch so that the host can check the status of the PD32 after the reset command.

The DRAM and EPROM are fairly self-explanatory (Figure 4). Both EPROMs are selected whether the CPU performs a byte or word access since the EPROM can be read only. The DRAM, on the other hand, is organized as four banks of 512K bytes each. (A total of 2 megabytes is supported and automatically sized by the EPROM software.) There are upper and lower bytes (to create the 16-bit word).

The 32000 series has no alignment restrictions on data read/writes or instruction fetches. Thus the memory has to be organized so reads and, more importantly, writes to individual bytes are supported. The 32016 signals the data bus width (whether the read or write is 8 or 16 bits) using the A0 and HBE signals. If A0 is low, the low byte is active (assuming HBE is high). If HBE is low then the high byte is active. These signals are shown on the DRAM controller schematic (Figure 3).

DRAM Control

Explaining the interaction between the DP84412 (U18) and the DP8409AN-2 (U20) is beyond the scope of this article, so we'll just treat them as a single unit (refer to the series 32000 hardware databook). This design allows a wait state to be inserted into a

(continued on page 14)

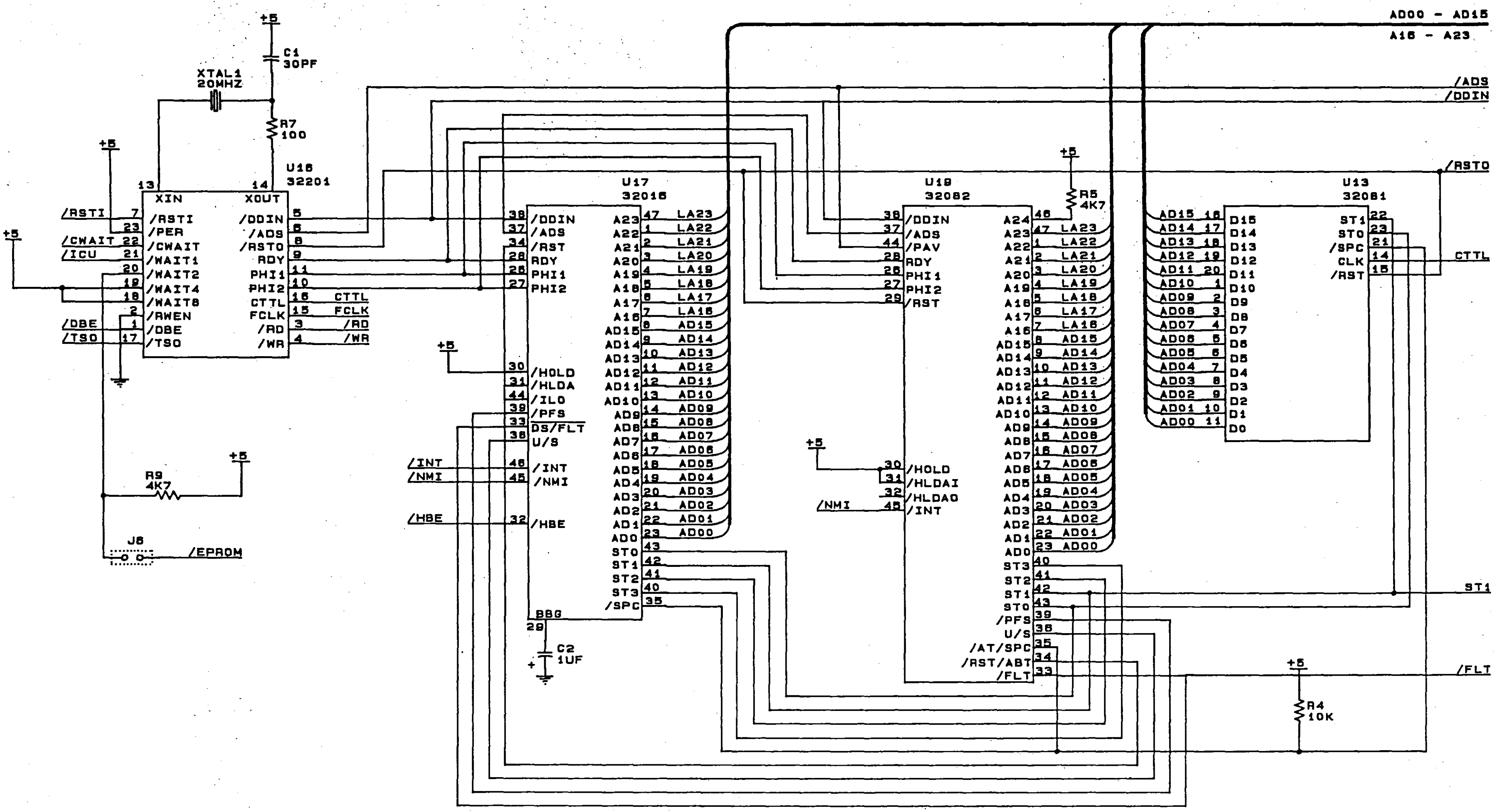
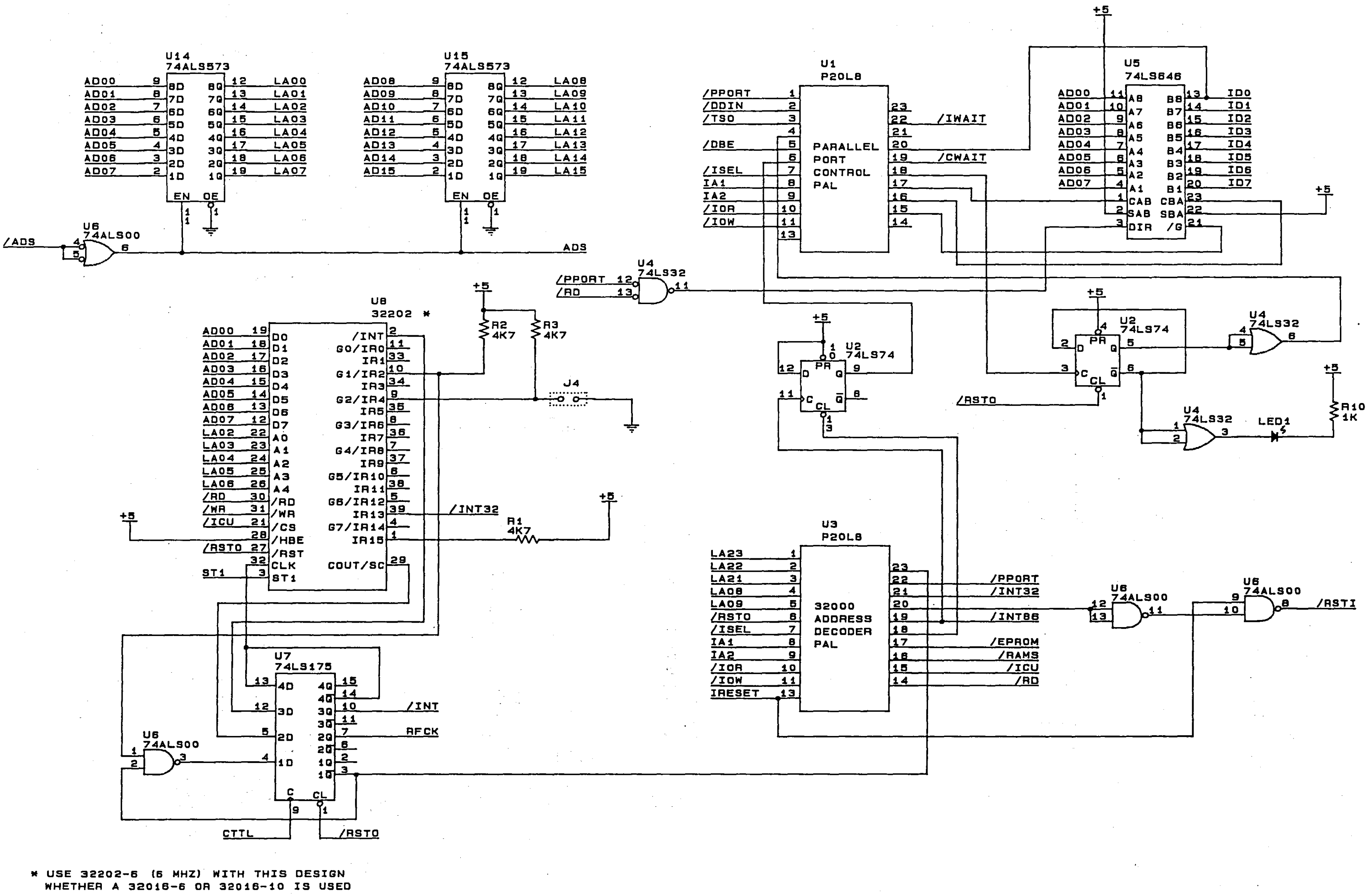


Figure 1 - Main Processor Section



* USE 32202-6 (6 MHZ) WITH THIS DESIGN
WHETHER A 32016-6 OR 32016-10 IS USED

Figure 2 - Interrupt Generation And Address Decoding Circuits

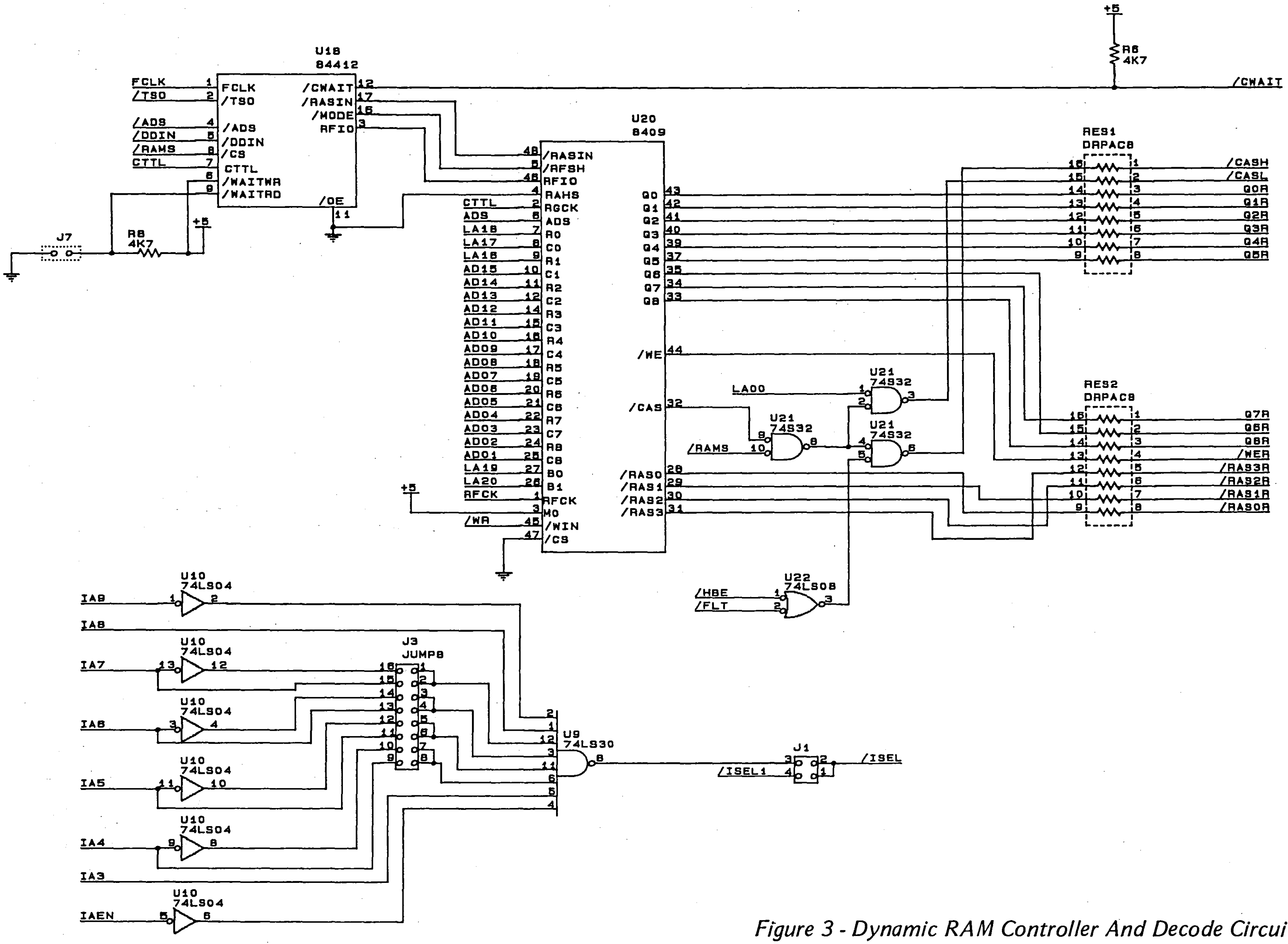


Figure 3 - Dynamic RAM Controller And Decode Circuit

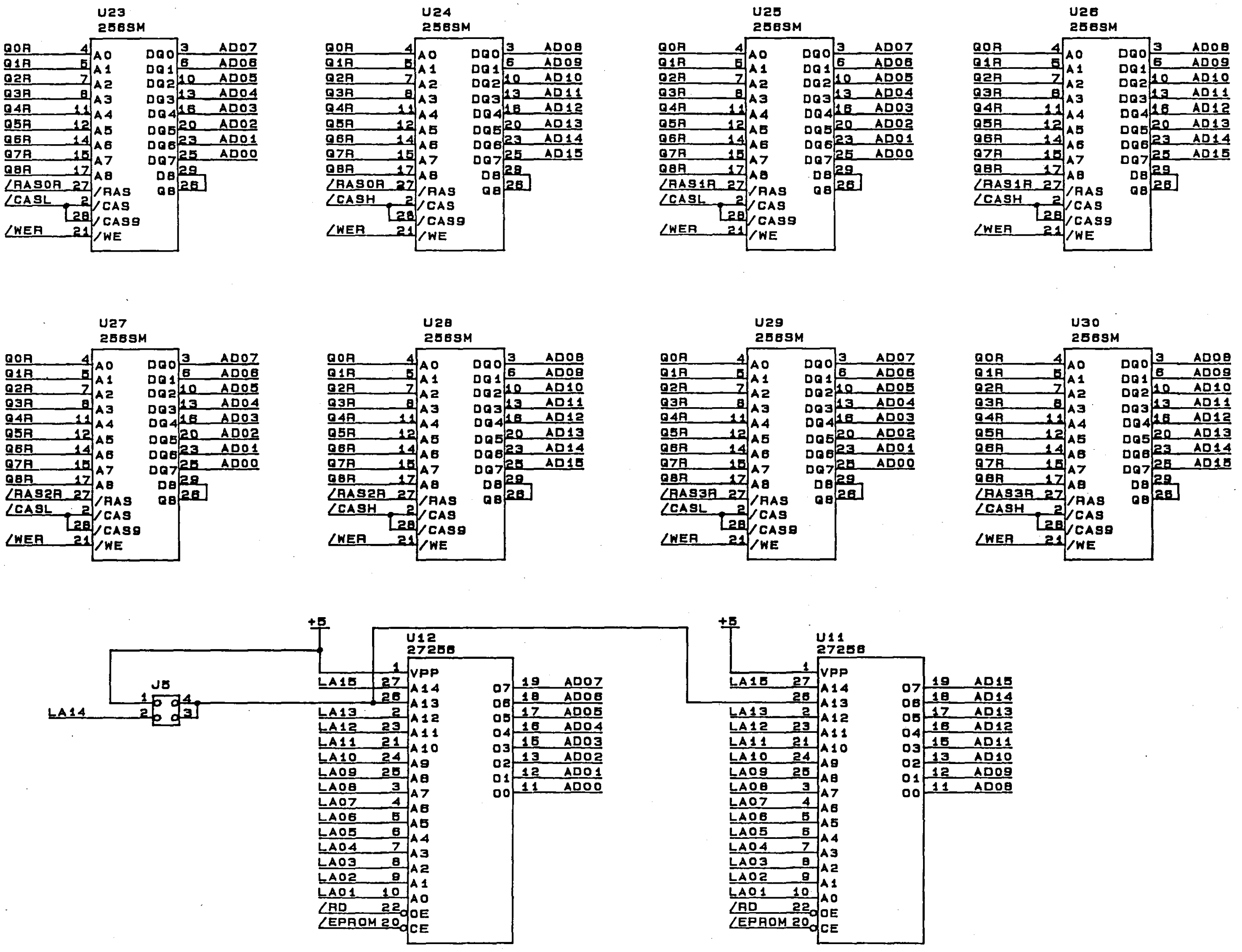
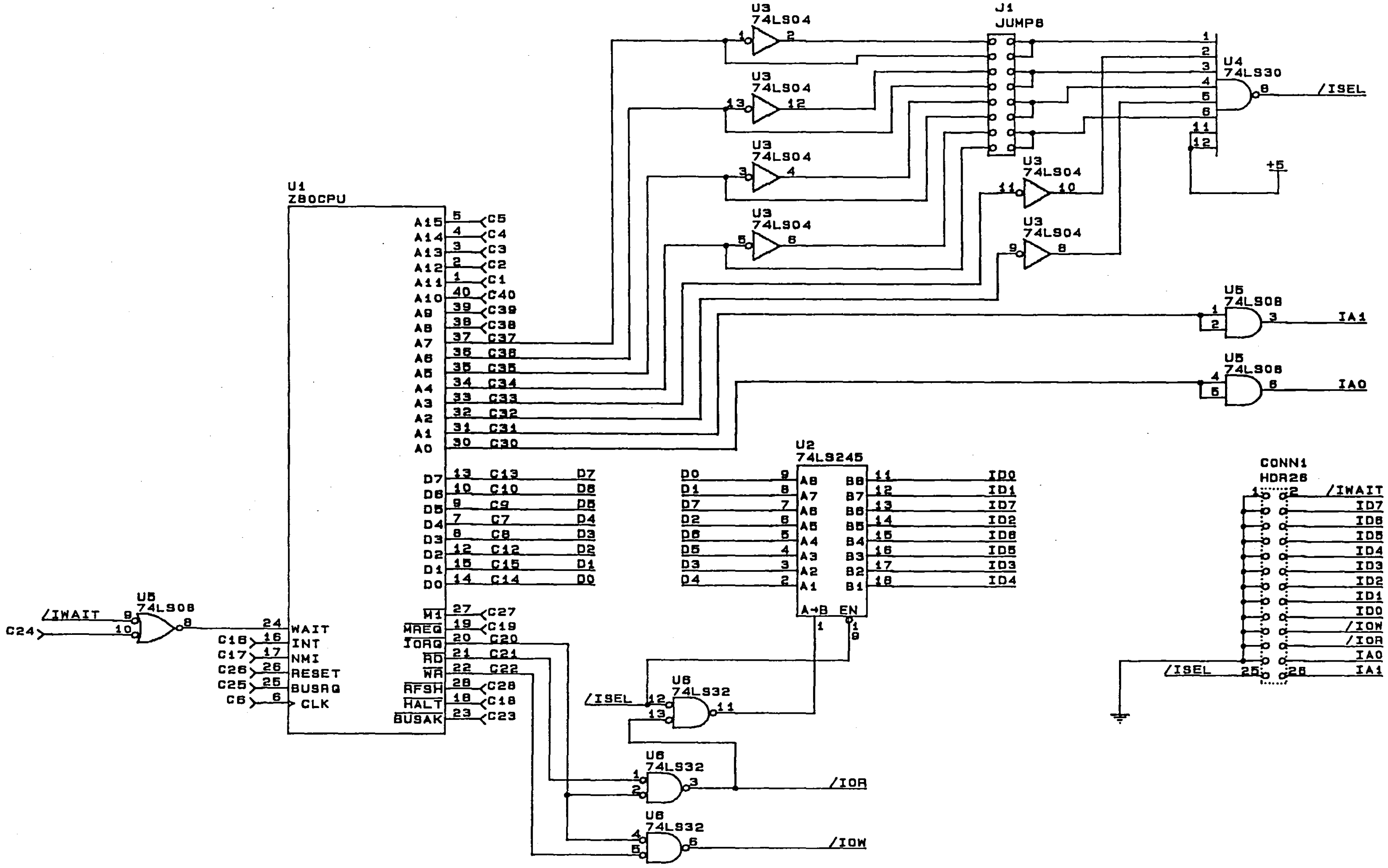


Figure 4 - Memory Circuit



* ALL PINS THAT START WITH THE LETTER 'C' EG C23
REFER TO THE ORIGINAL CONNECTIONS THAT THE Z80 CPU
MADE TO THE PCB SOCKET

Figure 5 - Interfacing A Non-IBM PC/AT Host To The PD32

(continued from page 8)

DRAM read cycle by jumpering J7. Even using 150ns DRAM, you need this wait state if you're using a 10MHz PD32 and a DP8409AN-2 (slow version). If you have a DP8419 (fast part) in the system you won't need the wait.

U20 multiplexes address lines for the DRAMs and generates the RAS, CAS, and WE signals. All the outputs of U20 that connect to the DRAMs are designed to drive high capacitance loads (typical of large DRAM arrays).

The series resistors reduce undershoot voltages caused by fast signals from U20, PCB trace inductance, and DRAM capacitive input loads. U20 and U18 also generate refresh cycles for the DRAMs with period RFCK (16 usec square wave from one of the internal 16-bit counters in the ICU).

The refresh cycles and CPU/MMU cycles are arbitrated by U18. Once a refresh cycle is started, U18 keeps asserting wait states until it's complete. The circuitry that generates the CASL and CASH signals for the DRAMs is necessary, first to support the byte read/write capability of the CPU, and second, the word read/write requests from the MMU when it's performing translation table updates or fetches.

The FLT signal from the MMU causes the CPU to release the bus. In this design FLT is used as the equivalent to HBE from the CPU. The MMU will always assert A0 low since it accesses word aligned data only.

One final point regarding chip select for the DP8409AN-2 (U20). In order to use the slow (cheap) DP8409AN-2 in systems with fast clocks (such as the 10MHz PD32) and still not violate its specifications, we keep it enabled.

Because it's always enabled it will constantly generate RAS/CAS cycles for all CPU bus cycles (including accesses to EPROM, etc.). Then we use the RAMS signal to gate off the CAS signal from U20 unless we're really accessing DRAM. Thus the DRAMs will see only a RAS cycle (a refresh cycle) when DRAM is not selected.

The final part of the schematic is the decode circuitry in Figure 3 which is used to qualify the address from the IBM PC/AT. If a Z80 is being used as the host, this decode circuitry is bypassed via J1, with the decoding being accomplished on a daughter board. CONN2 in Figure 5 is used to connect to a daughter board via a 26-line ribbon cable. The schematic of the Z80 daughter board in Figure 6 shows how to interface a non-IBM PC/AT host to the PD32.

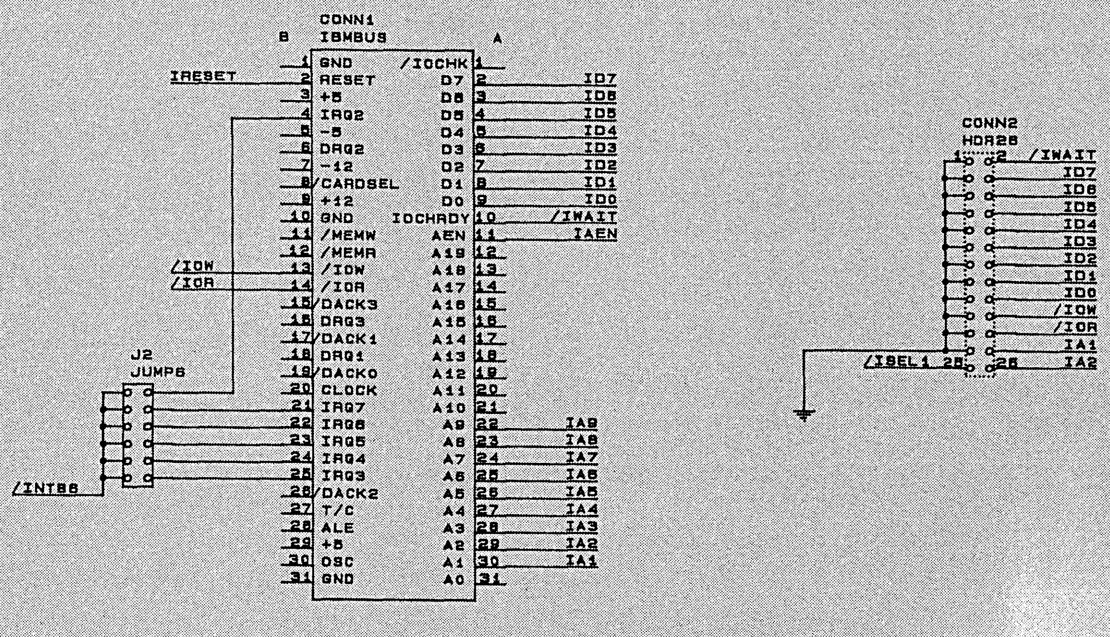
For more information regarding construction, debugging, software sources, and latest schematics, as well as the PD32 user's group, contact:

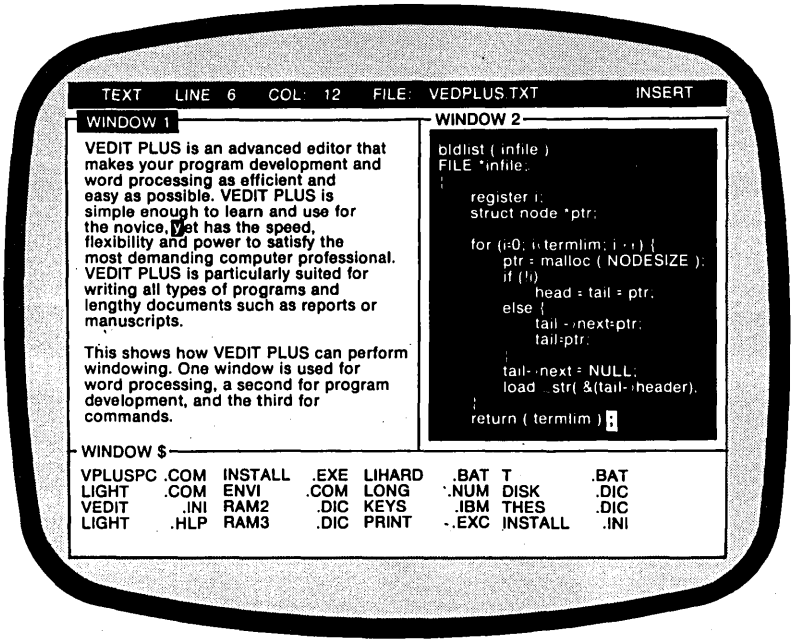
Dan Efron
8910 Westmoreland Lane
Minneapolis MN 55426
(503) 382-7643 (Micro C RBBS)

Note: George Scolaro is maintaining the copyright on the hardware design and is restricting copying to small-quantity, non-commercial users. Any commercial production of this system must be by license through George.

■ ■ ■

Figure 6 - Using CONN2 To Connect To A Daughter Board





```
TEXT    LINE  6    COL:  12    FILE:  VEDPLUS.TXT            INSERT
```

WINDOW 1

VEDIT PLUS is an advanced editor that makes your program development and word processing as efficient and easy as possible. VEDIT PLUS is simple enough to learn and use for the novice, yet has the speed, flexibility and power to satisfy the most demanding computer professional. VEDIT PLUS is particularly suited for writing all types of programs and lengthy documents such as reports or manuscripts.

This shows how VEDIT PLUS can perform windowing. One window is used for word processing, a second for program development, and the third for commands.

WINDOW 2

```
bldlist ( infile )
FILE *infile;
{
    register i;
    struct node *ptr;

    for (i=0; i<termlim; i++) {
        ptr = malloc ( NODESIZE );
        if (!i)
            head = tail = ptr;
        else {
            tail->next=ptr;
            tail=ptr;
        }
        tail->next = NULL;
        load_str( &(tail->header).
    }
    return ( termlim );
```

WINDOW $

```
VPLUSPC .COM   INSTALL   .EXE   LIHARD    .BAT   T          .BAT
LIGHT   .COM   ENVI      .COM   LONG      .NUM   DISK       .DIC
VEDIT    .INI  RAM2      .DIC   KEYS      .IBM   THES       .DIC
LIGHT   .HLP   RAM3      .DIC   PRINT     .EXC   INSTALL    .INI
```

VEDIT PLUS

MULTIPLE WINDOWS

POP-UP MENUS

KEYSTROKE MACROS

EXECUTE DOS PROGRAMS

For over six years VEDIT has been the choice of professionals who demand the most powerful editing software available. CompuView has once again enhanced this power with the latest VEDIT PLUS - you can now open windows to simultaneously edit several files, access editing functions with pop-up menus, use keystroke macros to speed editing and run other programs within VEDIT PLUS.

Whether your needs are program development, technical writing or word processing, VEDIT PLUS is your answer. VEDIT PLUS is simple enough to learn for the novice, yet has the speed, flexibility and power to satisfy the most demanding computer professional. Its powerful macro programming language helps you eliminate repetitive editing tasks.

If you take your editing seriously, you need VEDIT PLUS. With over 40,000 users, you can depend on VEDIT PLUS to perform consistently and reliably. As have GE, EDS, U.S. Navy, GM, Sperry and many others. VEDIT PLUS supports color windows on the IBM CGA & EGA and even windows on most CRT terminals. Available for MS-DOS, PCDOS, CP/M-86 and CP/M-80. List price $185.

> "To sum things up, VEDIT PLUS is a small, fast, sophisticated editor with a wealth of features and a good macro language. It offers many rewards for the dedicated programmer."
> **Computer Language, Chris Wolf, Scott Lewis, Mark Gayman 6/86**

> "VEDIT PLUS is a wholly remarkable program: blindingly fast, extremely powerful, and highly flexible."
> **Profiles Magazine, Robert Lavenda 4/86**

VEDIT PLUS FEATURES

- Simultaneously edit up to 37 files of unlimited size.
- Split the screen into variable sized windows.
- 'Virtual' disk buffering simplifies editing of large files.
- Memory management supports up to 640K.
- Execute DOS commands or other programs.
- MS-DOS pathname and CP/M user number support.
- Horizontal scrolling - edit long lines.
- Flexible 'cut and paste' with 36 text registers.
- Customization - determine your own keyboard layout, create your own editing functions, support any screen size, any CRT.
- Optimized for IBM PC/XT/AT. Also 132 column & up to 70 lines.

EASY TO USE

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

FOR PROGRAMMERS

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

FOR WRITERS

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

MACRO PROGRAMMING LANGUAGE

- 'If-then-else', looping, testing, branching, user prompts keyboard input, 17 bit algebraic expressions, variables.
- CRT emulation within windows, Forms entry.
- Simplifies complex text processing, formatting, conversions and translations.
- Complete TECO capability.
- Free macros: • Full screen file compare/merge • Sort mailing lists • Print Formatter • Main menu

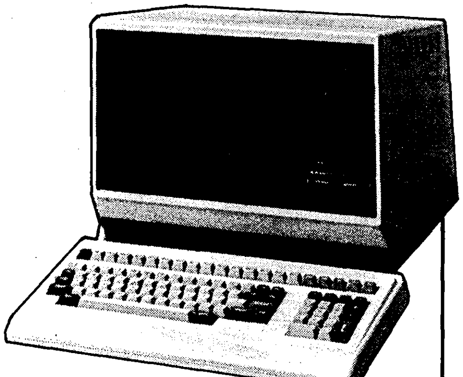
VEDIT and CompuView are registered trademarks of CompuView Products, Inc. MS-DOS is a registered trademark of Microsoft. CP/M is a registered trademark of Digital Research. WordStar is a registered trademark of MicroPro.

CompuView

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

SANYO

Prices don't get any lower.



SANYO

MBC 1200

FREE SOFTWARE FROM MICROPRO

- Thousands of FREE public domain software available
- COMPAT disk utility for over 50 CP/M formats—only $39
- Professional accounting software available
- 20 meg hardrive optional

$399

Special pricing for printers

MBC 1200 SPECIFICATIONS:

- Two Z-80A (main and subsidiary) CPUs with no-wait mode for fast execution, substantial memory capacity (RAM 64KB, ROM 4KB).
- High-resolution full graphic function with 640 x 400 dot matrix display.
- Choice of 33 or 40 line text mode.
- CP/M operating system complete with assembler, editor and all utilities.
- Easy-to-use Sanyo graphic BASIC.
- One (MBC 1200) or two (MBC 1250) internal double-sided, double-density, double-track, 5¼" slim-type 640KB formatted mini floppy disk drives.
- Keyboard with 15 user-programmable function keys.
- Interfaces for Centronics printer and one RS-232C port provided.
- Free software from Micropro: Wordstar, Mailmerge, Spellstar, Infostar, Calcstar & Basic.

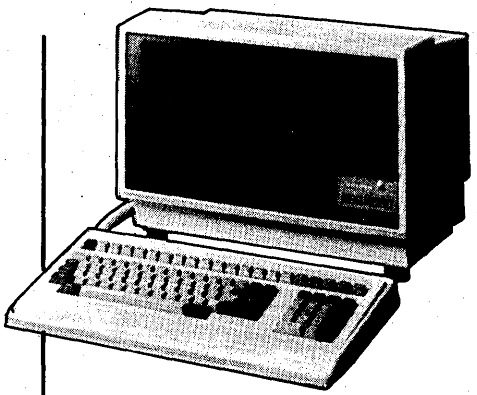
MBC 4000 SPECIFICATIONS:

- MS DOS - CP/M 86 operating system with editor, assembler and all standard utilities.
- 128KB RAM memory capacity, expandable to 640KB.
- Interfaces for one Centronics printer and one RS-232C port.
- 12" no-glare green monitor display screen for easy viewing.
- One (MBC 4000) or two (MBC 4050) internal double-sided, double-density, double-track, 5¼" slim-type mini floppy disk drives with 640KB formatted capacity.
- Keyboard with 15 user-programmable function keys.
- Free software from Micropro: Wordstar, Mailmerge, Spellstar, Infostar, Calcstar & Basic.

EDUCATIONAL DISCOUNTS

At Micro Supply Organization we offer the lowest prices on Sanyo computers and software. With prices like these you can afford the convenience of owning and operating more than one computer.

We also offer the User Support Hotline for questions concerning your computer or about software availability. Whether you need one or a dozen computers, Micro Supply Organization is the place to get them.



SANYO

MBC 4000

FREE SOFTWARE FROM MICROPRO

- 8086 Microprocesser
- Runs MS DOS & CP/M 86
- Floppy disk drive with 640K capacity
- RAM expansion to 640K available
- Utility pack including BIOS & schematics option

$499

Special pricing for printers

MSO
MICRO
SUPPLY
ORGANIZATION INC.

For our catalog with complete details and prices, send $2 to:
Micro Supply Organization, Inc.
4909 Stockdale Hwy. #180
Bakersfield, CA 93309

15% Restocking on Returned Orders

805/393-2247

GEM

MAXIMUM PERFORMANCE.

GEM-ST
Don't be fooled.
No hidden cost!

In keeping with industry trends MSO is bringing our customers high performance P.C. compatibles and accessories. The GEM-ST™ features the V20-8 chip which runs at three times the speed of the IBM-PC XT* and also runs CPM 8080 software.

FREE 5 module intergrated software

$1199

22 meg sub-system with tape back-up $599 w/purchase

GEM-ST Specifications:

CPU - V20-8 Chip & Run CPM Software
RAM - 640K
Clock Rate - 4.77 or 8 MHZ
Disk Interface and Drives -
(2) - 5-1/4" floppy drives 360K
ROM BIOS - Legal BIOS
Keyboard - AT Style Keyboard
Expansion Slots - 8 IBM* compatible slots
Power Supply - DC Voltage and Max Current
+ 5V @ 15A - - 5V @ .5A
+ 12V @ 6A - 12V @ .5A
Imput Voltage 115/230V
Meets UL/FCC Standards
Standard Interface - (1) - Serial RS-232-Port
(1) - Game Port
(1) - Parallel Printer Port
Clock - Clock calendar battery backed up
Video - TTL Hi-Res Monographics Card and TTL Monitor
Operating System - MS-DOS
Warranty: 1 year all parts.
90 day labor.

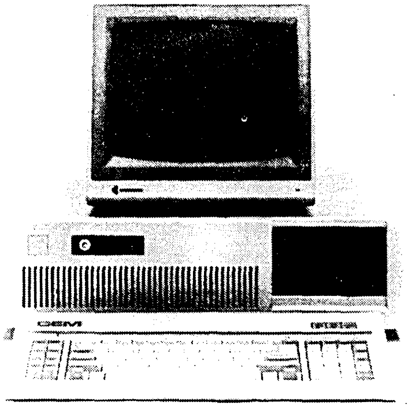
GEM-AT Specifications:

CPU - 80286
RAM - 1 Meg
Clock Rate - 8 MHZ
Disk Interface and Drives -
(1) - 5-1/4" floppy drive 1.2 Meg
ROM BIOS - Legal BIOS
Keyboard - AT Style Keyboard
Expansion Slots - 8 IBM* compatible slots
Power Supply - 200 Watt
Meets UL/FCC Standards
Standard Interface - (1) - Serial RS-232-Port
(1) - Parallel Printer Port
Clock - Clock calendar battery backed up
Video - TTL Hi-Res Monographics Card and TTL Monitor
Operating System - MS-DOS
Warranty: 1 year all parts.
90 day labor.

EDUCATIONAL DISCOUNTS

At Micro Supply Organization we offer the lowest prices on GEM computers. With prices like these you can afford the convenience of owning and operating more than one computer.

We also offer the User Support Hotline for questions concerning GEM computers. Whether you need one or a dozen computers, Micro Supply Organization is the place to get them.



GEM-AT
Performance for the power user.

The GEM-AT™ runs eight times as fast as an IBM-XT™, for your multi-tasking and multi-user needs.

Including MS-DOS 3.2 with G.W. Basic. The GEM-AT™ is a complete turnkey system with everything to plug and go!

FREE 5 module intergrated software

$1899

32 meg sub system with tape backup $799 w/purchase

MSO
MICRO
SUPPLY
ORGANIZATION INC.

For our catalog with complete details and prices, send $2 to:

Micro Supply Organization, Inc.
4909 Stockdale Hwy. #180
Bakersfield, CA 93309

15% Restocking on Returned Orders

MADE IN AMERICA

805/393-2247

Public Domain 32000: The Software

By Dave Rand and George Scolaro
72 Longfellow St.
Thousand Oaks CA 91360

There are two tricks to integrating two systems: hardware and software. Interestingly, the best trick is the software. This is the software part of the PD32 series.

One of our challenges in designing the PD32 software was to support UNIX(tm) and still make it portable enough to be moved to a number of different operating environments. Thus, the software implements a multi-channel interface to consoles, disks, printers, and the like, while using only the single 500K/second I/O channel available on the PD32.

Since the PD32 cannot be talking and listening at the same time, a protocol must be used. PD32 employs two different protocols; both will be discussed below.

Getting Started

When the PD32 first powers up, the PROM is bank switched into the PD32 memory space at physical address 0. The PROM code tests the CPU registers, initializes the Interrupt Control Unit (ICU), and waits a few microseconds for the RAM refresh to start.

Then it tests to see if that's the first power-on of the day. If it is, the PROM does 256 write cycles to the RAM, just to make sure the RAM is awake. A hard memory test follows, to check memory size and ensure that the memory is good.

If the memory fails, a 0x5a (5A hex) is output to the PD32 data port. If the memory is good, then a 0x00 is output, and the PC I/O processor knows everything's OK. Subsequent resets of the PD32 do not repeat the full self-test, but instead do a quick test of the RAM, unless you install the "always test" jumper.

Protocol #1

After the PROM completes the self-test, it drops into the first protocol. This simple protocol puts the host system in control of the PD32 and supports only three functions: reading PD32 memory, writing PD32 memory, and starting program execution. This is handled with a 6-byte packet.

```
 ---
| |      Request type (1 byte)
 ---
| |      Length (2 bytes)
| |
 ---
| |
| |      Address of data
| |      (3 bytes)
 ---
```

The request type must be (R)ead, (W)rite, or (E)xecute. The length field tells the PD32 how much data to read or write. The address field tells the PD32 where in memory to get or put the data. With this simple protocol, it's easy to do diagnostic programs from the PC side since the PC is the master and the PD32 is the slave. Figure 1 shows a sample of this protocol.

To let the other side know that data will be coming, the transmitting side always asserts the other system's

Figure 1 - Host-PD32 Communications Protocol #1

```
PC              PD32        Comments
<reset>                     PC issues a reset (hardware signal)
                <intPC>     PD32 issues an interrupt
                0x00        and sends the 'all ok' signal
<int32>                     PC issues an interrupt
'R',1000,0                  and tells the PD32 to read 1000 bytes
<int32>
[data]                      into address zero
<int32>                     PC issues an interrupt
'R',1000,1000               and tells the PD32 to read 1000 bytes
<int32>
[data]                      into address 1000
...
<int32>                     PC issues and interrupt
<W>,100,4000                and tells the PD32 to write 1000 bytes
                <intPC>     PD32 issues and interrupt
                [data]      and writes the data back to the PC
<int32>                     PC issues and interrupt
<E>,0,0                     and tells PD32 to execute the program
```

Figure 2 - Host-PD32 Communications Protocol #2

```
straddr = "Hello, world!\r\n"

PC          PD32                   Comments
            <intPC>                 PD32 issues interrupt
            'W',20,straddr,0,15     PD32 sends RPB to PC
<int32>                             PC issues interrupt
<R>,0,src,dst,15                    PC asks for data from straddr
            <intPC>                 PD32 issues interrupt
            [Hello, world!\r\n]     PD32 sends data across
                                    PC prints the string
<int32>                             PC issues interrupt
<C>,20                              PC sends completion code
```

hardware interrupt line. This gives the receiving computer plenty of time to load up registers and drop into the appropriate loops. This protocol assumes nothing but memory, and further assumes that the PC is always the master. For these reasons (plus a few more), once UNIX is running, we switch to a different protocol.

Protocol #2

The multi-channel protocol (MCP) was designed to give UNIX the best possible interface. Because it's an operating system, UNIX expects to see disks, printers, and consoles. It does NOT expect to see a single 500K/second channel! Therefore, we gave UNIX an interface with a very straightforward protocol designed to have low overhead and still provide maximum flexibility.

To initiate a request, a 20-byte Request Parameter Block (RPB) is passed, the format of which is below:

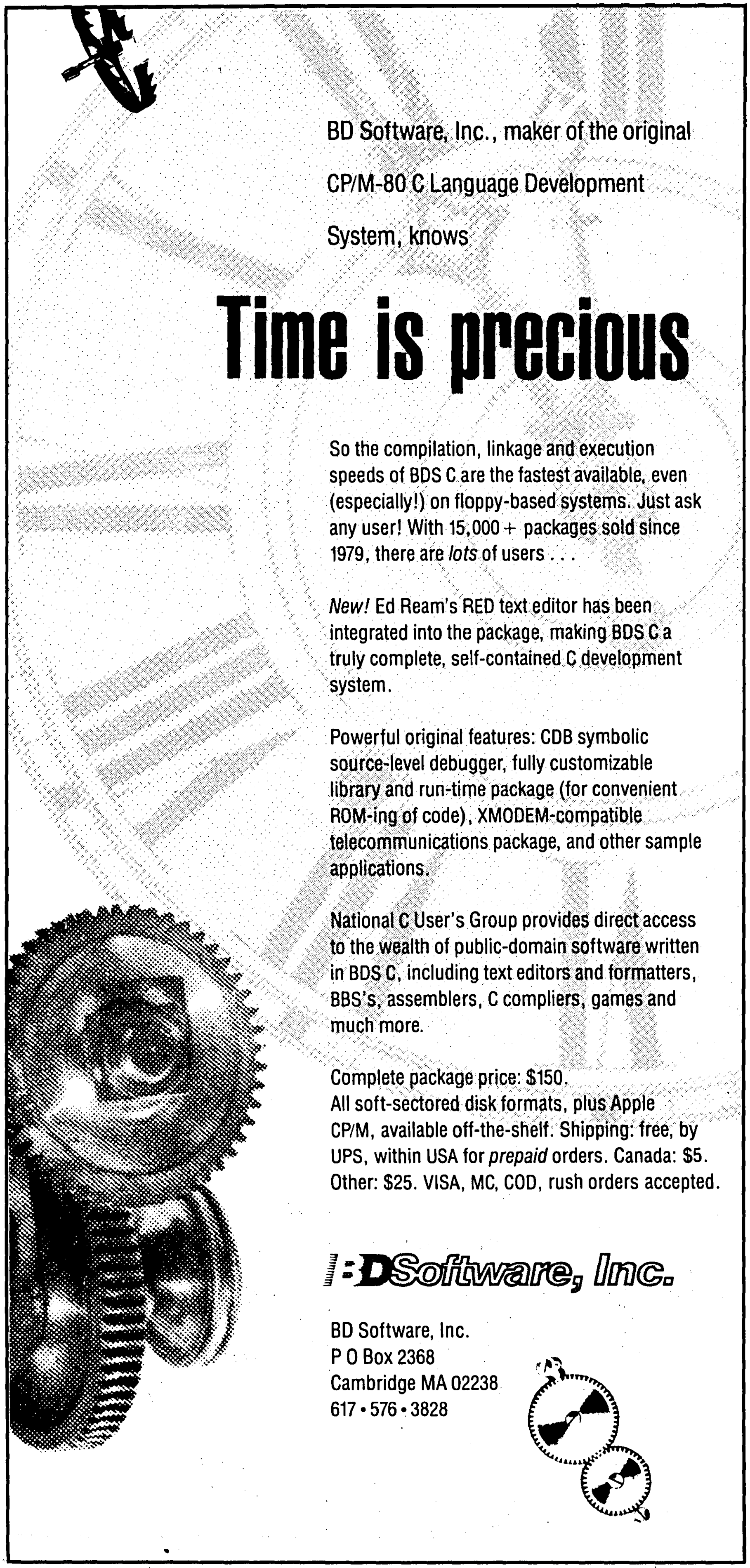
Offset	Contents
0	Request type (R/W/I)
1	Device ID (0-255)
2-5	Source address
6-9	Destination address
10-13	Block count
14-17	Data 1
18-19	Data 2

The RPB supplies enough information to allow a channel to be set up for a Read or Write operation. The Device ID specifies the channel and is assigned in the following way:

Device ID	Use
0	Memory only
1-9	Logical disk drives
10-19	Physical disk drives
20-29	Console drivers
30-49	Reserved
50-59	printer drivers
60-127	Available
128	IOCTL on device
129-254	Reserved for system use
255	Terminate

The grouping of like devices allows UNIX to use a single code fragment in

(continued next page)



(continued from page 19)

a device driver to service many physical devices.

There must also be a way of telling when an operation is complete, since transmitting the RPB does not comprise the whole transaction! The Completion Parameter Block (CPB) sent by the PC accomplishes this. Below is the format of the CPB:

Offset	Contents
0	Completion type (C)
1	Device ID (0-254)
2-5	Source address
6-9	Destination address
10-13	Block count
14-17	Data 1
18-19	Completion status (0-ok, nz-failed)

To write to the console, for example, set the request type to 'W', the ID to 20, the source address to the address of the string we want to print, and the length to the length of the string. At this point, transmitting the RPB to the PC will cause the string to be printed, then a CPB to be transmitted back to the PD32. See Figure 2.

A similar method is used for all other devices, including disk I/O. Watching UNIX come up via this protocol is interesting, however. The first 52 lines (of anything interesting) are reproduced in Figure 3. The whole thing is available on the Micro C bulletin board (503) 382-7643.

Note: Dave Rand is maintaining the copyright on the software and is restricting copying to small-quantity, non-commercial use. Any commercial production of this system must be by license through Dave Rand.

■ ■ ■

Figure 3 - This is a sample of the interface activity during UNIX load. The sample starts on the 73rd line, continues for 52 lines, and is shown to give readers a taste of UNIX code. The complete activity list is available on the Micro C bulletin board.

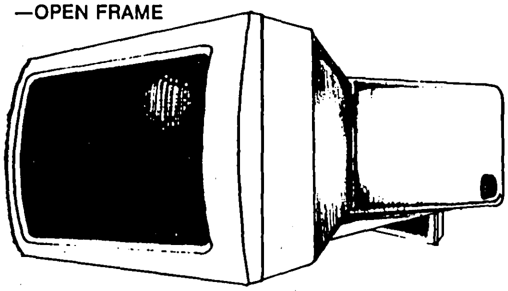
```
•
•
•

RPB: rt=W id= 20 src=  3278 dst=     0 cnt=    1
RPB: rt=W id= 20 src=  3278 dst=     0 cnt=    1
RPB: rt=I id=  1 src= 2de90 dst=     0 cnt=   12
RPB: rt=I id=  2 src= 2def4 dst=     0 cnt=   12
RPB: rt=R id=  1 src=   200 dst= 47400 cnt=  512
RPB: rt=R id=  1 src=   400 dst= 47400 cnt=  512
RPB: rt=R id=  1 src=   800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src= 14000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=  3400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=20cc00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=  1800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=126400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=128c00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=126800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=126c00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=127000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=127400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12c800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12cc00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12d000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12d400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12d800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12ac00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12b000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12b400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12b800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12bc00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=127800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=127c00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=128000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=128400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12a800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12a400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=   c00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=129800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=129c00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12a000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12c400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=37f400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=128800 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=129000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=129400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=12c000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=386c00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=  1c00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=20b000 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=20c400 dst= 47400 cnt= 1024
RPB: rt=W id= 20 src= 20136 dst=     0 cnt=    8
RPB: rt=W id= 20 src= 20136 dst=     0 cnt=   18
RPB: rt=R id=  1 src= 14c00 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=10a400 dst= 47400 cnt= 1024
RPB: rt=R id=  1 src=  4c00 dst= 47400 cnt= 1024
•
•
•
```

— MONITORS —

15" ELECTROHOME HI-RES.......$99.00
—COMPOSITE INPUT
—1000 LINE RESOLUTION
—GREEN PHOSPHOR
—ATTRACTIVE CASE

15" WELLS GARDENER...........$20.00
—24 VDC
—3 LINE TTL INPUT
—GREEN PHOSPHOR
—OPEN FRAME



12" SAMPO.....................$59.95
—AMBER, 3 LINE TTL INPUT
—ATTRACTIVE WHITE CASE
—WORKS ON IBM FOR TEXT APPLICATIONS!
—HIGH CONTRAST

9" AUDIOTRONICS...............$20.00
—3 LINE TTL INPUT
—OPEN FRAME

5" ZENITH......................$23.60
—3 LINE TTL INPUT
—GREEN PHOSPHOR
—OPEN FRAME

5" MOTOROLA..................$23.60
—3 LINE TTL INPUT
—BLACK & WHITE
—OPEN FRAME

SYNC-SEPARATOR KIT

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

— POWER SUPPLIES —

VARIABLE 0-10 VOLTS DC........$375.00
—0-200 AMPS
—VOLTAGE AND CURRENT METERS
—SINGLE PHASE AC INPUT
—MADE BY SCR

MEMOREX MULTI VOLTAGE.......$32.50
+5v 6.5 A
−5.2v 4 A
±15v 5 A

LAMBDA LXS SERIES

+5v 27.5 A	$34.95
+5v 9.0 A	$24.95

— FLOPPY DISK DRIVES —

3.5 EPSON SMD 100...............$95.00
—DOUBLE SIDED
—IBM LAPTOP COMPATIBLE
—AMIGA COMPATIBLE
—1 MB UNFORMATTED

5¼" OKIDATA ⅓ HEIGHT.........$89.00
—DOUBLE SIDED, 48TPI
—IBM PC/XT COMPATIBLE

5¼" TEC FB503.................$95.00
—DOUBLE SIDED 48TPI
—IBM PC/XT COMPATIBLE
—QUIET!

5¼" TEC FB504.................$95.00
—DOUBLE SIDED 96TPI
—720 KB FORMATTED!
—RUNS ON IBM PC/XT WITH J FORMAT SOFTWARE OR DOS 3.2 (NOT SUPPLIED)

SHUGART SA801R..............$179.00
—SINGLE SIDED, DOUBLED DENSITY
—600 KB FORMATTED CAPACITY
—NEW, NOT RECONDITIONED!

J FORMAT SOFTWARE PACKAGE..$43.95
ALLOWS USE OF 96TPI DRIVE ON IBM PC/XT 720 KB CAPACITY

WE STOCK A FULL LINE OF IBM COMPATIBLE PRODUCTS PLEASE CALL FOR LATEST PRICES

1 YEAR GUARANTEE ON IBM COMPATIBLES

SPECIAL OF THE MONTH!

EPROM PROGRAMMER FOR IBM PC
$109.00

- INTERNAL CARD WITH EXTERNAL ZIF POD
- PROGRAMS 2716 UP TO 27512
- SOFTWARE INCLUDED

— HARD DISK DRIVES —

3.5 MMI 10 MB..................$249.95
—10 MB HALF HEIGHT
—ST 212/412 COMPATIBLE
—SHOCK MOUNTED IN 5¼" FRAME
—1 YEAR GUARANTEE
—GREAT FOR PORTABLE!

HIGHER CAPACITY DRIVES AVAILABLE CALL!!

— HARD DISK CONTROLLERS —

XEBEC S1410A..................$99.00
REQUIRES HOST ADAPTOR

OMTI 5510 PC/XT CONTROLLER...$139.95
—SHORT SLOT, LOW POWER CARD
—COMES WITH CONFIGURABLE BIOS AND SOFTWARE TO HANDLE ANY 5¼" HARD DISK
—COMES WITH CABLES

PACKAGE DEAL FOR IBM PC/XT:
MMI OMTI CONTROLLER, CABLES
$375.00

MAXWELL DESOLDERING STATION

- SELF-CONTAINED (INTERNAL VACUUM PUMP)
- FOOTSWITCH
- VARIABLE TEMP/VAC LEVEL
- 10 SPARE TIPS (Extra tips not available)

$149.95

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.

PALs — What They Are And When To Use Them

By Dave Rand and George Scolaro
72 Longfellow St.
Thousand Oaks CA 91360

My experience with PALs, so far, has been mixed. They are the answer to a designer's prayers and the bane of every independent repairman.

Dave and George used PALs in the design of the PD32. The following is as intelligible a look at PALs and GALs (this is now a family magazine) as any we've seen. And, for those of you waiting for me to say, "I never met a PAL I didn't like," I hope you're satisfied.

PAL (Programmable Array Logic) is a term used to describe a family of devices first introduced by MMI and now second-sourced by several companies including Texas Instruments, National Semiconductor, and Advanced Micro Devices. PALs are a blessing for the hardware designer and at the same time a possible curse for the purchaser of a system that has PALs in it.

Drawbacks Of PALs

— If a manufacturer goes out of business and does not release the PAL equations, then replacing faulty PALs may be impossible.

— PALs have security fuses that, once blown, make it very difficult to find out how the PAL was programmed.

— PALs tend to hide certain sections of a design that if not documented make it very difficult to understand the design.

— PALs are difficult to program if you don't have access to relatively expensive programmers.

On The Positive Side

— PALs are a cheap, yet moderately secure, way to customize circuitry so it's not easily duplicated by a competitor.

— PALs help speed design and help reduce parts count.

— PALs provide speed for critical circuit areas. PALs are currently available with a maximum of 12nsec from input to output (propagation delay).

PALs (at least bipolar PALs) are programmed by "blowing" fuses in a fixed array. You can select which fuses to blow to implement a logical function by studying the switching array, or by using a PAL language.

Languages such as PALASM (from MMI) and ABEL (from Data I/O) let you specify logic functions in a high level language. The PAL language is then translated to a fuse buffer by the PAL compiler. There are also PAL simulators which let you test the design.

You supply the test vectors (a list of expected outputs for specified inputs) to test the software equation. The test vectors can also be used to test a programmed PAL (hardware can have bugs too).

Which PAL Should You Use?

Some programmers have PALASM built in, such as the Storey Systems PAL programmer (one of the more inexpensive units around). Choosing the PAL type is initially a difficult step for an inexperienced designer to take, so here are a few hints:

— Determine how many output signals you will need. Most PALs can have output pins reassigned as inputs, so this is the first step.

— Determine the number of output pins that need to be latched.

— Determine the number of output pins that must also be used as terms within the PAL. Some PALs do not allow all output pins to be used as feedback terms.

— Determine if outputs must be tristate. Not all PALs support tristate outputs.

— Determine the number of input pins that are required (excluding feedback terms).

— Finally, based on a PAL that has satisfied the above requirements, commence writing the equations. PALs allow only so many 'OR'ed terms, so the equations will also help in deciding which PAL to use.

Of course, it's normal to have a requirement that cannot be solved with a single PAL. Also, certain types of problems cannot inherently be solved by PALs. PALs implement sum of products logic by using a programmable AND array whose outputs feed a fixed OR array. Therefore, PALs can be used only when the problem can be fit into the size of the OR array (which is fixed for different PAL types).

Certain designs require the use of bipolar PROMs, especially when the number of OR terms is relatively large (more than about eight). On the other hand, PALs can be used in many situations where PROMs cannot, such as where a device with programmable tristating or registers is required.

How To Use Your PAL

Because there are so many types of PALs, an example that shows all the uses is impossible. Instead, let's look at a simple but typical use of a PAL — the PAL16L8. It has eight outputs, tristating on all outputs, speed from 12nsec to 35nsec, and various power ratings (dependent on speed). Our example (shown in Figure 1) is written in PALASM since that's one of the most common PAL languages.

This example shows most of the features of a typical PAL file. The equations may be entered in any order. PALASM uses the pin list (in order from pin 1 to pin 20) on lines 5 and 6 to assign mnemonic names to physical pins. Line 1 specifies the PAL type so that PALASM can check the validity of pin assignments and equation syntax.

In line 8 the IF (SEL) syntax specifies that this output is to be in tristate if SEL is not valid (high). The '*' sepa-

Figure 1 - Using The PAL16L8 With PALASM

```
1  PAL16L8
2  2ND LINE IS USUALLY THE USER'S PART NUMBER, NAME AND DATE
3  3RD LINE IS THE DEVICE APPLICATION NAME
4  4TH LINE IS THE USER'S COMPANY NAME, CITY AND STATE
5  /RD /WR A0 A1 A2 A3 AEN FLAG1 /FLAG2 GND
6  /D0 /OUT1 /OUT2 /OUT3 /OUT4 /OUT5 /SEL NC NC VCC
7
8  IF (SEL) D0 = FLAG1 * /A0
9                + FLAG2 *  A0
10
11 IF (VCC) SEL = /A0 * /A1 * /A2 * /A3 * RD
12              +  A0 * /A1 * /A2 * /A3 * RD
13
14 IF (VCC) OUT1 = /A0 * A1 * /A2 * /A3 * RD
15
16 IF (A0 * A1 * /A2 * /A3 * RD) OUT2 = VCC
17
18 IF (VCC) OUT3 = /A0 * /A1 * A2 * /A3 * RD
19               + /A0 * /A1 * A2 * /A3 * WR
20
21 IF (VCC) OUT4 =  A0 * /A1 * A2 * /A3 * RD
22               +  A0 * /A1 * A2 * /A3 * WR
23
24 IF (VCC) OUT5 = /A0 *  A1 * A2 * /A3 * RD
25               + /A0 *  A1 * A2 * /A3 * WR
26               +  A0 *  A1 * A2 * /A3 * RD
27               +  A0 *  A1 * A2 * /A3 * WR
28               +  A3 * RD    ;COMMENTS HAVE
29               +  A3 * WR    ;SEMICOLONS
30
31 FUNCTION TABLE
32 A0 A1 A2 A3 /RD /WR /D0 /SEL FLAG1 /FLAG2
33 ---------------------------------------
34  L  L  L  L  L   X   X   L    X     X
35  L  L  L  L  L   X   L   L    L     H
36  L  L  L  L  L   X   L   L    H     L
37 ---------------------------------------
38 DESCRIPTION
39 JUST ABOUT ANYTHING YOU WANT TO SAY ABOUT
40 THE PAL GOES HERE
```

rating terms in line 8 indicates logical 'AND'ing (product term), and the '/' indicates negation of the term. Thus if SEL is true (low on the /SEL pin), then D0 will be true (low) if FLAG1 is true (high) or if FLAG2 is true (low).

The '+' on line 9 indicates logical 'OR'ing of the product terms. The SEL term on line 8 shows an output being used as an input term. The IF (VCC) syntax on line 11 means that the output, SEL in this case, is never to be tristated.

Line 16 shows that the tristate control may be a product term (never a sum of products). Since only a single product term may be used to control tristate signal, complicated tristate equations must be constructed as in line 8 and then used as a term.

The rest of the lines (up to 31) should be studied based on the information presented above. Lines 31 to 37 are the test vectors for both simulation and functional testing of the PAL equations. The pin names that are to be tested start after line 31 and may extend to several lines as necessary. The 'L' means low, 'H' means high, and 'X' means don't care. Any input that has an 'X' will be cycled low and high during the simulation and functional test to insure that the output signal is not influenced. Output signals with an 'X' will be ignored.

After line 38, everything is treated as comment and is typically used by the PAL designer as notes.

Any pin on a PAL (input or output) that is not required may be labeled 'NC.' Note that in this example we assigned an input pin the mnemonic AEN but didn't use it in the equations. PALASM ignores this pin as it ignores NC pins. The spare pins on a PAL may either be left floating or connected to signals that may later be incorporated into the PAL equations.

Needless to say, PALs are only the beginning of user programmable logic. With gate arrays, silicon compilers, and standard cell technology, designers already can produce many compact custom devices.

Make Way For GALs

Recently Lattice Semiconductor introduced a new family of PALs called GALs (generic array logic). GALs offer the same features of PALs with the advantage of being electrically erasable. (Editor's note: I asked Sandy about this and she said GALs have more advantages than that.)

The other feature of GALs, the generic part, is that most of the PAL family may be replaced by either a GAL16V8 or GAL20V8 (20- and 24-pin respectively). GALs have a user programmable architecture word that specifies which PAL they're supposed to emulate.

A designer can use GALs to develop a working design without spending a small fortune on one-time-only PALs. The current pricing of GALs is very competitive with PALs, and since they are functionally interchangeable, either may be used in the final design.

■ ■ ■

Z sets you free!

WHO WE ARE

Echelon is a unique company, oriented exclusively toward your CP/M-compatible computer. Echelon offers top quality software at extremely low prices; our customers are overwhelmed at the amount of software they receive when buying our products. For example, the Z-Com product comes with approximately 80 utility programs; and our TERM III communications package runs to a full megabyte of files. This is real value for your software dollar.

ZCPR3

Echelon is famous for our operating systems products. ZCPR3, our CP/M enhancement, was written by a software professional who wanted to add features normally found in minicomputer and mainframe operating systems to his home computer. He succeeded wonderfully, and ZCPR3 has become the environment of choice for "power" CP/M users.

Multiple Commands per Line

You can easily use multiple commands per line under ZCPR3. Simply separate the individual commands with semicolons. For example, "PIP B:=A:*.TXT;STAT B:*.*" will copy files and then show you the STAT results.

User-Programmed menu systems

ZCPR3 comes with three different menu systems that you can use to create custom menu-driven "front ends" for your computer. This is especially useful for setting up menus for your spouse or co-workers to use the computer, as they never have to see the A> prompt. All they have to do is press a single key to run any single or multiple CP/M programs, and when the task is done, control is automatically returned to the menu (ordinary CP/M menu programs cannot do this).

Extended Command Processing

When you type a command under CP/M, it will only look for the program in the current drive and user area. ZCPR3 gives you more flexibility by additionally searching other disks and user areas when resolving commands. You have full control of this function, called the PATH. This is probably the one element of ZCPR3 that is missed most if you return to "ordinary" CP/M.

Also, ZCPR3 supports the capability of grouping all your commonly used utility programs into a library file (*.LBR). This is great for systems with a small number of directory entries per disk, as the library file only uses one entry. It also has the advantage of reducing disk space requirements for a given set of programs, allowing you to put more programs on a disk. And the programs in the library file are invokable from the command line just like any other program not in the library.

Other Features

There's much more to ZCPR3, like named directories, online help system, etc., but it can't be described on one page. If you would like more information, consider the books shown below.

Z-SYSTEM

Perhaps the only shortcoming of ZCPR3 is that it is not a complete replacement for CP/M. This is what the Z-System does. The Z-System contains ZCPR3 and an additional module, ZRDOS, and is a complete replacement for CP/M. ZRDOS adds even more utility programs, and has the nice feature of no need to warm boot (^C) after changing a disk. Hard disk users can take advantage of ZRDOS "archive" status file handling to make incremental backup fast and easy. Because ZRDOS is written to take full advantage of the Z80, it executes faster than ordinary CP/M and can improve your system's performance by up to 10%.

INSTALLING ZCPR3/Z-SYSTEM

Echelon offers ZCPR3/Z-System in many different forms. For $44 you get the complete source code to ZCPR3 and the installation files. However, this takes some experience with assembly language programming to get running, as you must perform the installation yourself.

For users who are not qualified in assembly language programming, Echelon offers our "auto-install" products. Z-Com is our 100% complete Z-System which even a monkey can install, because it installs itself. Z-Com includes many interesting utility programs, like UNERASE, MENU, VFILER, and much more.

Echelon also offers "bootable" disks for some CP/M computers, which require absolutely no installation, and are capable of reconfiguration to change ZCPR3's memory requirements. At present, only Kaypro computers have this option available.

BOOKS

We sometimes joke around the office that we are really in the business of publishing, not selling software. We have books. Lots of books. We have to have lots of books, considering how powerful our software is and the large quantity of different packages we offer. Here are our best sellers:

ZCPR3: The Manual

This is the "bible" for the ZCPR3 user. An exhaustive technical reference, bound softcover, 350 pages. Contains descriptions of each ZCPR3 utility program, a detailed discussion about the innards of ZCPR3, and a full installation manual for those doing their own installation. You could order it from B. Dalton, but why? Get it from us.

The Z-System User's Guide

For those who are not technically inclined. This is an excellent tutorial-style manual filled with examples of how to use the power of ZCPR3/Z-System most effectively, written by two highly experienced Z users. (One user is a lawyer, the other a writer; this proves that anyone can use Z and benefit from it.)

ZCPR3: The Libraries

The extensive documentation for the libraries of ZCPR3, SYSLIB, Z3LIB, and VLIB. A must for any serious user of these programming tools. Loose-leaf notebook style; easy to work with as it will lay flat on your desk.

THERE'S MORE

We couldn't fit all Echelon has to offer on a single page (you see how small this type is). We haven't begun to talk about the many additional software packages and publications we offer. Send in the order form below and just check the "Requesting Literature" box for more information.

Item	Name	Price
1	ZCPR3 Core Installation Package	$44.00 (3 disks)*
2	ZCPR3 Utilities Package	$89.00 (9 disks)
3	Z3-Dot-Com (Auto-Install ZCPR3)	$99.00 (6 disks)*
4	Z3-Dot-Com "Bare Minimum"	$49.95 (1 disk)
5	Z-Com (Auto-Install Z-System)	$119.00 (7 disks)*
6	Z-Com "Bare Minimum"	$69.95 (2 disks)
12	PUBLIC ZRDOS Plus (by itself)	$59.50 (1 disk)
13	Kaypro Z-System Bootable Disk	$69.95 (3 disks)
20	ZAS/ZLINK Macro Assembler and Linker	$69.00 (1 disk)
21	ZDM Debugger for 8080/Z80/HD64180 CPU's	$50.00 (1 disk)
22	Translators for Assembler Source Code	$51.00 (1 disk)
23	REVAS3/4 Disassembler	$90.00 (1 disk)
24	**Special –** Items 20 through 23	$150.00 (4 disks)
25	DSD-80 Full Screen Debugger	$129.95 (1 disk)
27	The Libraries. SYSLIB, Z3LIB, and VLIB	$69.00 (8 disks)
28	Graphics and Windows Libraries	$49.00 (1 disk)
29	**Special –** Items 27, 28, and 82	$129.00 (9 disks)
40	Input/Output Recorder IOP (I/OR)	$39.95 (1 disk)
41	Background Printer IOP (BPrinter)	$39.95 (1 disk)
42	Programmable Key IOP (PKey)	$39.95 (1 disk)
43	**Special –** Items 40 through 42	$89.95 (3 disks)
60	DISCAT Disk cataloging system	$39.99 (1 disk)
61	TERM3 Communications System	$99.00 (6 disks)
64	Z-Msg Message Handling System	$99.00 (1 disk)
81	**ZCPR3: The Manual** bound, 350 pages	$19.95
82	**ZCPR3: The Libraries** 310 pages	$29.95
83	Z-NEWS Newsletter, 1 yr subsciption	$24.00
84	**ZCPR3 and IOPs** 50 pages	$9.95
85	**ZRDOS Programmers's Manual** 35 pages	$8.95
88	**Z-System User's Guide** 80 page tutorial	$14.95

*Includes **ZCPR3: The Manual**

ORDER FORM

Echelon, Inc.

885 N. San Antonio Road, Los Altos, CA 94022 USA
415/948-3820 (order line and tech support)

NAME ______________________________

ADDRESS ____________________________

____

TELEPHONE _______________ DISK FORMAT ____________

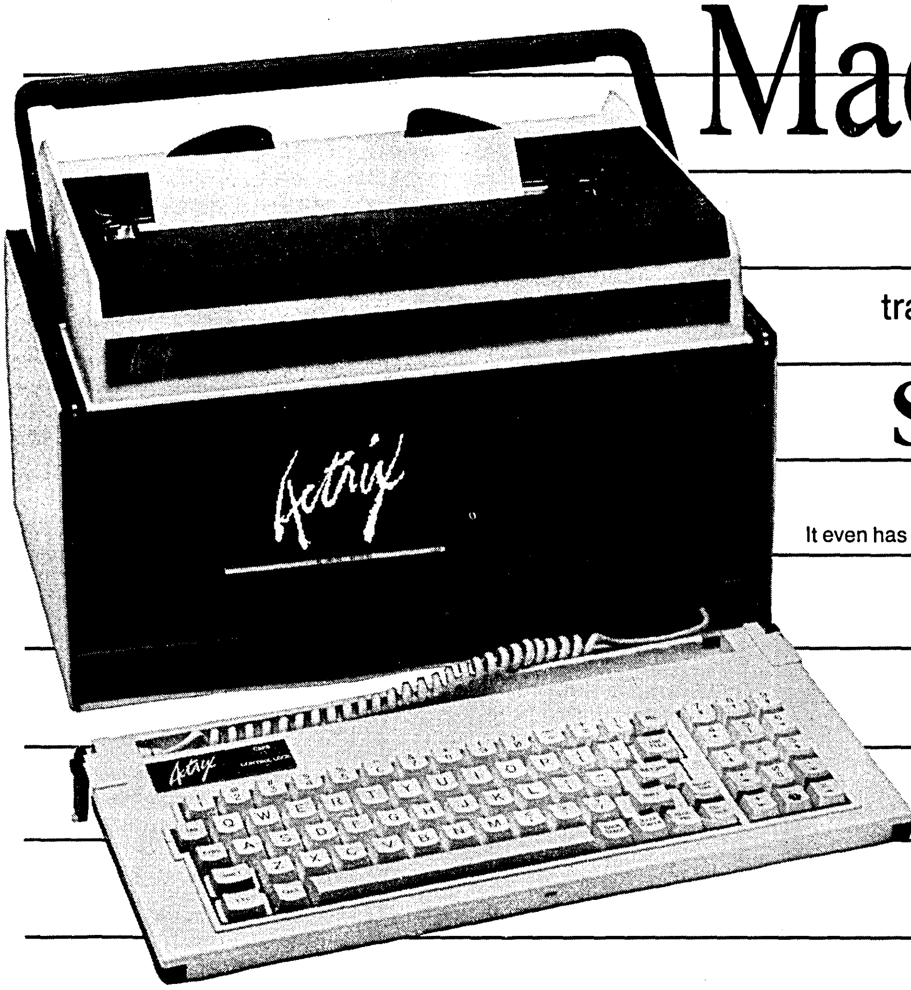
□ REQUESTING LITERATURE

Payment to be made by:
□ Cash
□ Check
□ Money Order
□ UPS COD
□ Mastercard/Visa:
____
Exp. Date _______________

California residents add 7% sales tax.
Add $4.00 shipping/handling.

ITEM	PRICE
____________	____________
____________	____________
____________	____________
Subtotal	____________
Sales Tax	____________
Shipping/Handling	____________
Total	____________

The Ultimate CP/M Machine



All this in one transportable box for

$595

It even has a 30 day money back guarantee!

More Hardware, Less Money!

- 2 double density disk drives
- 64K memory
- 80 CPS dot matrix printer
- 80 character x 25 line amber screen
- 300 Baud modem
- 4 expansion ports (2 @ RS 232, 1 @ parallel, 1 @ IEEE 488)

Phone toll free
out of CA **800-222-5401**
in CA **800-222-5403**
right now to order your Actrix.

Actrix™

More Software, No More Money

- CP/M 2.2 operating system
- C Basic and M Basic
- Perfect Writer word processor
- Perfect Speller spelling checker
- Perfect Calc spread sheet
- Perfect Filer
- P Font
- Telecommunications program

160 B Great Oaks Blvd.
San Jose, CA 95119

VISA MasterCard

By Ron Miller
1157 Ellison Dr.
Pensacola FL 32503

Taking C Into Assembly Language's Domain

Ron takes on resident programs (or temporary shell programs) this time. It's amazing what he can do with a C compiler and a couple of library functions.

Last fall the "in thing" was resident programs. The MS/PC-DOS community had just discovered the joys of tinkering with the operating system, especially using Turbo Pascal .COM files, and legions of how-to code marched across the pages of PC magazines. (Oh, the virtues of interrupt-snatching.)

Recently, I found myself stumbling back to residency to solve some very practical problems. Since the techniques involve potentials of C that are seldom discussed, perhaps a reminder or two might speed up some of your programming.

Beyond The Batch File

I've been hacking out some business software recently, mostly (I'll admit) in Pascal. Despite all its virtues, C doesn't handle random files very easily, and the El Cheapo C I own doesn't offer a genuine "double" float type. Even for government work, I need something better than 32 bits.

Anyhow, when it came time to tie programs together — my own efforts and those of the Lotus Development Corporation — I first turned to batch files.

To someone used to the limitations of CP/M's .SUB files, the MS-DOS batch utilities are like a dream. For automating dull and repetitive command line entries, such as those needed to run a C compiler and linker, they're perfect. For a salable package, however, they have problems:

1. Since .BAT files are just text files, they're open to "improvements" from folks who think they know what they're doing. I have better things to do than troubleshoot batch files for businessmen who shouldn't fiddle with PCs. Also, to justify our fees we computer types need to preserve what the sixteenth and seventeenth century Englishmen would have termed our "mystery." What they can't see must be impressive.

2. The functions available for controlling program flow are very limited and agonizingly slow. Batch file "for" and "if" utilities make one think fondly of BASIC.

3. Batch files offer little or no control over the operating environment for the programs they call. The conditions you start with will be the conditions throughout.

4. Batch files cannot call other batch files and then recover control of the program flow. Turn the corner and there's no going back.

Homegrown C-Shells

Several weeks of frustration drove me to dig out two C library functions, "system()" and "exec()." Though their exact configurations may vary from this compiler to that, they are available in every package I've seen.

On MS-DOS machines they seem to be domesticated versions of the DOS 2.0+ 4BH function call, which lets one program execute another by a sort of stack-the-pancakes effect: the subprogram is loaded (for system calls, that subprogram is COMMAND.COM) in the first free memory space above the calling program. Control is then passed to the subprogram, which does its thing and then returns control to the calling program.

(While retyping this column I finally opened up Micro C's issue #31 and saw Laine Stump's discussion thereon. Is that what they mean by parallel processing? For what it's worth, I too think Turkey is the Land of heart's desire. Are there still red pears at the Yalava market, Mr. Stump? How about the Doner kebab in Bursa?)

If all this reminds you of the discussions of resident programs you read last fall, it should. All that's different with a DOS 4BH call is that control passes directly back to the "lower" or

Figure 1 - Simple C Shell

```
main()
{
  system("copy c:\budget\taxes.wk c:\lotus");
  system("cd \lotus");
  exec("123.exe","");
  system("copy c:\lotus\taxes.wk c:\budget");
  system("del c:\lotus\taxes.wk");
}
```

Figure 2 - Copy Routine For Invalid Filenames

```
filesend(source,target)
 char *source.*target;
{
  int fout,fin;
  char buffer[BUFSIZE];  /* set BUFSIZE as big as you'd like, for speed */

  if( (fout=open(source,0)) == -1) error(); /*or however you wish to */
  if( (fin=creat(target,0) == -1) error();  /* handle errors */
  while( write(fin,buffer,read(fout,buffer,BUFSIZE)) == BUFSIZE);
}
```

resident program after the "higher" program has finished, and the resident program removes itself from memory with an interrupt 20H call when it's finished. No need to reboot the system to get rid of an unwanted patch to the operating system. Function 4BH is a shell utility, and the shell goes away when it's no longer needed.

At the simplest level, a C-shell merely offers a memory-expensive substitute for a batch file. For example, if you wanted to copy a file on the "C:\budget" directory to your Lotus directory and then call up the spread sheet (your own version 1A that can't use subdirectories), and then transfer the file back home, you could just write the code shown in Figure 1.

Still, who would bother his compiler with such trivia? A sane person would use a batch file. But while you are in C you have all the power of the language.

Suppose you wanted to keep other folks' hands off your "taxes" file. If you owned Lotus 2, you could encode it with a password to keep your mother-in-law out of your records. Even without that, however, you could make your secrets hard to get to by storing them as "tax es.wk1" in the "budget" directory. Try typing that into a "copy" command.

Though no MS-DOS resident command can handle the blank space in "tax es.wk1," both C and assembly language can handle it without a blink. For the two copy routines above, let's substitute calls to a file read and write function (see Figure 2).

Just include a filesend("c:\budget\tax es.wks", "c:\budget\tax-es.wks") and a filesend("c:\lotus\tax-es.wks","c:\budget\tax es.wks") in your C program, and you're safe. Well, safe from your mother-in-law, but not from a hot-shot kid who knows DEBUG. As we used to say back in Texas, "There haint no hoss that kaint be rode, and there haint no code that kaint be busted." But the casual snooper can enter "copy tax es.wks..." or "/fstax es" all day with no success.

Nor need one stop with name changes and transfers. Once launched

(continued next page)

Figure 3 - An On Screen Clock

```
; Module for masm. DeSmet fans could go in-line with this.
CODE         SEGMENT PUBLIC BYTE
             ASSUME        CS:CODE ,DS:DATA
             PUBLIC        _ourint,_putds
savds        DW            ?                ;storage for DS
_puts:       MOV           CS:savds,DS      ;storing DS in code segment
             RET
_ourint:     PUSH          AX          ;heck--save everything in sight
             PUSH          BX          ;but let's use the subprogram's stack
             PUSH          CX
             PUSH          DX
             PUSH          DI
             PUSH          SI
             PUSH          ES
             PUSH          DS
             PUSH          BP
             MOV           DS,CS:savds
             MOV           ES,CS:savds
             CALL          _procss    ; calling C-code
             POP           BP
             POP           DS       ;restore everything again
             POP           ES
             POP           SI
             POP           DI
             POP           DX
             POP           CX
             POP           BX
             POP           AX
             IRET                            ;and back to subprogram
/**************************/

          /* the C module */
#include <regs.h >          /* header for interrupt structures */
#define clkint 0x1c        /* clock interrupt */
#define timeaddr 0x46C     /* ticks past midnight */
#define scrseg 0xb800      /* video ram segment--you could test for this */
#define offset  3840       /* position on screen = lower left corner */
extern unsigned _code,_data;     /* obvious segment addresses */
unsigned us[2],them[2],hand,index,tick,hour,minute,second;
char time[]=
  " \160 \160 \160:\160 \160 \160:\160 \160 \160 \160"; /*reverse video */
long midtick;
/***************************/
procss()       /* interrupt handler itself: called by assembly code*/
{
  if(tick++%9) return; else tick=0;  /* look every half second */
  peek(0,timeaddr,&midtick,4);       /* get tick count from low memory */
  hour=midtick/65543;                /* 65543 ticks/hour */
  midtick %= 65543;
  minute = midtick/1092;
  midtick %= 1092;
  second = midtick/18;
  time[2]=0x30+hour/10;     /* stuffing array for poking into video ram */
  time[4]=0x30+hour%10;
  time[8]=0x30+minute/10;
  time[10]=0x30+minute%10;
  time[14]=0x30+second/10;
  time[16]=0x30+second%10;
  poke(scrseg,offset,time,18); /* Who needs an operating system, anyway? */
 }
/******************/
main(argc,argv)  /* put compound command line in quotes as argument */
 int argc;
 char **argv;
{
  int ourint();  /* declaring assembly language routine */

  if(argc != 2){
     puts("USE: clock \"command_line\"\n");
     exit(0);
     }
  putds();   /*assembly language routine for storing DS in CODE  segment */
  peek(0,clkint*4,them,4);       /* preserve old interrupt */
  us[0]=ourint;
  us[1]=_code;
  poke(0,clkint*4,us,4);         /* insert our own handler */
  system(argv[1]);           /* run applications program */
  poke(0,clkint*4,them,4); /*nice children clean up after playing */
}
```

(continued from page 27)

into this C-shell business, I realized that I could alter and restore Lotus configuration files to ensure that my files were treated as I jolly well wanted them to be, while letting my client have his own setups before and after. Memory could be altered, flags set, ports written and read, files hidden and unhidden, screen modes changed, resident modules called and erased.

If you can do it in C — and just what CAN'T you do in C? — it can be used to control the flow and access of a larger program. Only ethics and the haunting possibility that the power may go off in the middle of your best subversion, stands between you and total mastery of your (and everyone else's) programs.

Semi-Resident C

And then there are the interrupts. By altering and restoring the interrupt table down in low memory, your shell can control the operating system in which commercial software works. This is, of course, the traditional province of resident programs. If the interrupt vectors are redirected toward part of a shell program, you too have patched the operating system, albeit only temporarily, if you restore the original interrupt table when you are finished.

The old game of grab-hold-of-the-keyboard-interrupt is of course an obvious choice. The possibilities are delightful. Suppose you wanted to let somebody examine a file in a commercial word processor but not alter it. A bit of judicious interrupt 16H filtration, and the "write file" commands are not available. And all without patching the word processor itself.

When that routine's over, give them back their old interrupt 16H. Or seize the printer interrupt. I've used this trick to capture Lotus graphs in a binary file which can in turn be sent to the printer from my Kaypro (attached to my sole dot matrix printer). Another cable might have been simpler, but it was fun to see the Great Grey Lunchbox doing graphics.

To illustrate the technique, I'll do something less mundane. I'll take over the clock interrupt and use the ticks to update an on-screen clock display every half second or so. Maybe you'd like to see a clock on the screen while running Word Perfect. Though this will require a smidgen of assembly language to save some registers, if we are careful with stack variables (i.e., avoid them like the plague) we can do the real work in C. The whole program is given in Figure 3. Find a batch file that can do that.

A Few Elaborations

1. It's unnecessary to pass the clock interrupt beyond my program, because interrupt 1CH is a cul-de-sac. Things would be more complicated if I had captured the keyboard or printer interrupt.

2. A more elegant run of resident C code could set up a short stack in the data segment of the C module, so we wouldn't have to declare all the variables as statics. Unfortunately, some Cs — mine, for instance — use the DS register to address stack locations, too. It's usually simpler to use the other guy's stack.

Figure 4 - Going All The Way

```
main(argc,argv)
 int argc;
 char **argv;
{
  int ourint();
  char *malloc();
  unsigned topseg  /* bottom of freeable memory */
  struct regs rr;

  putds();
  us[0]=ourint;
  us[1]=_code;
  poke(0,clkint*4,us,4);
  topseg = 1 +  (unsigned)malloc(0)/0x10 ; /*get paragraph boundary*/
  rr.ax=0x3100;  /* terminate but stay resident */
  rr.dx=topseg + _data - _code; /*program size, in paragraphs */
  interrupt(0x21,&rr);   /* bye bye */
}
```

3. Since DOS is not reentrant, you're likely to freeze things up if you call a DOS or BIOS function from an interrupt. So I'm writing directly to the screen. In multitasking circles mine is what is called a "badly behaved program." Frankly, I think lockups are even worse.

4. In some implementations, this program will gobble up 64K+ for one little 7K semi-resident program. You can trim things down considerably if you're willing to free up the extra memory with a DOS 4Ah call before the system() call. Just remember that if you chop off the data segment, previous stack variables — argv and argc, for example — are sent into the great beyond. There's a reason why .COM files, which have no stack segment, are almost always used for resident programs.

Fully Resident C

But it doesn't have to be that way. The code in Figure 3 can easily be made permanently resident if the ending of main() is changed just a tad. Things are actually simpler. No need to save the old interrupt vector. Since malloc() returns the top of the static heap, I'll locate the first paragraph boundary above my variables with a fake allocation call and then terminate and stay resident.

Figure 4 shows the new version of main(). Just don't try this if you plan to use automatic variables in your resident C code.

You can, of course, preserve the old stack segment in the code segment, leave room for a short stack, store the new stack registers... Oh well, you get the idea. Take it from me. Switching stacks isn't much fun. I did it, but only after some know-it-all told me that .EXE files wouldn't work as resident programs. (And if I can write resident programs in C, why not device drivers in C?. I'll let you know.)

■ ■ ■

EXPAND YOUR KAYPRO OR XEROX

Most versatile. Easiest to use. Finest quality. Claims are one thing, but the proof is in our products. We guarantee satisfaction.

Nothing can improve your computer's productivity like a hard disk.

HARD DISK SYSTEMS by Emerald Microware

If you have been thinking about moving up to a hard drive, why wait any longer? We have everything you need from the host board to a complete turn-key system. Runs on the Xerox, Kaypro, or almost ANY Z80 system.

HDS Host Board – The hardware you need to connect your computer to your hard disk controller board.

- Interfaces to the WD1002 controller board
- Plugs into the Z80 socket, no other wiring required
- Switch selectable I/O addressing
- Comes fully assembled and tested

THE WINCHESTER CONNECTION by MICROCode Consulting

The most comprehensive hard disk software package that you can buy.

- Works with one or two hard disks – 5 to 64 meg
- Menu operated install, no software to assemble
- Complete testing and error handling
- Warm boot from hard drive
- BIOS drivers install above or below CP/M, your option
- Allows custom partitioning and mixed drives types
- Includes manual, format, test, park, backup, and swap utilities

WD1002-05 HARD DISK CONTROLLER BOARD by Western Digital

- 5 ¼″ profile with standard power connector (+5 only required)
- 40 pin host interface
- Connects to ST506 compatible drives
- WD2797 floppy disk controller and interface on board
- Can control up to three hard drives and four 5 ¼″ floppies

COMPLETE HARD DISK SUBSYSTEMS by Emerald Microware

- Hard disk subsystems with host, cables, software, hard drive, and cabinets with power supply. Fully tested and ready to go.
- Internal hard disk kits availble for Kaypro computers
- 10 and 20 meg units in stock, others by order
- Available for Kaypro, Xerox, Morrow, and others

10 Meg Hard Drive Subsystem	$745.00
20 Meg Hard Drive Subsystem	$940.00
Host board with software	$ 89.00
WD1002-05 Controller Board	$185.00
20 Meg Kaypro Internal kit w/KayPLUS rom	$995.00

ACCESSORIES FOR THE KAYPRO AND THE XEROX 820

Xerox 820-2 Main Computer Board	$ 95.00
Xerox 820-2 Floppy Controller board	$ 75.00
Xerox 820-2 Main board w/Floppy Controller	$150.00
Xerox 5 ¼″ Drive cabinet w/cable	$ 25.00
Xerox internal video cable	$ 8.00
Board mount power connector	$ 2.50
Parallel ASCII keyboard (not standard Xerox)	$ 15.00
Dual 5 ¼″ Disk Drives – DSDD 48 TPI, in cabinet with standard Xerox cable	$265.00
Panasonic DSDD 48 TPI Disk Drives	$114.00
Panasonic DSQD 96 TPI Disk Drives	$129.00
UniForm	$ 64.95
UniDOS for IBM	$ 64.95

We guarantee satisfaction on all of our products or your money back. Include $4.00 shipping and handling, $7.00 for COD, call for Blue Label charges. Phone hours: 8:30am to 5:00 pm weekdays; check our bulletin board – RQP/M, 5:30pm to 8:00am seven days a week, for our latest stock and prices.

WE SUPPORT OUR CUSTOMERS

INTRODUCING THE KayPLUS ROM PACKAGE by MICROCode Consulting

The most important element in the performance of your Kaypro computer is its monitor rom. With **KayPLUS** you get all of the advantages of a Kaypro 10, even on your Kaypro 2.

- Install up to four floppies and two hard drives
- Boots from floppy or hard disk
- 32 character keyboard buffer
- Automatic screen blanking
- 12 disk formats built in
- Full automatic disk relogging
- Internal real-time clock and ram card support
- Can use 96 TPI and 3 ½″ disk drives
- Includes manual, format, configuration, diagnostics, sysgen, and hard disk utilities
- No software assembly required
- Available for '83 and '84 series Kaypros – Specify Model

KayPLUS ROM Set	$ 69.95
KayPLUS ROM Set with QP/M	$125.00

Need to keep better track of your files?
Date stamp them with QP/M.

QP/M by MICROCode Consulting

Full CP/M 2.2 compatability with many more features. You've seen the replacements that eat up memory and need auxillary programs to run. Not QP/M. Fifteen internal commands, automatic disk relogging (no more control C), user area selection from colon, 31 user areas, drive search path, archive bit maintenance, and transparent time/date stamping, all in the same space as CP/M 2.2. Installs from a convenient customization menu, without any software to assemble.

QP/M Operating System, complete bootable copy	$ 80.00
QP/M without BIOS	$ 60.00

Still running single density on your 820-1?

PLUS2 ROM by MICROCode Consulting
X120 DOUBLE DENSITY BOARD by Emerald Microware

Clearly the most versatile double density package for the Xerox 820-1. Run up to four floppy disk drives at once, both 8″ and 5 ¼″ at the same time. Software compatable with Kaypro and Xerox 820. Supports all standard printers, and most add ons like the Ferguson Ram Board. You get mini-monitor functions, autoboot capability, 19 built in disk formats, and bank mode operation for more space in your TPA. Lets you run 48 TPI disks on 96 track drives. Works with UniForm and QP/M.

PLUS2 ROM Set and X120 Board A&T	$ 135.00
PLUS2 ROM Set and X120 Bare Board	$ 62.00
PLUS2 ROM Set only	$ 49.95
120 Bare Board only	$ 22.00

Other kits, parts, and packages available

EMERALD
MICROWARE

P.O. Box 6118 Aloha, OR 97007 (503) 642-1860

Controlling The Real World With Your PC

By Bruce Eckel
c/o John Fluke Mfg. Co.
P.O. Box C9090 M/S 266D
Everett WA 98206

An Introduction To Microcomputer Electronics

This is for all of you who wrote: "I don't understand half of what you put in Micro C" on your renewal forms. Bruce goes back to the beginning with this series but he'll have us doing some interesting projects before he's done.

In junior high school I joined the radio club. They tried to teach us how transistors worked by saying, "Well, you put a little current in here, and it controls the current flowing from here to there, except you have to understand these funny curves, and..." They lost me.

What I REALLY learned from that experience was that if you connected a transistor the wrong way, you could touch it and it would feel hot, but by the time it felt hot, it was already gone.

I'm now older and wiser. I understand the funny curves. I also think I understand what the radio club did wrong when they tried to teach me about transistors. They gave me too much all at once, and I couldn't do anything simple with it. Instead of experimenting a little at a time, I was expected to design an entire amplifier.

Switch First, Amplify Later

If the transistor had first been presented to me as a switch, I would have understood it. A switch is something we use every day. Once I got comfortable with the idea of a transistor, and had used it to turn some lights or something on and off, I could have been introduced to the idea that, under certain conditions, the transistor also works as an amplifier.

In this series of two articles, I'll show you how to hook a stepper motor up to the parallel port of your computer, and how to make it move. We'll learn how the stepper motor works, how to use transistors as switches, how to use optocouplers, and we'll talk a bit about power supplies.

Starting Out

I wanted to know about stepper motors, so I tore one out of an old disk drive and used a nine volt battery to make it move a step at a time, so I could figure out the pattern. Then I wondered: "How can I hook this up to my computer so it can do nifty things?"

This question led to the problem of making sure that the 24 volt supply (powering the stepper motor) could not possibly damage the five volt system (computer) that was controlling the motor.

In this article, we'll solve the problem of setting up a microcomputer, which uses five volt DC power, to control other voltages.

Next issue, I'll finish the project by describing a stepper motor, connecting it to this month's circuit, and programming the computer to make it turn.

Streams Of Bits In Parallel

We'll be using the parallel port instead of the serial port on your computer because the serial port sends out a stream of bits through a single wire, while the parallel port has a wire for each bit (and then some). Since we want to control several lines (there are four on a typical stepper motor), the parallel port is a natural.

The parallel port is like a bank of eight switches, each of which can be on or off independent of the others. By writing to this port, you can change one line while leaving the others untouched, or change several lines, or all of them at once. Thus the parallel port can be used to control eight "on-off" events in the world (we could multiplex and control more, but let's keep it simple).

But there's a problem: these control lines are always somewhere between 0 and 5 volts and can handle only a small amount of current; their value is in the information they contain and not the power they can handle. If we hook a higher voltage to them, or demand a higher current, they'll be damaged.

We could solve this problem a number of ways, but the simplest, most foolproof way is to use optocouplers (also known as optoisolators).

Optocouplers

From the outside, these look like ordinary integrated circuit (IC) chips; inside, they have a light-emitting diode (LED) at one end and a photo-transistor at the other. There's no electrical connection between the LED and the photo-transistor; they're "connected" by light.

The optocoupler works like the light communication system I made when I was a boy. Each of us held a light bulb inside a foil-lined can. The light was controlled by a telegraph key made from another can. In the dark we could "talk" as far away as 1/2 mile.

The only way to keep an optocoupler from doing its job (i.e., getting information across without letting the voltages across) is to exceed its "breakdown voltage." It's possible to have such a tremendous voltage between the LED side and the photo-transistor side that current arcs from one side to the other, and everything, including the optocoupler, is destroyed.

Fortunately, the breakdown voltage of optocouplers is generally in the hundreds or thousands of volts. The greater the distance between the LED and the photo-transistor, the greater the breakdown voltage. But be careful — if you're controlling very high voltages, be sure the coupler was designed for it.

So, we have an optocoupler; we have 5 volts on one side (from the computer) and 24 volts on the other (for the stepper motor or whatever). How do we connect it?

LEDs

Let's start with 1/2 of the optocoupler: the light-emitting diode (LED).

The voltage across an LED is the same all the time (about 1.4 volts since it's a gallium arsenide junction). So, it's CURRENT which is important.

Any diode (an LED is really just a diode) works this way; the voltage drop across the device depends on what the junction is made of (silicon, for example, is 0.7 volts) — sometimes you'll hear people talk about "diode drops." If you don't want to burn it out, you'll need to limit the current passing through it. For the garden-variety LED, that's about 20mA. But 10mA will work fine. (mA stands for milliampere or 1/1000 of an amp.)

To control the current when we have a fixed voltage, we insert a resistor in the circuit. We can determine the size of the resistor by using Ohm's Law (V = I times R. The voltage across a device is the current through the device times the resistance of the device). We want to find R, so we rearrange the equation: R = V/I. Pretend the diode isn't important; we have 5 volts and we want 10mA, so 5V/10mA = 500 ohms.

If you're a pragmatist, the above calculation is fine. You know they don't make 500 ohm resistors (the closest you can get is 470 ohms) and resistors and diodes will have errors in them anyway. We're in the ball park so we won't hurt anything, and if it doesn't quite work we can try different values.

In our little circuit, three variables affect the voltage — the computer, which outputs about 5 volts; the LED, which has a voltage across it of 1.4 volts; and the resistor, which must cause the rest of the drop (5 - 1.4 = 3.6 volts).

Once the LED is on, the flow of current is controlled by the resistor. So we apply Ohm's Law again, assuming we want about 10mA, and get: R = V/I = 3.6 volts/10mA = 360 ohms.

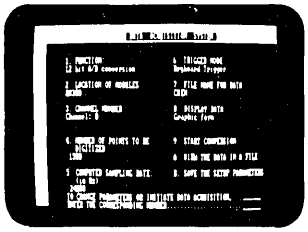
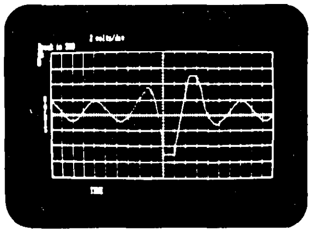
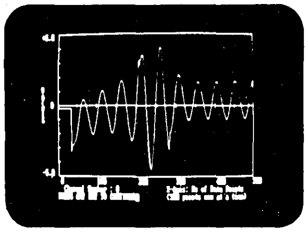
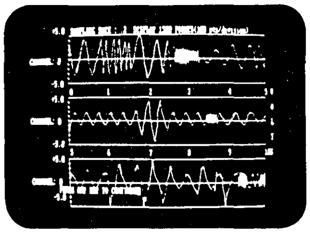
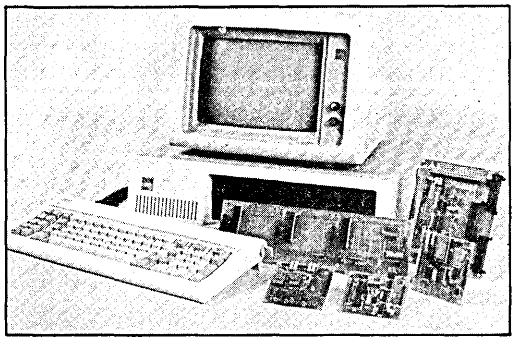
But when we try to buy a 360 ohm resistor, we find they don't make one of these either (resistors are a lot like shoes in this respect). We compromise, try a 470 ohm resistor, and, sure enough, it works. (The wattage rating of the resistor isn't important here. 1/8, 1/4 and 1/2 watt are standard ratings.)

So we've taken the signal from the computer and made an LED turn on or off inside this little black box. What do we do with the signal at the other end — the one caused by the light hitting the base of the photo-transistor, causing it to turn on and off?

My first impulse was to connect the collector and emitter like two ends of a switch into the stepper motor circuit. Unfortunately, the optocoupler is good for only one job: keeping voltages apart. Its output still can't control anything significant.

Thus, once you get the signal across,

(continued next page)

(continued from page 31)

you still have the problem of passing enough current to move the stepper, close your relay, activate the solenoid, light the lamp, or whatever. The optocoupler won't handle enough current to do what we want, and even if it did, the light hitting it from the LED isn't enough to turn it on all the way. Here's why:

A Little About Transistors

The transistor can be used two ways: as a switch, or as an amplifier.

We're using it as a switch, so we must turn it on all the way (Wham!) or turn it off all the way (Slam!). In between Wham and Slam is the "linear region" (or, as I came to know it, the "region of funny curves"). This is where we use it as an amplifier. If you're trying to use it as a switch and get caught in this region, it doesn't work very well because it isn't turned on all the way.

So we need to take this signal, which is "Really Off" and "Sort of On" and turn it into "Really Off" and "Really On." Fortunately, there's a device designed for this (the Darlington transistor) which comes in its own package with a heat sink.

A Darlington transistor (or Darlington pair) is actually two transistors hooked up one right after the other to look like one transistor which requires only a very little signal (like the output of an optocoupler) to turn itself all the way on or off (see Figure 1).

The first transistor is good at turning on, while the second is good at passing lots of current (but not as good at turning on), so together they make one transistor which is good at turning on and good at passing current. Perfect for what we want.

The signal comes out one of the lines of our computer's parallel port through a resistor to control the current and through an LED, which is one-half of the optocoupler. The signal turns into light (for a one), or no light (for a zero), travels through the optocoupler, strikes the base of the phototransistor, and turns it on.

The photo-transistor drives the base of the Darlington pair, and Voila! We've isolated the computer from the 24 volt system it's going to control. (Actually, by selecting the right Darlington and optocoupler you could control much higher voltages.) The collector and emitter of the Darlington are like the two wires of a switch.

The Circuit

Let's walk through the circuit I built (see Figure 2). We'll put some LEDs on the outputs and use the computer to turn them on and off, and that will be the end of this article. If you simply must hook a stepper motor into your circuit before next issue, make sure you put some diodes in the circuit to prevent the stepper coil inductance from punching holes in the Darlingtons. I'll explain that completely in the next article, but I thought I'd warn you.

A circuit diagram doesn't show everything. For example, it doesn't show how to get the signals from your computer to the circuit, and what the circuit is built on.

To get the signals from your computer to the circuit, you have to find a connector that mates to your computer's Centronics port. I found this no easy task, because some people don't know what a "standard, usual kind of Centronics connector" is, and other people don't know what a "36-line micro-ribbon connector" is (they're the same thing).

And, just to be different, a lot of IBM parallel connectors are the "DB" types, but not DB-25s (25-pin connectors) like the RS-232C connectors. Instead, they're DB-36s. Fortunately, the Radio Shack deities generally smile upon connector problems for computers, and you can find them there. (Editor's note: Most of the clones are now using female DB-25s as parallel connectors. All you need is the male DB-25 and a guide to which PIO output lines are connected to which DB-25 pins, although a simple voltmeter will tell you pretty quickly which lines are active.)

Can You Slice Bread Around Here?

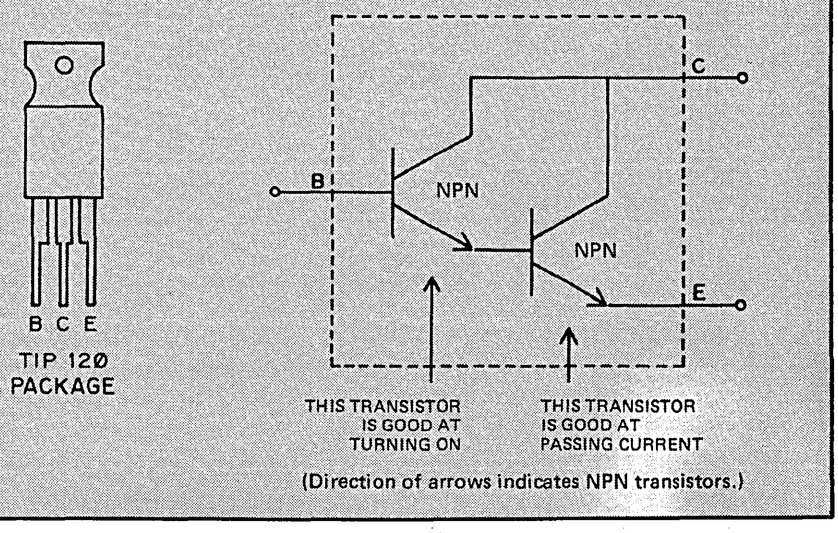
Once you get the connector and solder the wires on all the pins of the connector, and then figure out what the wires mean (more about that in a minute), what do you do with them?

Well, the early pioneers would (allegedly) go down into the kitchen and steal one of Mom's breadboards, stick little nails, wires, and blobs of solder on the board, and try out their experiments (thus the term "breadboarding"). (Also the term "disinheritance.")

Mothers were no doubt responsible for the development of these neat chunks of plastic with lots of little holes in them that you can push wires into. They're still called breadboards, of course.

Breadboards now come in strips about two inches wide and eight inches long. Don't be fooled by the people who want to sell you a bunch of these strips glued down on a metal board for a lot of money. They're the

Figure 1 - A Darlington Pair (often shown as a single transistor)


C
B
NPN
NPN
E
B C E
TIP 12Ø
PACKAGE
THIS TRANSISTOR IS GOOD AT TURNING ON
THIS TRANSISTOR IS GOOD AT PASSING CURRENT
(Direction of arrows indicates NPN transistors.)

same strips; buy your own and glue them on a board.

Also don't get fooled by the expensive boxes filled with carefully stripped wire of all sizes, lengths, and colors. Those are only for professionals; definitely out of our league. Buy your own wire and strip it yourself, but make sure it's solid wire, not stranded — ask the salesman or look at the professional box of wire for the gauge (thickness of the wire — a bigger number means a smaller wire).

With a breadboard you can try out circuits without soldering. Just figure out which holes are connected to which other holes, push the wires into the holes, and presto! — you're an electrical engineer! (Perhaps you should memorize a few equations in case there's a test.)

If parts don't work, or you hook it up wrong, you don't have to desolder or snip or anything — just pull stuff out of one hole and push it into another. This way you can very quickly try out ideas. (A lot like FORTH.) Only after the design works do you get out the soldering iron, or printed-circuit etching kit.

Centronics Port Wires

Usually, you can find the numbering of the wires on one of the connectors, but it's printed small and is easy to miss. The Centronics port wires 2-9 represent bits 0-8. The rest of the lines are either grounds or handshaking lines. You want at least one ground, but we'll ignore the handshaking lines.

If your connector has a different configuration, you'll have to look up its pin configuration in one of your manuals, or figure it out by tracing the wires with an ohmmeter or continuity tester. Your printer manual will have the wiring of its connector.

About Power Supplies

One more problem: we have effectively isolated our computer from the part we're controlling. So effectively, in fact, that we left all the power on the computer side! We need some power on the "outside" too.

We can solve the problem with batteries or a power supply. Batteries are okay, but you can get frustrated when they run down in the middle of the night. It's nice to have a power supply.

I found a good switching power supply with several different voltages for around $30. Astec makes it, and you can get it by mail from several outlets. I got mine from American Design Components, 62 Joseph St., Moonachie, NJ 07074. JDR Microdevices, a reliable mail-order firm, also carries them. (Note: stepper motors taken from 5 1/4" drives run on 12V rather than 24V. There are many more supplies to choose from if you don't need 24V.)

This power supply doesn't come in a metal box — you just look at its guts. For decency's sake I recommend you put it in a box. (There's 110V AC on the input side of that little beastie — not a good thing for fingers, little or big.)

So Now You Have A Lab

Almost. You have a power supply, a breadboard, and a connection to your computer so you can quickly generate digital control signals with software (this is really nice, since otherwise, you might spend a lot of time making circuits to do the same thing). But you still need a meter.

I work for Fluke, so recommending a meter from them probably sounds like hype. But I use the meters, and have talked to people who have designed them, and have seen what else is out there. In addition, customer service is very good about fixing/replacing broken meters. I recommend one of the LCD "70-series" meters, which you can get for around $100. If you're not ready to spend that much, get an inexpensive one at Radio Shack. They aren't as nice and won't last as long, but I got a lot of use out of them as a boy.

Back To The Circuit

I connected an LED and resistor to the emitter of the Darlington transistor (the emitter always has the arrow on it, and the arrow points in the direction of "conventional" current flow; i.e., flow of positive charge, or holes). (Again see Figure 1.)

(continued next page)

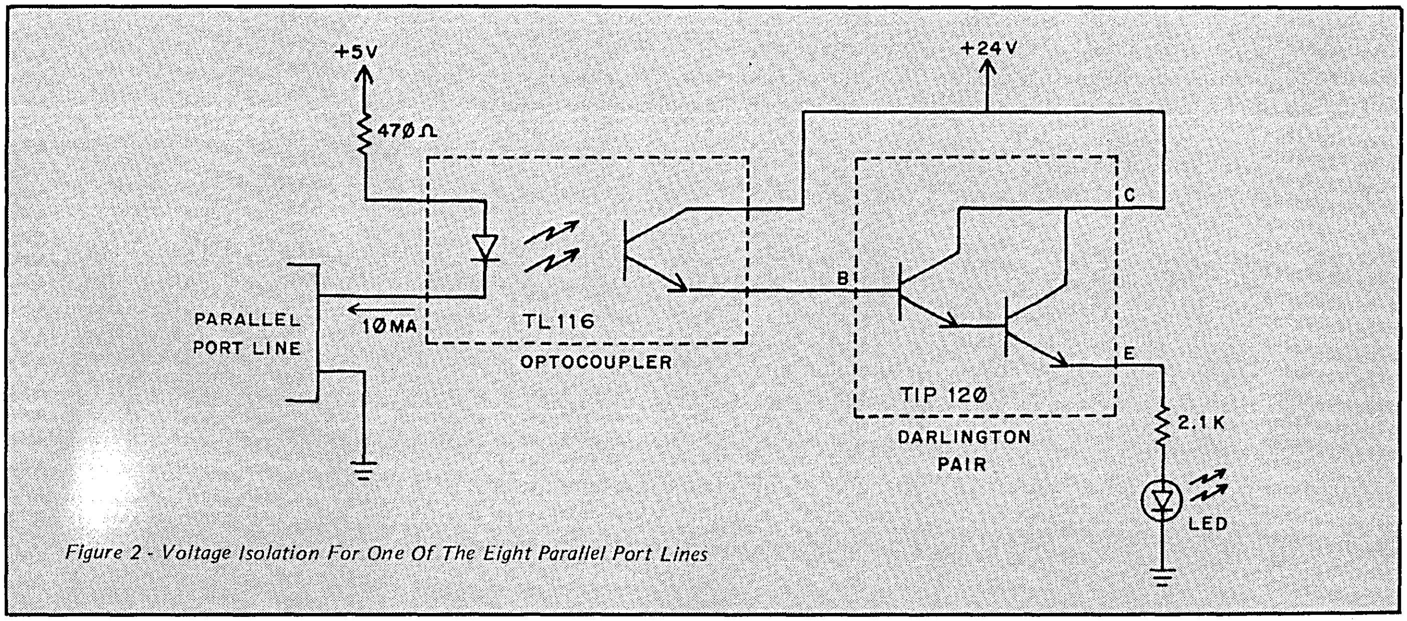


Figure 2 - Voltage Isolation For One Of The Eight Parallel Port Lines

(continued from page 33)

The "real" charge carriers, the electrons, are going the other way. This is an NPN transistor. I remember it by saying "No PiNpoint" — the pin isn't pointing at the flat spot of the transistor. When it is, that's a "PiNPoint" — PNP.

For now, this LED and resistor will be our output circuit. Next issue, we'll replace it with one of the stepper motor coils and a diode (there'll be three other circuits for the other three coils). The resistor is there, as before, to prevent too much current from going through the LED (these LEDs have no self-control).

The collector is connected to the source voltage, which will be 24 volts next issue (or whatever your stepper needs), but can be anything now.

When the Darlington is off, no voltage appears on its emitter pin. When it's on, it has a voltage drop between its collector and emitter of about .2 volts. So the voltage appearing on the emitter is 24 - .2 = 23.8 (24 volts if you're a pragmatist).

We need to choose a resistor which will keep the current down to about 10mA and not fry our LED. (The magic words "volts," "current," and "resistor" mean bring out Ohm's Law! — 24V/10mA = 2400 ohms, or 2.4K ohms.)

Yes, you guessed it — there isn't a 2.4K ohm resistor! (Unless you want to pay more, and go from the El Cheapo 10 percent tolerance to the more expensive 5 percent tolerance.) We need to go up a little, to 2.7K, or down a little, to 2.2K. It doesn't really matter, but let's go down.

And that's the circuit (you'll probably want to duplicate it for each of the eight parallel port lines). All that's left is to turn the LEDs on and off.

Are We Having Software Yet?

You probably thought you could just PIP a file to the parallel port and watch the LEDs flash, right? Well, so did I. But it doesn't work that way. Some LEDs will turn on and the whole thing just sits there.

The port is waiting for a handshake (remember, this is a business machine). When you send something through the operating system, it uses a device driver, a little piece of software which knows how to work the buffer and knows exactly where the parallel port is (not just a device name, like PRN:, but the location in the memory or the input/output address).

Figure 3 - Using the OUT PORT and BYTE Commands in BASIC

```
10 ' Random output at random intervals
20 K$ = "" : WHILE K$ = "" : K$ = INKEY$ :'Press any key to quit
30 OUT &H19, INT(255 * RND) :'Send out a random
                              Number between 0-255
40 FOR I = 1 TO INT(300 * RND) : NEXT :'Wait a random period
50 WEND

Or --

10 ' Bouncing up and down; delays adjusted to look like a
     real bouncing ball.
20 K$ = "":WHILE K$="" :K$ = INKEY$
30 FOR I = 0 TO 7 : OUT &H19, 2^I :'Selects I-th bit
40 FOR J = 1 TO I*15 + 5 : NEXT : NEXT
50 FOR I = 7 TO 0 STEP -1 : OUT &H19, 2^I:'Bounce back down
60 FOR J = 1 TO I*15 + 5 : NEXT : NEXT
70 WEND
```

It takes a byte out of the bag, sticks it in the port (our LEDs just came on) and waits. Waits for what? Waits for the special "handshake" line to go high and low which tells the computer that it got that first byte and it's ready for another. The MS-DOS device driver is watching that line, so as long as we go through the device driver, we either learn to be courteous or we wait.

The solution is to go AROUND the device driver, by addressing the port directly. Here's where you get to do detective work.

I have a Kaypro 2X, and I know where its parallel port lives (19 hex). For XTs and other clones the common parallel port addresses are 3BC hex and 378 hex.

Look first in your computer manuals; find the most technical looking part with lots of tables, strange words, and numbers, and look for the port location. Call someone who has a computer like yours. Call the dealer. Call the manufacturer. Persist.

Once you have the port address, use the command in your programming language which will put a byte directly to a port (assuming your ports aren't memory-mapped, in which case you would simply stick things in that memory location).

I'll use BASIC and the OUT PORT and BYTE commands in my examples. (See Figure 3.) In Turbo Pascal, you'd use the "port array" like this: Port[PORT] := BYTE;

You may need to tweak the timing constants on these examples, since your machine may run faster or slower than mine.

Now Do Something

You now have switches to the outside world and a programming example. You can use these switches to drive relays to handle more current and bigger voltages.

Next issue, we'll see how a stepper motor works and how to hook it into our system and program it.

■ ■ ■

ERAC CO.

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

KAYPRO EQUIPMENT

9" Amber CRT	$39.00
9" Green Monitor	35.00
12" Green Monitor	45.00
81 Series ROM's	7.00
Keyboard (Kaypro)	75.00
Keyboard Siemens w/case	65.00
Wiring Harness	7.50

KAYPRO MAINBOARDS

K2 With Sockets	$39.00
K2 Populated - Tested	129.00
K2 with PRO8-3 Mod.	239.00

This one runs at 2,4 & 5 MHz, up to 4 drives SS, DS, or Quad drives. New ROM has screen dump and step speed select. With manual & software, tested and guaranteed 30 days.

K2 PRO8-3 Mod. EXCHANGE	149.00

We will update your K2 or K4 as above. Mail working Main board.

We can install double or quad drives and check complete package	CALL
K10 Populated - Tested	250.00
WD1002-5 Hard Disk Controller	200.00
Host Interface Board	15.00

CABINETS

K2X	$35.00
K10	35.00

Bottom portion with front and back only.

CRT Bezel	12.00

MISC. CABLES

9" 50 Pin - 50 Pin Header	$2.50
14" 40 Pin - 40 Pin Header	2.25
9" Hard Disk - 20 Pin Header	2.00
14" Dual Inline 16 Pin Male (2)	2.00

FLOPPY DISK DRIVES

— ⅓ HEIGHT —

Okidata DSDD	$99.00

Supplied w/sub-panel to make ½ ht.

— ½ HEIGHT —

Mitsubishi M4851 DSDD 40 Tr	$139.00
Mitsubishi M4853 DSDD 80 Tr	139.00
Qume Trak 142 5¼" DSDD	99.00
Tandon TM 848-1 8" SSDD	149.00
Shugart 475 DS Quad 1.2Mb	159.00

Sweda Cash Registers
Models 303, 2640 & 2650
From major supermarket . . . $99 ea.

ERAC CO.
8280 Clairemont Mesa Blvd., Suite 117
San Diego, CA 92111 (619) 569-1864

HOURS: Mon. - Fri. 9 - 6 — Sat. 10 - 4

MINIMUM ORDER — $15.00

TERMS: VISA, MasterCard, Certified Checks, Money Order, NO COD. Visa and MasterCard add 3%. *Personal checks must clear BEFORE we ship.* Include shipping charges. California residents add 6% Sales Tax. For more info please call — *DON'T WRITE!* We don't have time to answer your letters.

IBM/PC COMPATIBLES

Mainboard, 8 Slot, Case, Power Supply . . . $225

To make this a complete system, add
A) Memory B) Floppy Controller
C) Drive D) Keyboard E) Video Card
F) Video Monitor G) Multifunction Card

A) **MEMORY**

256K 150 NS	$25
512K 150 NS	50
640K 150 NS	68

B) **FLOPPY DISK CONTROLLER**

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

C) **5¼" FLOPPY DISK DRIVES**

Mitsi 4851	$139
Qume	99
Okidata	99
Hi-Tek	99

D) **KEYBOARDS**

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

E) **VIDEO CARDS**

Tomcat with Parallel and Lightpen Port	$53
Hercules compatible Video Board	75
Color Graphics Adapter	69
Enhanced Graphics Adapter (EGA)	275

F) VIDEO MONITORS

Roland MB-122G, 12" (no case)	$39
Samsung MD-1254G, 12" Green	72
Samsung MD-1254G, 12" Amber	78
Goldstar MBM-2015, 12" Amber	106
Mitsi AT-1332A 13" RGB TTL	295
Goldstar MCH-4015 13" RGB 30 MHz	420
NEC Multisync 30 MHz 800*560 EGA	535

G) **MULTI FUNCTION CARD**

Parallel & Serial Port, Game Port Floppy Controller, Clock & Cal	$96

CPU/SUPPORT

MC68000-8 CPU	$9.99
C80186-3	27.00
C80186-6	9.00
Z80 CPU	1.00
Z80A CPU	2.50
Z80A PIO	2.50
Z80A SIO	5.00
Z80A DMA	10.00
8088	10.00
8089-3	29.00

EPROMS

2716	$3.50
2732	3.75
2764-25	3.50

PRINTERS

Centronics 703	$395.00
Centronics 588 - As Is	125.00
NEC2000 w/Single Sheet Feed	999.00

SYSTEM EXAMPLE #1
For the Hacker (Cheap)

Mainboard, Case, Power Supply	$225
256K Memory 150 NS	25
Floppy Controller (2 Drives)	36
Floppy Drive ½ Ht DSDD	99
Keyboard Cherry (no case)	38
Video Board with Parallel and Lightpen Port	53
Roland MB-122G, 12" Green Monitor (no case)	39
	$515

SYSTEM EXAMPLE #2
FCC Approved (Not Cheap)

Mainboard, Case, Power Supply	$225
640K Memory 150 NS	68
Multi I/O, Parallel, Serial, Floppy, Clock/Cal.	96
2 DSDD Floppy Drives (minimum)	198
Enhanced Graphic Adapter (EGA)	275
NEC Multisync Color Monitor (The Best)	535
AT Style Keyboard	69
	$1466
Oh, you wanted a turbo board	20
and a 20M Hard Drive & Controller	410

Now how much would you pay?
JUST $1896

HARD DISK DRIVES

10M Rodin RO-252	$230
10M ST-212	185
20M ST-225	385
19M ST-419	230
20M Half Height with Controller	410
32M Half Height with Controller	667
60M with Controller	1150
70M Vertex V170	725

HARD DISK CONTROLLERS

WD-1027 with Cable	$156
Omni-5510	150
Adaptec 2770A (Get 15M on 10M Drive)	156

POWER SUPPLIES

Elgar 400W Unint. Power Sup.	$375.00
+5V/1A, −5V/.2A, +12V/1A, −12V/.2A, −24V/.05A	15.00
+24V/2.2A	8.00

SWITCHERS

5V/9.5A, 12V/3.8A, −12V/.8A	$39.00
5V/3A, 12V/2A, −12V/.4A	29.00
5V/10A	25.00
5V/75A, +12V/8A, +12V/8A	55.00
5V/20A	35.00

MISCELLANEOUS

Headset/Boom Microphone	$3.95
Nicad Pack 12V/.5AH	6.50
5 Blade Muffin Fans	7.50
Joystick 4 Switches 1" Knob	5.50

RFI/EMI Protection For Your Computer

By Rex Buddenberg
3425 Elmwood Dr.
Alexandria VA 22303

There are lots of ways that computers can be noisy. In this case, Rex isn't talking about fans or hard drives — he's talking about turning on the system and destroying Cheers. Stifling electrical radiation is a popular black art right now because the FCC is cracking down on manufacturers whose products radiate noise. Here are some quick and cheap ideas that just might make your system less prone to disconnection during prime time.

In a recent class, one of my professors suggested ways to eliminate some of the radio frequency interference (RFI) and electromagnetic interference (EMI) between my computer and other electronic devices in the neighborhood.

The tips aren't guaranteed to make everything quiet, but like chicken soup, they can't hurt. These practices apparently aren't well known — I found about half my equipment wired incorrectly.

Some Theory

Just look inside your computer, terminal, printer, etc. There's a clock circuit in the system that puts a square wave pulse onto the clock line (a copper trace).

A true square wave is about the noisiest kind of wave known: those nearly vertical rising and falling edges are combinations of odd harmonics. A 1MHz square wave for instance, would contain 1, 3, 5, 7, 9, 11, 13, 15... MHz components. If the rise time for that 1MHz signal is 10ns, then the highest component would be at least 100MHz.

(Editor's note: The clock trace is only one of many lines that carry signals. One of my systems generates a hashy noise right on top of my favorite FM station. If I select a blinking cursor, the noise blinks on and off with the cursor.)

Tacking a terminating resistor onto a signal line sometimes reduces its radiation a bit. Four-layer circuit boards usually radiate less than two-layer boards, but the most common means of limiting interference is to enclose the board in a metal box. Unfortunately, the power leads usually compromise the box. The biggest culprit is the ground wire in the power cable.

If your equipment is wired like Figure 1, the ground wire acts as an antenna. It receives all those RF noises running around inside the box and transmits them outside. (Take a portable FM radio and pass it around the cord. What do you hear?)

Reverse Engineering?

Since the ground wire acts as a receiving antenna as well as a transmitting one, the reverse situation is also possible — external equipment can dump noise into your computer.

By rewiring the ground wire as in Figure 2, you preserve the electrical grounding characteristics (this is important to avoid smoke-tested hacker), but you interrupt the RF circuit. In other words, you're keeping the "innies" in and the "outies" out.

Make sure you use husky bolts on this job (3/8"), and don't try it on a plastic case!

If you have a socketed power lead, see Figures 3 and 4.

The same principles apply. Don't use the ground tab supplied on the EMI filter or appliance socket; instead, take the inside ground to the socket mounting screws. Make sure the socket is mounted from the "outside in" and that it makes clean electrical contact with the cabinet wall.

One More Thing

You can also quiet things down by adding a strong external bond from

Figure 1 - The Wrong Way To Wire Your Equipment

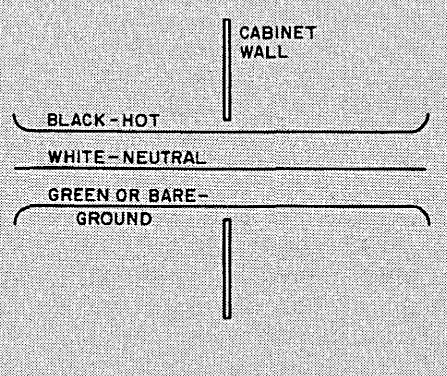
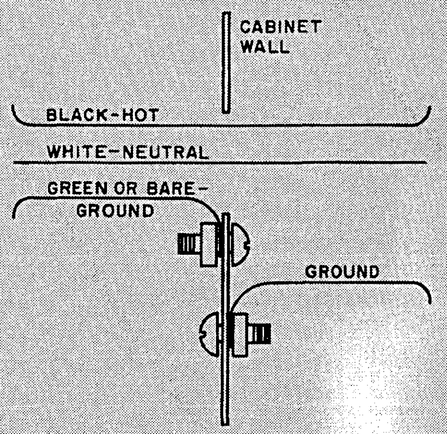


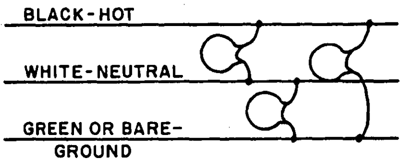
Figure 2 - . . . And The Right Way



your equipment cabinet to a solid ground such as a water pipe.

And An Afterthought

While you're inside the cabinet and monkeying around with the wiring, install three metal oxide varistors (MOVs) across the power leads as shown below. (These little guys conduct whenever the voltage across their leads exceeds their rating — usually 130V.)



My batch of MOVs say "V130LA20A" (130 volt, 20 amp). 10A sizes are adequate for most computers and environments. The MOVs should be around $1 each and are available at most parts houses (including many Radio Shacks).

■ ■ ■

Figure 3 - The Right Way To Install An EMI Filter

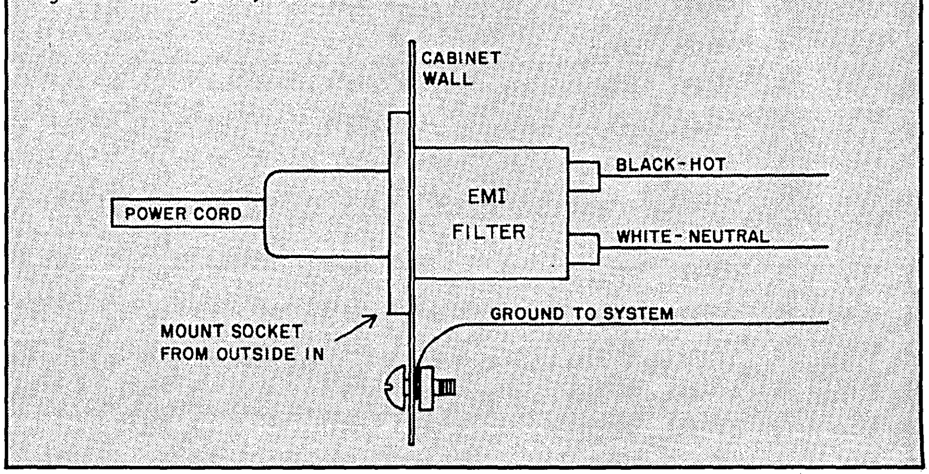
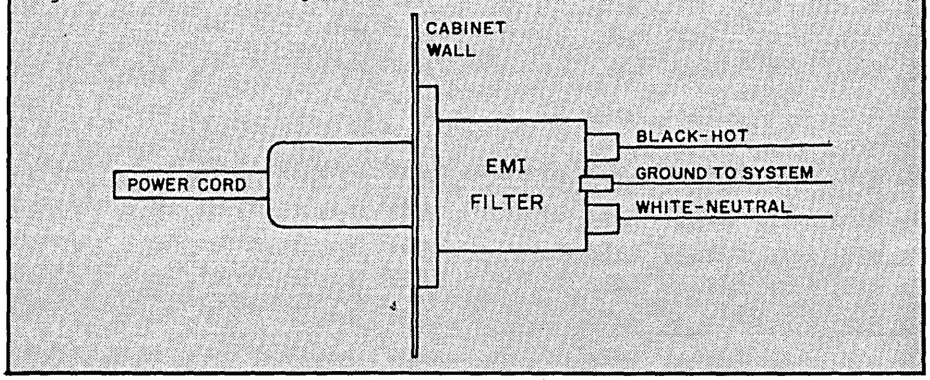


Figure 4 - . . . And The Wrong Way



The SOG Report: Exciting New Systems

By David Thompson

It was Saturday evening. I had just finished dinner when I glanced up at my companions, all 256 of them (the fire marshall's limit). They had the faces of marathoners — deep lines, sunken eyes, heads propped on hands. But they and I shared the exhilaration of a task well chosen and well done. We had run another SOG.

SOG V included: 29 formal talks, a half dozen informal sessions, recreation (rafting, horseback riding, chair lifts to a mountain top), babysitting, meals, and products from all over the world. But that was the normal part, the easy part. The marathon began after supper and continued into the wee hours, sometimes until breakfast (7 a.m.).

Reminiscing

At the first SOG we'd had none of the formal talks, displays, babysitting, or recreation, yet the SOG I faces had shown the same excitement, the same feeling of being somewhere special, of helping create something that might someday make a difference to a lot of people.

At SOG I the common interest was the Big Board I, the first compact, low power, single-board Z80 system on the market.

It was cheap ($650 for an empty board and a bag of parts), and it was small enough to fit on the side of a disk drive (OK, OK, so it was an 8" drive). It supported the most popular operating system, CP/M.

At SOGs II, III, and IV, the scope expanded to include Big Board II, Xerox 820, and Kaypro, all Z80, CP/M machines. Then came Slicer, Deficon, and finally the clones.

This year, we cussed and discussed nearly every processor from the 4004 to the Transputer. (George Morrow had the audacity to suggest that the 8008 was the first real processor, so we discussed the 4004.) We talked languages and operating systems, even Unix.

Most of the informal discussions this year centered around ways of building up systems around the latest, greatest technology (and where to find operating system support).

It's curious that we were having those same discussions at SOG I.

Many of the Big Board folks have since built clones, but they complain that the feeling isn't the same. Fifteen

This is the SOG that was. We hope to see you next year.

1986 SOG SPEAKERS SCHEDULE Central Oregon Community College - Bend, Oregon

		8-9 am	9-10 am	10-11 am	11-12 am	12-1 pm	1-2 pm	2-3 pm	3-5 pm
FRIDAY JULY 25th	Ochoco Hall Room 104	**Jack Dennon** *The V20, V30 & CP/M Emulation*		**Mike Weisert** *Modula Graphics Tools*	**Michael Frieling** *Developing Tools For Expert Systems*	LUNCH		**Dean Klein** *The V Series & The 80386: Dual Instruction Set CPUs*	In Pence 28: *SOFTWARE FORUM* **Dave Rand** **Michael Frieling** **Earl Hinrichs** **George Scolaro**
	Pence Hall Room 28		**David Thompson** *An Inside Look At Micro C*		**Trevor Marshall** *Designer's View Of The 68020*		**Dave Rand** *In The Public Domain*		
	Ochoco Hall Room 231	**Don Latham** *Goal-Oriented Program Building*	**George Watts** *Pitfalls of Small Business Ventures*	**Joseph Bartel** *68000 Multi-User System: Alternative To LAN*			**Andy Bakkers** *The Transputer: Life After The Microprocessor*		4-6 pm In the Pinckney Center: **Allyn Franklin's** *Drive Workshop* **McTek** *PC Workshop*
	Ochoco Hall Room 5		**Larry Fogg** *Intro to Turbo Pascal: Novice Programming*	**George Scolaro** *32-Bit UNIX On A Shoestring (New Life for Z80s)*				**Jim Tanaka** *Artificial Intelligence & Human Cognition*	
SATURDAY JULY 26th	Ochoco Hall Room 104	**Earl Hinrichs** *MS-DOS Device Drivers*	**Laine Stump** *International Computing*		**Don Fletcher** *Intro To 8088 Assembly Language Programming*	LUNCH	In the Pinckney Center: **George Morrow** *New Directions In Computing*		3-5 pm In Pence 28: *HARDWARE FORUM* **Joseph Bartel** **Don Fletcher** **Trevor Marshall** **Dean Klein**
	Pence Hall Room 28		**Doug Anderson** *Poor Man's Networking*	**Peter Casey** *Unique Features Of C*	**Chris Jones** *CAD (and more) On The 68020*				
	Ochoco Hall Room 231	**John Kirk** *Math & Graphics For Geometric Models*		**John Kirk** *Source Code Translation Between Languages*					4-6 pm In the Pinckney Center: **Allyn Franklin's** *Drive Workshop* **McTek** *PC Workshop*
	Ochoco Hall Room 5		**Sigi Kluger** *Programming The 68020*		**Chris Brock** *Hardware & Software Considerations For A Multi-User System*				

minutes worth of plugging boards together is different than gluing chips on a board. Also, MS-DOS supports serial and parallel interfaces, and so many utilities come with it that there's a lot less reason to hack. (See Don Fletcher's MS-DOS driver article in this issue if you're still interested in hacking.)

PD32

Thanks to SOG, we've got two spectacular projects coming up for you. The first, the Public Domain Project — 32000 (PD32), is in this issue.

The PD32 is a 32016 system designed by Dave Rand and George Scolaro. They came up with the system after National started selling a 32016 starter kit for a paltry $69 (6MHz). The kit includes the processor, memory management unit, clock, floating point processor (the same one Definicon is using), and an interrupt handler. But that's just the beginning. They also include a pair of monitor ROMs, serial transfer software, books, instructions, etc., etc., etc.

Kits are available from any parts house that handles National. Jameco Electronics is selling the 32016 Designer Kit for $59.95 (no doubt the 6MHz chips). The 10 Meg parts should be about $120 a set (but the 6MHz parts will often run 8 or 9MHz).

Jameco Electronics
1355 Shoreway Rd
Belmont CA 94002
(415) 592-8097

The PD32 is a public domain system because Dave and George are releasing the board, artwork, circuit design, and software drivers to the public.

It's also public domain because a number of SOG attendees got together to form a users group to promote the system and produce hard-to-get items. The group will be making boards, burning PALs and ROMs, and making them available at cost. Micro C will be the information center.

Dan Efron volunteered to head the distribution of boards and parts for the system (while attending college). You can contact him via his parents' address (he says it'll be a day delay at best).

Dan Efron
8910 Westmoreland Lane
Minneapolis MN 55426
(503) 382-7643 (Micro C RBBS)

Dan says he will be a regular participant on the Micro C bulletin board so you can reach him that way. Be sure to check out the PD32 area on the board.

I talked to Dan three weeks after SOG and he told me that Ken Berry will be helping in the project. Ken will be putting together kits of peripheral parts (including RAM sticks). Dan says he's getting some very good quotes from board houses.

Meanwhile, George Scolaro said he was expecting to get the test run of boards back from the prototype shop by early September. If there weren't too many cuts and jumpers, he's probably already shipped those boards to the dozen folks who were on the list. When the artwork is finalized, George'll be sending it along to Dan so the production run can get started. They're hoping to have shippable boards and kits a short time after you read this.

Host System

The PD32 will take some ideas from the Definicon board. All I/O will come and go through a host system.

"Aha!" you say. "Another PC leach."

Well, yes and no. Yes, it will plug quite happily into a clone. However, it will also run on a Z80 CP/M machine. They've written software interfaces (in C) for both types of machines.

Plus, there is full Unix Version V for the system at only $500. (Only $500 for an operating system?! Has this guy been hanging out with Digital Research?)

However, the price includes compilers, editors, 31 disks worth of utilities — and all multi-user. You could put together a complete Unix system with 2 Meg of RAM for under $1,000, including the operating system. (You'll need at least 10 Meg of winnie on the base system to run Unix properly.)

Even considering the cost of the host system, this has to be the cheapest way known to man or hacker to get into a full-blown, multi-user Unix system.

On The Other Hand

Maybe you don't want a full-blown Unix system and just maybe you don't have a winnie on your Z80 system. (Unix alone requires 10 Meg.)

"Aha, I've finally got you," you say. "This won't plug into my stock Big Board and single handedly support all the programmers on Michigan's upper peninsula. I've got to add a winnie."

Neil Koozer attended SOG V and also got excited about the PD32. He's been working with National chips for some time now. He dropped me a letter a couple of weeks after SOG:

"When I started coding for the 32000 I thought I had died and gone to heaven. If you do assembly coding for an 8-bit machine or for one of those segmented things, you know that you spend most of your energy working around obstacles. It took me five days to write a disassembler for the 32000.

"I have no interest in Unix or in a hard disk; all I want is an assembler and enough of an operating system to run the assembler. (So he has a basic foundation to build on.)

(continued next page)

(continued from page 39)

"In the interest of promoting a 32000 users group, I will be donating all the 32000 stuff that I write. I'm in the middle of writing a high/low level language and a debug monitor."

Neil Koozer
Kellogg Star Route, Box 125
Oakland OR 97462
(503) 459-3709

Talk about a group effort. If you want to get involved in the PD32 project, get in touch with Dan or Neil.

68000

The 68000 system which follows is a different kind of project. It isn't as close to reality as the 32000 board because it saw first light at the SOG, but it has lots of potential.

This is the generic 68000 system, and its claim to fame is that Hawthorne Technology will be providing a complete, single-user operating system for $50 per copy.

(There's also a second 68000 project that's in the "waiting for more information" stage, but we'll deal with that as Andy Bakkers gets more details from Germany.)

The Generic 68000

Motorola fans also showed up in force at SOG V, and, as usual, discussions centered on the lack of a cheap, standard operating system for the processor. Sure, there are proprietary systems for the Macintosh, Amiga, and ST. There are also expensive ones like CP/M 68K and OS-9.

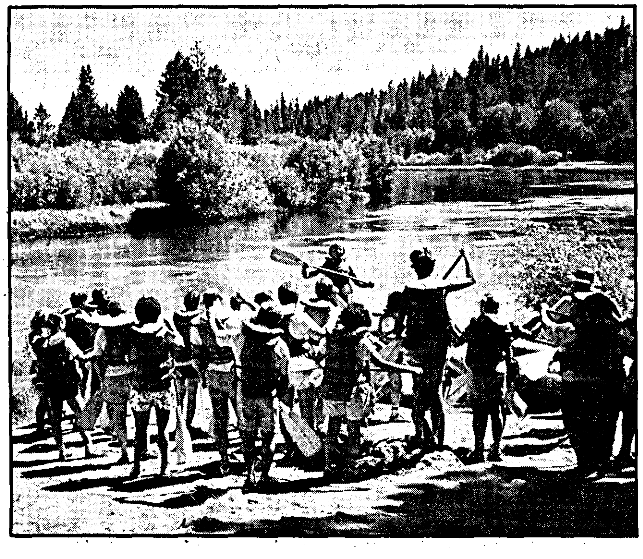
In fact, there isn't much incentive to design new 68000 hardware because the operating systems are so expensive. Just the operating system, two compilers, and a few utilities total over $1,000 in OS-9/68000. And that's what each user pays. System designers also have to fork over significant sheckles to get the source of the BIOS.

Well, at the SOG we discussed the 68000's operating system problem, cussed it, and then changed it. Maybe permanently.

Joe Bartel of Hawthorne Technology offered to sell a single user version of his operating system for $50 per copy. It would include source, have built -in hooks so you could add drivers for nearly anything, an assembler, and a simple mid-level language of his own design. (It's best described as FORTH with added IF/THEN/ELSE type structures.) The operating system supports MS-DOS disk formats.

If that project comes together in time (hardware, software, assembler...) we'll have a full-blown 68000 issue around the first (or second) of the year. I'm guessing it'll happen.

Allyn Franklin teaches disk drive alignment and repair to a sitting room only crowd. In the background, vendors display their wares. A PC workshop and a hardware forum were running concurrently with this workshop.



Guide demonstrates specialized equipment (paddle) as rafters prepare to drift off on the annual Tour de Splashe. The placid Deschutes takes a dive about half a mile downstream.

Of course, one of the projects would be porting the operating system over to a number of 68000 computers. Anyone already running a 68000 or thinking of designing a new hardware package (hooray, hooray) should talk to Joe.

Joe Bartel
8836 SE Stark
Portland OR 97216
(503) 254-2005

Joe mentioned a number of options for the hardware:

1. Plug-in board for the PC
2. 4 Meg 68000 replacement board for the Kaypro
3. Same size board as the motherboard in an XT, so it could use cheap cabinets, power supplies, etc.

Joe also noted that there's a Pascal compiler in the PC-SIG library, written in Pascal, and he's writing a database handler.

Educational 68000

In his talk this year, George Morrow mentioned that there's still that great divide — hardware folks do hardware, software folks do software — and it's the in between places where low level software and hardware meet that most folks don't understand.

How do you create a system from scratch, from just a pile of parts connected together? Where does the system monitor come from? Who creates the first assembler? What does the assembler run on if there's no operating system, and how do you write an operating system without an assembler?

Of course, if you just want to build the hardware and purchase an operating system complete with basic (not BASIC) tools rather than tangle with a Chinese computer puzzle, you'll be able to do that too.

Andy Bakkers came by the office before heading back to Holland. He suggested that folks start at absolute zero with the 68000. What he meant was that the user should build the system (wirewrap or PC board), create a simple monitor, and then bootstrap the system.

Bootstrapping would mean creating an assembler and then a high level language. Other languages and tools could then be written either in assembly or the high level language.

He suggested that the assembler and some simple debugging tools be included in the monitor. Then, once the monitor was finished, builders could move on to C or Pascal.

He mentioned that a company in Germany is selling a 68000 board (and parts) along with such a monitor. (The package is aimed at educational use.) The assembler built into the monitor is complete enough that users can construct a complete compiler, working with Wirth's Pascal book.

Andy will be gathering some additional information about this system for us.

The Rest Of SOG

We'll include excerpts from many of the talks in Micro C over the next year, so you won't miss out entirely.

We started the schedule with the idea that there would be only two things happening at a time (and folks could remain in the room for the following hour if they wanted to continue the discussion). We weren't entirely successful. Oh well, maybe next year we'll try for three at a time, or expand the formal talks into Sunday, or...

The all-day white water rafters (60 people) wanted a two or three day raft trip next year. We'll see about setting up one of those.

The last dorm room was assigned on May 1 (almost three full months before SOG). So if you have a hankering to stay in the dorm next year, plan to get your reservation in early. Fortunately, there are lots of motels within a couple of miles, and there's plenty of parking for campers and trailers next to the dorm.

■ ■ ■

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

DIGITAL RESEARCH COMPUTERS (214) 225-2309

CAF 8 MHZ AT COMPATIBLE - $1399

★ PRODUCED BY CAF: A $500 MILLION/YEAR TAIWANESE CONGLOMERATE.
★ BURNED IN FOR 48 HRS. ★ INCLUDES DOS 3.1 AND MANUALS.
★ 512K RAM, EXPANDABLE TO 1MEG ON M.B. ★ 1.2 MEG FLOPPY INCLUDED.
★ 80286 WITH OPTIONAL 80287 ★ ON BOARD REAL TIME CLOCK.
★ AT STYLE KEYBOARD WITH ENLARGED ENTER AND SHIFT.
★ 195W POWER SUPPLY ★ COMPLETE SERVICE MANUAL AVAILABLE: $40.
★ COMPLETE SYSTEM DOCUMENTATION INCLUDED, PLUS DOS MANUALS.
★ FLOPPY AND HARD DISK CONTROLLER CARD INCLUDED.
★ ERSO BIOS FOR MAXIMUM AT SOFTWARE COMPATIBILITY.
★ LIMITED 1-YEAR FACTORY WARRANTY. ★ FCC CERTIFIED!
★ 1.2 MEG FLOPPY CAN READ AND WRITE 360K DISKETTES. ADD $15 UPS.

CMI 6640 33 MB. HARD DISK
39 MS - AT COMPATIBLE.
VERY LIMITED STOCK — $325

MEX-PC MODEM SOFTWARE PACKAGE $69.95

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

MEX-PACK LIST IS $99.95 — **SPECIAL $69.95**

STANFORD TURBO XT - $399

4.77 OR 8 MHZ ★ FCC CERTIFIED ★ 640K MOTHER BOARD (O K) ★ .135 WATT P.S. ★ XT-AT 5160 KEYBOARD ★ FLIP TOP CASE ★ 8 EXPANSION SLOTS ★ ASSEMBLED AND TESTED ★ WITH BIOS ★ FLOPPY CONTROLLER CARD & CABLES ★ WITH DOCUMENTATION. ADD $12 UPS.

A.R.C. XT TURBO - $529

4.77 OR 7.4 MHZ ★ 8 EXPANSION SLOTS ★ SLIDE OUT PC CASE ★ 5160 XT-AT KEYBOARD ★ MS DOS 3.1 + BASIC DISKETTE ★ 256K RAM, EXPANDABLE TO 640K ★ 4 LAYER JAPANESE MOTHER BOARD ★ FULLY TESTED AND BURNED IN. ★ SUPERIOR, CLEAN BIOS. ★ 135 WATT P.S. ★ SUPERB DOCUMENTATION ★ FCC CERTIFIED. ADD $15 UPS.

XT OPTIONS:

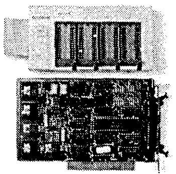
★ V20.8 MHZ NEC PROCESSOR. REPLACES 8088. - $12.95
★ TWO 1/2 HT. D.S. DD. FLOPPIES INSTALLED: $175/Pair
★ 256K DRAM. 150 NS. INSTALLED AND TESTED: $28/9
★ 64K DRAM. 150 NS. INSTALLED AND TESTED: $10/9
★ HARD DISK: ST225 WITH CONTROLLER CABLES & FORMATTED: $425
★ ZUCKER COLOR CARD WITH PARALLEL PRINTER PORT: $86
★ MONOCHROME/GRAPHICS VIDEO CARD WITH PRINTER PORT: $89
★ PLEASE CALL FOR MONITOR PRICING

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

SCHEMA: Schematic Capture and Drawing Package

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

NEW!
PC/XT EPROM PROGRAMMER
$199

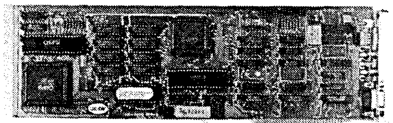


★ LATEST DESIGN ★ PROGRAMS UP TO 4 DEVICES AT ONE TIME ★ FEATURES EASY TO USE MENU DRIVEN SOFTWARE THAT RUNS UNDER PC OR MS-DOS. ★ USES AN INTELLIGENT PROGRAMMING ALGORITHM FOR SUPER FAST (8X) EPROM BURNING. ★ THIS PLUG-IN BOARD ATTACHES TO AN EXTERNAL MINI CHASSIS CONTAINING 4 TEXTOOL Z.I.F. SOCKETS. ★ NO PERSONALITY MODULES REQUIRED ★ AUTOMATIC VPP SELECTION: 12.5V, 21V, OR 25V. ★ EPROM DATA CAN ALSO BE LOADED FROM OR SAVED TO A DISKETTE. ★ PROGRAMMING SOFTWARE SUPPORTS: 2716, 2732, 2732A, 2764, 2764A, 27128, 27128A, 27256, 27256A, 27512, AND 27512A. ★ ASSEMBLED AND TESTED, BURNED. IN WITH MANUAL. $199 WITH SOFTWARE.

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

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

PC/XT AT
ENHANCED GRAPHICS ADAPTER

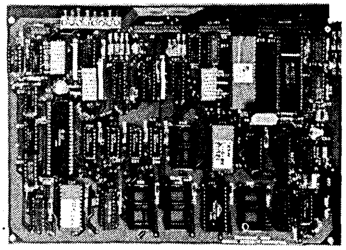


$279

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

ZRT-80 CRT TERMINAL BOARD!

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



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

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

$\$89^{95}$ #ZRT-80

A&T ADD $50

(COMPLETE KIT, 2K VIDEO RAM)

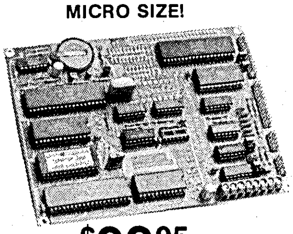
BLANK PCB WITH 2716
CHAR. ROM. 2732 MON. ROM
$\$49^{95}$
SOURCE DISKETTE - ADD $10
SET OF 2 CRYSTALS - ADD $7.50

THE NEW 65/9028 VT
ANSI VIDEO TERMINAL BOARD!
★ FROM LINGER ENTERPRISES ★

A second generation, low cost, high performance, mini sized, single board for making your own RS232 Video Terminal. Use as a computer console or with a MODEM for hook up to any of the telephone-line computer services.

FEATURES:
★ Uses the new SMC 9028 Video Controller Chip coupled with a 6502A CPU.
★ RS-232 at 16 Baud Rates from 50 to 19,200
★ On board printer port!
★ 24 X 80 format (50/60 Hz).
★ For 15,750 Hz (Horiz.) monitors.
★ 3 Terminal Modes: H-19, ADM3A, and ANSI X 3.64-1979
★ Wide and thin-line graphics.
★ White characters on black background or reversed.
★ Character Attributes: De-Inten, Inverse, Underline and Blank.
★ Low Power: 5VDC @ .7A, ± 12VDC @ 20MA.
★ Mini size: 6.5 X 5 inches.
★ Composite or split video.
★ 5 X 8 Dot Matrix characters (U/L case) with descenders.
★ Answer back capability.
★ Battery backed up status memory.
★ For ASCII parallel keyboard.

MICRO SIZE!



$\$99^{95}$ (Full Kit)

ADD $40 FOR A&T

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

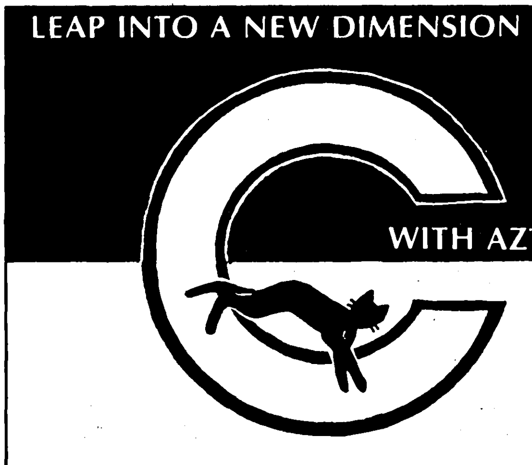
NEW OPTION
PC/XT keyboard ROM. Allows use of IBM style keyboard. $15

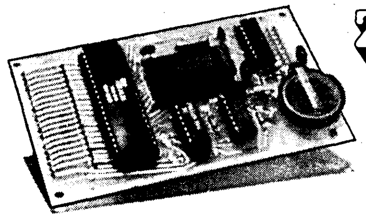
Digital Research Computers

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

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

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





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

Laine Heads Home After The SOG, And Makes It

In his own one-track way, Laine talks about Yugoslavia, militia, Taiwan, clones, floppy driver boards, SOG V, the X-16, the Slicer, and environments in MS-DOS. (Actually, he really covers more, I just forgot what it was.) This column is obviously just another Micro C attempt at blatant sensationalism.

We're sitting in central Yugoslavia waiting for a police roadblock to filter back to us. We would have been down the highway to Belgrade long ago, but our bus driver was caught passing the line of cars now ahead of us and the militia confiscated his passport.

We had to turn around and drive the 10 kilometers back to the militia post. The other lane was solid cars the entire way. And only one militia car with three officers to sift through them.

The line extended an hour beyond the militia station when the officers realized the fruitlessness of their situation and with one wave of the hand, overheated cars, sweating bodies, and screaming beasties were dispersed into the Yugoslavian afternoon.

The militiamen have now returned to their post and are arguing with our bus driver over the merits of his driving style, politics, and most likely, religion. It's cooler now, sitting on the grass next to the militia house, waiting for our driver to come back. He'll no doubt tell us that the bus has been confiscated along with everything and everyone on board and we'll all be relocated to the Adriatic Coast where we will be pressed into service as latrine technicians and goat herders.

The bus will probably be used to decoy other Turkish busses across the border. Our belongings, including my little Toshiba portable and the ribbon re-inkers I brought to sneak across the border into Turkey, will be donated to the People's Work Training Center where they will be used for practice in the Advanced Sledge Pounding Class.

And How Was YOUR Trip Home From SOG ???

Later: Well, we ended up waiting at the militia post for about two and a half hours before we could go, and by then the line of cars was backed up again. (We lost six hours in all.)

SOG V

Yes, I did make it to SOG. Had a GREAT time, too. Since everyone else will be talking about it this issue anyway, I'll just take a quick, surreal dip through the highlights:

Grey boat slamming into white waves - Tom into the water by accident - water fights - Tom into the water on purpose - volleyball in a dust storm with the gang - "Watch how fast this scrolls!" - "Hey, come over and look at my 68000." - bi-window visual translation - "But ours works with a mouse." - Clone Wars - "Why don't you give me a copy of that?" - "Where's the beer in this place, anyway?" - dinner at The Pine Tavern - "So I'm taking back two rocks from Mount Bachelor for the kids' presents." - "We'll get some chips and be right over." - "When you hit the print screen key, you generate a keyboard hardware interrupt which generates a print screen interrupt which is intercepted by the GRAPHICS program which generates a video interrupt which is intercepted by the EGA card and then you generate a printer interrupt which is intercepted by the MODE program which generates a serial COM interrupt, which ..." - "Let me know how your project goes." - "Has anyone seen my beer cooler?"

A brief list of my favorites at SOG: PC-Tech's X16 and 4 Meg "Above-Board" card (for 850 bucks!), Logitech Modula, Shawn's Megabyte Slicer, new (finally!) monitor ROMs from Slicer, and EXPRESS 2.2 in flaming, instantaneous Hawaiian color on Dean's EGA card on the X16.

Update On Taiwan

Thirty-five PCs arrived from Taipei while I was away, along with 35 3" disk drives to mount in them. Nothing is really together yet, and already I have problems.

Flakey Flakey Flakey

The motherboards we got are 8MHz turbo boards from Auto Computer (doesn't really matter, you'd get the same board no matter where you ordered from). Unlike most Taiwan junk, they actually WILL run at 8MHz. Almost. All is well until there's a disk error and then, instead of printing out a nice error message, the machines just lock up and die. Totally.

This isn't the only time they die, either. At random times during the day they simply decide to go home early and don't even have the decency to mark their names "OUT" on the office bulletin board. So I guess we just use them at 4.77 until I discover the culprit or find some 20MHz crystals (6.67MHz).

And Still More Flakes

We actually had fairly decent luck with DOAs this time — only one bad power supply, one video card (caused by the power supply), one winchester faceplate fallen off, and one color monitor power supply shish kebabed. Of course, that's only if you don't count the stripped screw holes, wrong sized nuts, poorly mounted drives, and the floppy controller cards.

Loss Of Control

Three months ago, when I connected a 3" drive to one of the sample Taiwan units to make sure they were compatible, everything worked fine. The new machines and drives work fine together too — until you try writing to the disk. Then your entire disk gets trashed!

When I replaced the new controller with an earlier version I had no problem. The two cards used nearly the same parts; the new one just had a different layout to conserve space. Then I noticed that someone had cut costs by removing all but four of the "fingers" for ground connections on the card edge connector which hooks up to the data cable! The pinout is cleverly designed so that every other wire is ground, helping to reduce crosstalk between channels. But a ground which is connected only at one end does absolutely NO GOOD. Oh, Ataturk! What have our neighbors done now?!?

Not so fast, though. I had seen these symptoms before when I installed a single 5″ drive with no terminating resistors, so I checked the OEM manual. On this drive the termination is built in; all the drives have terminator packs. Each signal has 1K ohm of pullup (providing about 5 milliamperes of pull-up current). They say it's enough. It should be. Every other machine I have, including the X16 and my Toshiba portable AND my Slicer, can handle the little buggers just fine.

Well, perhaps a grounding problem can cause the same symptoms. I tried tying all the grounds on the floppy cable together, but it didn't help. Must be the termination after all.

All right, let's try it with a terminated 5″ drive together with the 3″. AOK. How about with two of the little guys? Sure thing.

Now I know. These $%@%%!*! floppy cards don't have as much "drive" as the old card, and they need 500 ohms of termination on the cable. Wonderful! Where am I going to put it? The built-in termination on the drive is in a cute little SIP resistor network. Cute little SIP resistor networks aren't available in Turkey. Looks like I'll have to solder eight tiny 1K ohm resistors onto 35 floppy disk drives. That makes 560 solder connections! Why can life never be simple?

Postscript: I have decided instead to have a "terminator plug" manufactured to be plugged into the unused cable plug. I also rechecked the parts on both controllers and discovered the sole difference is that the old controller uses 7438 NAND gates to drive the outputs (as suggested in the drive manual) while the new controller uses 74LS38s. Is this enough difference to cause what is happening? Somebody clue me in....

Editor's note: Low power Schottky parts (74LS38s in this case) have relatively weak outputs (which isn't a problem as long as they are talking to other LS parts). Standard TTL parts (7438s) generally have much stronger outputs and that's obviously important in this application.

The Lesson

The great deals found in Taiwan are not all peaches and cream (or strawberries and sugar). You never know exactly what you'll get. In fact, you can't even guarantee that the screws holding your winchester drive in the cabinet will be the right size. Most of the time they aren't. Either they're too big and the threads are stripped or they're too small and held in with

(continued next page)

(continued from page 45)

fingernail polish. I no longer have ANY respect or trust AT ALL for Taiwanese computer manufacturers. At least none of them we've dealt with so far (and they were the best out of the 30 or so we met with when we went to Taipei).

Oh, yeah. Except Multitech. But they couldn't sell to us anyway.

By the way, the 3" drives we purchased are Panasonic (Matsushita, National) "super thin" 363s, and they are wonderful little units indeed. Small, light, low power, quiet. I only wish we had such good luck at the other end of the cable.

On The Pleasant Side

We didn't buy the monitors for our clones from Taiwan. We bought them from a local junk dealer who had yanked them out of an old IBM mainframe before he melted it down for scrap (what a lovely sight that must have been). Of course we tried one before we bought, and it worked.

The price was right (about $15 each) but they all had 110 volt transformers and no cabinets. Ergun (my companion on the Taiwan trip) is handling the reconditioning. He's getting the cabinets made by a friend who usually builds speaker cabinets. The transformers will be hefty 220 volt affairs custom made by a local manufacturer (oh, the strange and wonderful things that can be had in this country).

In the meantime I've sacrificed the monitor from my Slicer so Ron can use it on the X16 over at the main office. This has forced me to hook my Slicer up to a serial terminal (a Wyse 50) for the first time in several months. I can't BELIEVE how fast it is! On the clones you can actually see the cursor moving across the screen; on the Slicer/Wyse you can barely even see the screen scroll! Typing the "next page" key in the editor brings up the entire next page long before I can even start to say "one thousand one" (somewhat faster than memory mapping on a clone with a color card, although slightly slower than a monochrome card). I'm talking running MS-DOS and going all the way through the operating system for each and every character here!

Clones may be trendy right now, but for sheer word-chopping speed I still couldn't do without my Slicer. Especially since somebody took away all my 40-track drives, and the Slicer is the only system I have which knows how to double-track a 40-track MS-DOS format on an 80-track drive.

Environmental Awareness

Okay. Enough chit-chat. On to constructive things. You'll remember that in the last episode of "86 World" we learned how to call one program from within another program by using the DOS "Exec" function call. It's a useful tool to have tucked off in your \turbo\lib and \asm\lib directories somewhere.

Very useful, but limited in an inconvenient way. The problem with the Exec procedure given last issue (for Turbo Pascal, Assembly Language, and Microsoft BasCom) is that it assumes COMMAND.COM (the command processor) is in the root directory of the currently selected drive.

However, this isn't always the case. If you (or the person you sold your slick new text editor to) happen to boot from drive C: and be working on a "data only" disk on drive A: and then try to execute an external command, then what happens?

Exec will croak (not even ribbet or sneeze), that's what. The Exec function call doesn't know a thing about PATH or any of that good stuff.

Enter COMSPEC

Fortunately somebody at Microsoft foresaw this problem and put a variable in the MS-DOS "Environment" called COMSPEC. COMSPEC is set at boot time to the proper drive/path/filename to find COMMAND.COM. All we have to do is call the program described by COMSPEC instead of just COMMAND.COM. Simple!

Figure 1 - GetEnvironment For Pascal

```
CONST string128 = string[128];


{--------------------- GetEnvironment ---------------------}
{    return the value of the Environment Variable 'EnvVar'    }
{----------------------------------------------------------}
FUNCTION GetEnvironment (EnvVar : String128) : String128;


CONST
    NULL = ^@;


VAR
    EnvPtr : ^CHAR;
    EnvSeg : INTEGER ABSOLUTE CSEG:$2C;
    EnvOfs : INTEGER;
    Found  : BOOLEAN;
    TmpStr, MatchStr : String128;


begin
TmpStr := '';
EnvOfs := 0;
EnvPtr := Ptr(EnvSeg, EnvOfs);
Found  := FALSE;
WHILE (EnvPtr^ <> NULL) AND (NOT Found) DO

    { check for match of current variable name }
    begin
    MatchStr := '';
    WHILE (EnvPtr^ <> NULL) DO
        { extract an environment variable name and its value }
        begin
        MatchStr := MatchStr + EnvPtr^;
        EnvOfs := EnvOfs+1;
        EnvPtr := Ptr(EnvSeg, EnvOfs);
        end;
    IF (EnvVar = Copy (MatchStr, 1, POS('=',MatchStr)-1)) THEN
        { Found the variable, get its value to return }
        begin
        TmpStr := COPY(MatchStr, POS('=',MatchStr)+1, Length(MatchStr));
        Found := TRUE;
        end  {  if variable found }
    ELSE
        { Not Found, move past NULL to next variable }
        begin
        EnvOfs := EnvOfs+1;
        EnvPtr := Ptr(EnvSeg,EnvOfs);
        end;  { variable not found }
    end;  { while not found and not end of environment }


GetEnvironment := TmpStr;   { transfer result into return variable }
end;  { GetEnvironment }
```

WHOA! Hold It!

Slow down here! First of all, what is the "Environment" anyway? And how do we find this "COMSPEC"? Good questions.

Environment

First, the Environment is a collection of strings of the form "NAME=string" stored somewhere down in the depths of low memory. You can view and change them from the user level with the SET command (try it now — just type SET at the A> prompt and see what it shows).

From a program, the environment can be scoped out by following the clue of the "Environment Segment" pointer stored at location 2Ch ($2C for all you Turbo-ers) in the Program Segment Prefix (simply the bottom of the code segment for .COM programs). The environment variables are stored starting at location 0 in the segment indicated by "Environment Segment." They consist of a series of C-style (0 terminated) strings followed by a final 0. In Assembly Language DB statements the environment would look something like this:

```
DB      'M2LIB=C:\M2\LIB',0
DB      'PATH=C:\;C:\BIN;\',0
DB      'CLUB=Sierra',0
DB      'NEIGHBORHOOD=Watts'
        .
        .
DB      0
```

So, how do we find COMSPEC? Just scan through the environment looking for the word COMSPEC and then use everything from the = to the next 0 as the filename of our command processor. But let's not hardcode this thing. Let's be really neat and make a general purpose function called "GetEnvironment" that will return the value of ANY environment variable.

GetEnvironment In Pascal

Armed with only the above information, my roommate Ron set out to do exactly that, and in a few hours (probably less, but I wasn't there so I don't know) he came up with the routine in Figure 1 for Turbo Pascal.

(continued next page)

(continued from page 47)

There really isn't much to explain about it. Just notice how he declared the EnvSeg variable to be at "CSeg:$2C" with the absolute directive. CSeg is a standard function in the MS-DOS version of Turbo Pascal.

CSeg returns the address of the beginning of the Code Segment. The "$2C" after the colon is the offset, i.e., the exact location within the Code Segment of the environment.

Editor's note: For more on Code Segments and offsets, see Don Fletcher's "Writing A Printer Driver For MS-DOS" in this issue of Micro C.

After that he just put the value at EnvSeg into the segment of a pointer variable (with the PTR procedure), and from then on it was turned into simple character crunching.

Since we started talking about this in terms of COMSPEC and Exec, let's make the patch to Exec (see Micro C, issue #31, page 41) so that it uses GetEnvironment. First Copy the routine GetEnvironment into the beginning of the same INCLUDE file as Exec. Then replace the line:

```
ComFile := '\command.com'+chr(0);
```

with:

```
ComFile := GetEnvironment('COMSPEC')
        +chr(0);
```

Now your programs will always be able to find COMMAND — as long as there is a system disk in the drive you booted. (Some feats are beyond the reach of mere specks of dust like us.)

Figure 2 - GetEnvironment For Assembly Language

```
;
;       GetEnvironment- return a pointer to the value of a string
;                       in the MSDOS "Environment"
;
;       ENTRY: DS:SI --> a string (in Pascal Format) containing var. name
;
;       EXIT:  IF (NC) ES:DI --> the value of said variable
;              IF (C) nothing (not found)

ENVSEG  equ     word ptr cs:2Ch ;ptr to environment in base page

GetEnvironment  PROC    Near
        PUSH    AX
        PUSH    CX
        PUSH    SI

        MOV     ES,ENVSEG
        XOR     DI,DI
        XOR     AX,AX
        CMP     ES:[DI],AL      ;see if no environment at all
        JZ      GetEnv9
GetEnv1:
        POP     SI              ;retrieve string address
        PUSH    SI
        MOV     CL,[SI]
        XOR     CH,CH
        INC     SI
        REPE    CMPSB           ;try to match
        JNZ     GetEnv2

        CMP     ES:byte ptr [DI],'='
        JZ      GetEnv3         ;FOUND
GetEnv2:
        MOV     CX,0FFFFh       ;NOT FOUND
        REPNZ   SCASB           ;scan until 0
        CMP     ES:[DI],AL
        JZ      GetEnv9         ;if double 0 then end of environment
        JMP     GetEnv1         ;else try next one
GetEnv3:
        INC     DI              ;point past '='
        STC                     ;indicate FOUND
GetEnv9:
        POP     SI
        POP     CX
        POP     AX
        CMC
        RET
GetEnvironment  ENDP
```

And In Assembly

The Assembly Language version accomplishes the same results although in a slightly different manner. Ron copied entire "VAR=VALUE" strings into a local variable and then compared VAR, returning a newly constructed string containing the desired VALUE. For Assembly Language we'll simplify things by directly scanning through the environment (no local variables) and just returning a pointer to the proper VALUE in EnvSeg (instead of making a copy to return). This method may require a few extra steps for some uses, but in many (if not most) it will provide the necessary information without wasting time or space; that's the beauty of Assembly Language.

Figure 2 shows GetEnvironment for Assembly Language. To find the value of an environment variable, just put the address of a Pascal string (length byte followed by string) containing an environment variable name in DS:SI and call GetEnvironment. On return, if the Z flag is set, ES:DI will point to the value of the variable. If Z is not set then the variable wasn't found.

To upgrade the Assembly version of Exec (Micro C, issue #31, pages 42-43), add the GetEnvironment procedure into the source file, then change the FINDCOMSPEC procedure (bottom of page 43) to the new one defined in Figure 3.

Disclaimers

Our newly-made GetEnvironment has one big problem. It will work only in COM files. In an EXE file the "Program Segment Prefix" isn't in the code segment. Details, details. Use the TESTEXEC program in Figure 4 to test the new Exec.

In Turbo this is no problem, since the Turbo compiler can only generate COM files anyway. Be serious now — you aren't actually working on anything in Assembly big enough that it has to be an EXE, are you? At any rate, leaving this limitation for now

Figure 3 - New FINDCOMAPEC For Assembly Language EXEC

```
COMSPECSTR      DB      7,'COMSPEC'

FINDCOMSPEC:
        PUSH    SI
        MOV     PATHOFS,offset DEFPATH  ;set up default
        MOV     PATHSEG,DS

        MOV     SI,offset COMSPECSTR
        CALL    GetEnvironment
        JC      FINDCOMSPEC9

        MOV     PATHOFS,DI              ;found
        MOV     PATHSEG,ES
FINDCOMSPEC9:
        POP     SI
        RET
```

Figure 4 - New TESTEXEC To Test Modified EXEC

```
;
;       TESTEXEC.ASM - test the new exec procedure
;
;       This program MUST be made into a .COM file:
;
;               masm testexec;
;               link testexec+asmexec;
;               exe2bin testexec
;               ren testexec.bin testexec.com
;               erase testexec.exe
;
        EXTRN   Exec:NEAR

Code    segment byte public 'CODE'
        assume  cs:code,ds:code
        org     100h
TESTEXEC:
        MOV     SP,8000h
        MOV     BX,800h ;request 32k even though we'll never need it
        MOV     AH,4Ah  ;actually we are 'freeing' all but 32k
        INT     21h

        MOV     SI,80h  ;execute the command trailer
        CALL    EXEC    ;in the base page (CS:80h)

        MOV     AH,4Ch  ;(since this is a COM, CS==DS)
        INT     21h
Code    ENDS

        end     TESTEXEC
```

gives me something to talk about later.

Other Neat Uses

You aren't limited to using existing variables in the environment. You can put in your own with the SET command. Here's one I've been thinking about lately: "LANGUAGE=Turkish". Just retrieve this variable when your program starts and then load up the appropriate message file. An interesting idea when you work in a multilingual environment like we do.

Just watch out when and where you set the environment. Unfortunately, MS-DOS makes a copy of the environment for a "child" type of program, i.e., you can EXEC the SET command from within your program to change the environment, but since you're just changing a copy and not the original, the environment will appear unchanged. For now you'll just have to make do with using SET from command level or batch files.

Coming Up

Come on now! You think I'm simple minded enough to try and tell you what I'll write about next time? That never works and you know it! You can at least be sure that it will have nothing to do with the Statue of Liberty or migration patterns of the club-footed wildebeest. Well, at least nothing about the statue... ■ ■ ■

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.

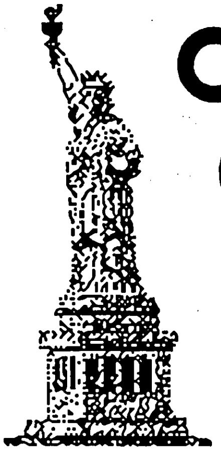
ConIX Shareware is available through CHI for just the cost of the diskette and shipping, or on-line via many popular bulletin board systems. Users who wish to register their copy pay regular advertised prices and receive the complete software package as well as full-service customer support with upgrade 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.

Product Trademarks - CP/M: Digital Research, ConIX: Computer Helper Industries, Microshell: New Generation Systems, ZCPR: R. Conn/Echelon, MS-DOS: Microsoft, UNIX: AT&T Bell Labs



ConIX™ Liberty Centennial Celebration Sweepstakes, 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 on Ellis Island*, with our *Liberty Centennial Sweepstakes.* We're simply saying thank you for being our customer by giving you a chance to win back the entire paid price of your order! Every 100 Sweepstakes entries will be placed into a separate drawing, thus making your odds of winning an incredible*1 in 100*! To enter, read, fill out, detach, and return the entry form below with your order.

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!

Sweepstakes Entry Form

Name: ______________________________

Address: ______________________________

City: __________________ State: ________ Zip: ________

Dear Computer Helper Industries:

Please enter my name in your Liberty Centennial Sweepstakes in which I am eligible to win a cash prize equal to the amount I am paying for the enclosed order. I understand that my Sweepstakes entry will be placed into a drawing of no more than 100 entries as soon as they are gathered by you, and that I will be notified by mail if I should win, at which time I will have up to ninety (90) days to claim my prize or forfeit same. I also understand that I will only be eligible to enter after you ship my order if it has been prepaid, or after payment is received if shipped COD, and I am not eligible if ordering by PO or if I've entered this Sweepstakes before.

Signature: ______________________________

Void where prohibited by law. May be cancelled without notice after first drawing

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	$ ______

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

Customizing The FTL Modula-2 Editor For The Kaypro

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

If you've been looking for Modula-2 for your Kaypro or a customizable text editor which will let you customize your keypad, this is your column.

For years I've wanted a text editor which could use the numeric keypad and cursor keys on my Kaypro II. These keys send out unique codes, but the Kaypro BIOS translates them to codes that can be generated by other keys on the keyboard.

For example, the seven key on the keypad (let's call it KP7) sends the non-ASCII code E1h, but the BIOS translates it to ASCII 37h, the same as the seven key on the main keyboard (let's call it 7). Now if I want to associate a function with KP7, it also is associated with 7. I've gained a function, but lost a digit!

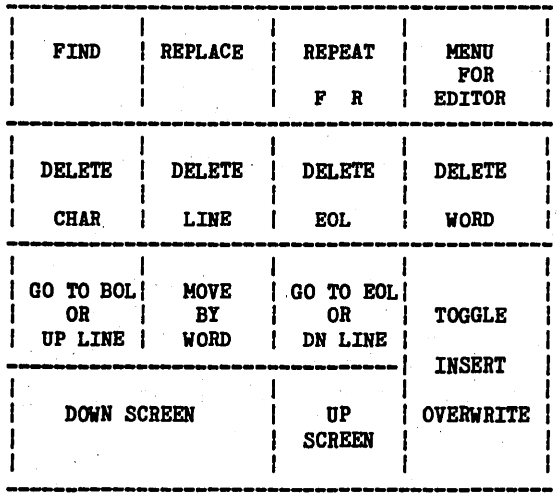
Any program that makes CP/M BDOS or BIOS calls to read a character from the keyboard gets the same character from CP/M whether KP7 or 7 is pushed. For an editor to distinguish between those keys, it must directly read the keyboard serial port. But no editor designed for a CP/M system would do that because, if it did, the editor could work only on the Kaypro.

Possible Solutions

That meant I either had to write my own editor, or get an editor with source code that I could modify. I wasn't dedicated enough to write my own, and, until recently, the source code wasn't available for any editor I might have wanted to customize.

But there's a third possibility, and probably some of you are using it; I could redirect the BDOS calls through a filter program in high memory (above the BDOS). I don't like doing this because the filter program has to be loaded separately from the editor. It also requires moving CP/M, a nontrivial procedure that reduces the TPA available to all programs.

Figure 1 - Custom Keypad For The Kaypro FTL Modula-2 Editor



```
-----------------------------------------------------
|            |           |            |            |
|   FIND     | REPLACE   |  REPEAT    |   MENU     |
|            |           |            |    FOR     |
|            |           |   F  R     |  EDITOR    |
-----------------------------------------------------
|            |           |            |            |
|  DELETE    |  DELETE   |  DELETE    |  DELETE    |
|            |           |            |            |
|   CHAR     |   LINE    |   EOL      |   WORD     |
-----------------------------------------------------
|            |           |            |            |
| GO TO BOL  |   MOVE    | GO TO EOL  |            |
|    OR      |    BY     |    OR      |  TOGGLE    |
|  UP LINE   |   WORD    |  DN LINE   |            |
---------------------------------------|  INSERT    |
|                        |            |            |
|     DOWN SCREEN        |    UP      | OVERWRITE  |
|                        |  SCREEN    |            |
|                        |            |            |
-----------------------------------------------------
```

My Choice

Well, now I have a custom editor. It's based on the FTL Modula-2 editor from Frank Hogg Labs (818-791-7979). What's more, I was able to modify and recompile that editor, which is quite powerful, in less than a day. Both the power of the editor and the ease of its modification are a direct result of the Modula-2 language and the FTL compiler. I'm impressed with both.

The FTL editor, which comes with the FTL Modula-2 compiler package (about $50), has a WordStar-like user interface, but has several advantages over WordStar as a program editor. The most important is that you can have up to three windows open into different parts of the same file or into different files. You can move easily between the windows and copy/move sections from file to file.

But that's not all: proper move by word, automatic restarting at the position you left off when you last edited a file, and nice user-definable macros (automatically saved to disk and reloaded for you). For an extra charge (about $30) you can get the source code for the editor, along with some useful utility programs.

My custom version adds several commands, remaps the cursor keys to work properly (try that with WordStar!), and provides 14 single-key commands from the keypad. These commands allow me to do most of my editing without control keys. Figure 1 shows the definition of my keypad commands.

How I Did It

In order to do this customization, I had to create a new assembly language module (KEYPORT.ASM) that reads a character from the keyboard port and translates the special codes. The FTL system comes with an assembler for just this kind of thing. I also modified three of the main modules in the editor: the one that handles terminal I/O (TERMINAL.MOD), the one that interprets keyboard entries (KEYBOARD.MOD), and the one that im-

Figure 2 - New Assembly Language Module For Editor

```
DEFINITION MODULE Keyport;

(* Used to directly read the Kaypro keyboard codes and
   map them for use by ME. GetKey is functionally equivalent
   to BusyRead from Terminal, but cursor and keypad mappings
   are different to allow keypad commands in ME.
*)

FROM System IMPORT WORD;

PROCEDURE GetKey() :WORD;

END Keyboard.
               ------------------------------------
;This is the assembly language implementation module
;
;       Keyport - read keyboard code directly
;
        label                   ;Modula-2 reference point
GetKey: in      a,07h           ;check for keyboard input
        and     01
        jr      z,out           ;done if none waiting, ret 0000h
        in      a,05h           ;get key value
        ld      hl,keymap       ;search for match using cpir
        ld      bc,mapout-keymap
        cpir
        jp      nz,out          ;done if not a special key
        ld      de,keymap
        or      a               ;clear the carry
        sbc     hl,de           ;get index of char in l
        ld      de,mapout
        adc     hl,de           ;point to output table
        ld      a,(hl)          ;get corresponding value in a
out:    pop     hl              ;get return address
        pop     de              ;remove allocation for return value
        ld      e,a             ;put return value in de
        ld      d,0
        push    de              ;put de on stack
        jp      (hl)            ;return

;mapping tables used to convert Kaypro II special key codes

mapin:
        db      0f1h,0f2h,0f4h,0f3h     ;up,down,right,left
        db      0b1h,0c0h,0c1h,0c2h     ;0,1,2,3
        db      0d0h,0d1h,0d2h,0e1h     ;4,5,6,7
        db      0e2h,0e3h,0d3h,0e4h     ;8,9,',' '-'
        db      0b2h,0c3h,0ffh          ;'.', ENTER, end of table

mapout:
        db      00h                     ;offset, saves code
        db      005h,018h,004h,013h     ;up,down,right,left
        db      003h,010h,006h,01fh     ;0,1,2,3
        db      007h,019h,01eh,01ch     ;4,5,6,7
        db      01dh,00ch,014h,00fh     ;8,9,',' '-'
        db      012h,016h               ;'.', ENTER
        end
```

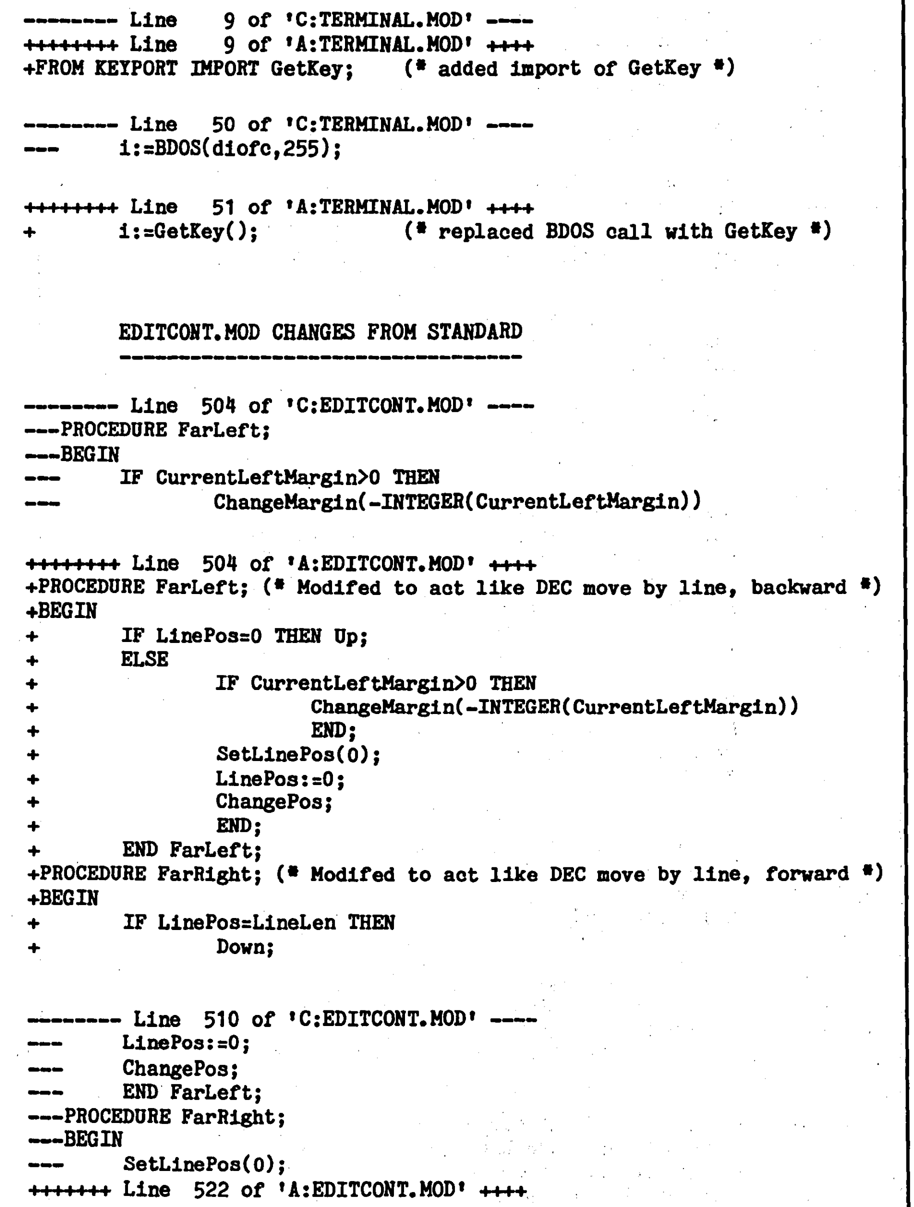
plements most of the commands (EDITCONT.MOD).

The first is required to use the new keyboard access routine. The second allowed me to replace some two-key commands with single-key commands (for example ^Q ^F to FIND a string). The third was to change two existing commands into a more useful form. The code for the Assembly Language module is given in Figure 2.

(continued next page)

(continued from page 53)

Figure 3 - Changes From Standard TERMINAL.MOD



```
-------- Line    9 of 'C:TERMINAL.MOD' ----
++++++++ Line    9 of 'A:TERMINAL.MOD' ++++
+FROM KEYPORT IMPORT GetKey;    (* added import of GetKey *)

-------- Line   50 of 'C:TERMINAL.MOD' ----
---     i:=BDOS(diofc,255);


++++++++ Line   51 of 'A:TERMINAL.MOD' ++++
+       i:=GetKey();            (* replaced BDOS call with GetKey *)



        EDITCONT.MOD CHANGES FROM STANDARD
        ----------------------------------

-------- Line  504 of 'C:EDITCONT.MOD' ----
---PROCEDURE FarLeft;
---BEGIN
---     IF CurrentLeftMargin>0 THEN
---             ChangeMargin(-INTEGER(CurrentLeftMargin))


++++++++ Line  504 of 'A:EDITCONT.MOD' ++++
+PROCEDURE FarLeft; (* Modifed to act like DEC move by line, backward *)
+BEGIN
+       IF LinePos=0 THEN Up;
+       ELSE
+               IF CurrentLeftMargin>0 THEN
+                       ChangeMargin(-INTEGER(CurrentLeftMargin))
+                       END;
+               SetLinePos(0);
+               LinePos:=0;
+               ChangePos;
+               END;
+       END FarLeft;
+PROCEDURE FarRight; (* Modifed to act like DEC move by line, forward *)
+BEGIN
+       IF LinePos=LineLen THEN
+               Down;


-------- Line  510 of 'C:EDITCONT.MOD' ----
---     LinePos:=0;
---     ChangePos;
---     END FarLeft;
---PROCEDURE FarRight;
---BEGIN
---     SetLinePos(0);
+++++++ Line  522 of 'A:EDITCONT.MOD' ++++
```

The modifications to the other modules are given (in DIF format) in Figure 3 and Figure 4. Finally I recompiled and linked all the modules to create the new editor. A submit file for this purpose is given in Figure 5.

Modula-2 makes modifying the programs very straightforward because the modules are independent except for the parameters passed back and forth in procedure calls. As James Albert pointed out in "Practical Programming in Modula-2" (Micro C, issue # 31), this is an easy language for doing complicated programs. The combination of strong structure and the ease of integrating really low level modules (directly reading a port is about as low a level as you can get) is unique to Modula-2.

I've worked extensively in various assemblers, as well as Pascal, C, and FORTRAN, and I think Modula-2 is the best general purpose language I've ever used. With the FTL compiler, it's also a fast enviromnent for program development. Turbo Pascal is nice, but FTL Modula-2 is much better for anything complex.

The FTL Modula-2 system doesn't compile as fast as Turbo Pascal because the various parts are not memory resident at all times. However, Modula-2 provides a much more sophisticated and flexible programming environment than Turbo. Its parts are still nicely linked together, e.g., the compiler loads the editor and points to the error when a syntax error is detected.

Using a RAM disk, my edit-compile-edit cycle time was about one minute on a 10K source file. The time to recompile and link the entire editor (see Listing 4) was ten minutes. These times are for a 5MHz Kaypro 8.

Here's an algorithm for modifying and recompiling the FTL Modula-2 editor:

1. Enter and compile the KEYPORT definition module.
2. Enter and assemble the KEYPORT implementation module.
3. Extract TERMINAL.MOD, STORAGE.MOD, and COMMAND.MOD from MODFILES.LBR.
4. Extract TERMINAL.DEF, STORAGE.DEF, and COMMAND.DEF from DEFFILES.LBR.
5. Modify KEYBOARD.MOD, EDITCONT.MOD, and TERMINAL.MOD.
6. Modify the submit file to fit your drive assignments. You will have to break it up into parts if you have only two double sided drives. I'm not sure that the editor can be recompiled with a stock Kaypro II, because of the limited disk storage. It would require a lot of disk swapping.
7. Enter the modified submit file(s) and SUBMIT them to generate the new version of the editor.

■ ■ ■

Figure 4 - Changes From Standard KEYBOARD.MOD

```
-------- Line  294 of 'C:KEYBOARD.MOD' ----
---     MainTable[13x]:=Left;
---     MainTable[18x]:=Down;
---     MainTable[04x]:=Right;
---     MainTable[05x]:=Up;
---
---     (*      Other control keys*)
---
---     MainTable[07x]:=DelOneChar;     (*Ctrl G*)
---     MainTable[19x]:=DelFullLine;    (*Ctrl Y*)
---     MainTable[16x]:=FlipInsert;     (*Ctrl V*)
---     MainTable[0ex]:=BreakLine;      (*Ctrl N*)

++++++++ Line  294 of 'A:KEYBOARD.MOD' ++++
+       MainTable[13x]:=Left;           (* Kaypro <- *)
+       MainTable[18x]:=Down;           (* Kaypro dn arrow *)
+       MainTable[04x]:=Right;          (* Kaypro -> *)A
+       MainTable[05x]:=Up;             (* Kaypro up arrow *)
+
+       (*     Added for remapped Kaypro Keypad   *)
+
+       MainTable[10x]:=FarLeft;        (* kp1 *)
+       MainTable[1cx]:=Find;           (* kp7 *)
+       MainTable[1dx]:=Replace;        (* kp8 *)
+       MainTable[1ex]:=DelToEOL;       (* kp6 *)
+       MainTable[1fx]:=FarRight;       (* kp3 *)
+
+       (*      Other control keys - some remapped to keypad  *)
+
+       MainTable[07x]:=DelOneChar;     (*Ctrl G or kp4 *)
+       MainTable[19x]:=DelFullLine;    (*Ctrl Y or kp5 *)
+       MainTable[16x]:=FlipInsert;     (*Ctrl V or kpENTER *)
+       MainTable[0ex]:=BreakLine;      (*Ctrl N *)

-------- Line  307 of 'C:KEYBOARD.MOD' ----
---     MainTable[03x]:=DownScreen;     (*Ctrl C*)
---     MainTable[12x]:=UpScreen;       (*Ctrl R*)
---     MainTable[14x]:=DelWord;        (*Ctrl T*)

++++++++ Line  315 of 'A:KEYBOARD.MOD' ++++
+       MainTable[03x]:=DownScreen;     (*Ctrl C or kp0 *)
+       MainTable[12x]:=UpScreen;       (*Ctrl R or kp. *)
+       MainTable[14x]:=DelWord;        (*Ctrl T or kp, *)

-------- Line  316 of 'C:KEYBOARD.MOD' ----
---     MainTable[0bx]:=DefLabel;       (*Ctrl K*)
---     MainTable[7fx]:=DelBack;        (*Del *)
---     MainTable[0cx]:=DoFind;         (*Ctrl L*)
---     MainTable[1fx]:=DelBack;        (*^- on Osborne 1*)

++++++++ Line  324 of 'A:KEYBOARD.MOD' ++++
+       MainTable[0bx]:=DefLabel;       (*Ctrl K or kp2 *)
+       MainTable[7fx]:=DelBack;        (*Del *)
+       MainTable[0cx]:=DoFind;         (*Ctrl L or kp9 *)

-------- Line  326 of 'C:KEYBOARD.MOD' ----
---     MainTable[0fx]:=Endinterpret;   (*Ctrl O*)

++++++++ Line  333 of 'A:KEYBOARD.MOD' ++++
+       MainTable[0fx]:=Endinterpret;   (*Ctrl O or kp- *)
```

Figure 5 - Submit File For Recompiling Modified Editor

```
md terminal.def
m2 terminal.mod
md storage.def
m2 storage.mod
md command.def
m2 command.mod
md files.def
m2 files.mod
md screenio.def
m2 screenio.mod
md makeedit.def
md editstat.def
md setupcal.def
m2 makeedit.mod
md editcont.def
md macros.def
md domenu.def
md menu.def
md editdisp.def
md editcont.def
md keyboard.def
m2 menu.mod
m2 editstat.mod
m2 macros.mod
m2 domenu.mod
m2 editdisp.mod
m2 editcont.mod
m2 me.mod
m2 keyboard.mod
m2 setupcal.mod
m2 macros.mod
ml me/d
```

Diddling With The Operating System From Pascal

By John P. Jones
6245 Columbia Ave.
St. Louis MO 63139
(314) 645-1596

Sometimes you need to deal with the operating system, especially when the language you're working in doesn't do what you need to do. Herein John shows you how to give your programs direct access to MS-DOS (and CP/M) so they can, for instance, do directory searches.

Within an application program there's a hierarchy of preferred service use. The tools and facilities which the language provides, or which can be written in the language, should always be used first. In some cases, even with a language as complete as Turbo Pascal, there may be a need for services at a lower level.

The first step below the language is the operating system. High level languages that are intended to operate on a variety of computers will use operating system calls to perform virtually all of their input/output functions. The operating system itself will be relatively hardware independent if it relies on calls to the system BIOS (Basic Input Output System).

In some cases there may be BIOS facilities which cannot be reached through normal operating system calls so applications may then need to do direct BIOS calls.

At the lowest level is direct hardware I/O. If you use this method in your programs, you cannot expect them to run on more than the most compatible machines.

Operating System Services

Turbo Pascal provides facilities to use all four levels of this hierarchy. This time I'll give some examples of the second level — operating system calls.

For CP/M, the most likely services you might need which aren't provided by Turbo are: reset disk (to allow a

Figure 1 - Search And Display Program For MS-DOS

```
Program get_directory;  { MS-DOS version 2 or above }

const
  get_dta = $2f00;        { get DTA address fxn }
  srch_first = $4e00;     { search first matching file }
  srch_next = $4f00;      { search next matching file }
  srch_attr = $0000;      { don't use file attributes in search }

type

  regset = record         { image of processor registers }
    ax,bx,cx,dx,bp,si,di,ds,es,flags : integer;
  end;

  dtatype = record        { image of Disk Transfer Area }
    null : array [0..20] of byte;  { used by DOS }
    attr : byte;          { file attribute spec }
    time : integer;       { coded time of day }
    date : integer;       { coded date }
    fsiz : array [0..1] of integer; { file size in bytes }
    fname : array[0..12] of char;  { filename, punctuated }
  end;

  dtaptr = ^dtatype;

var
  regs : regset;
  curnt_dta : dtaptr;
  searchname : string[64];       { room for a complete path }
  i : integer;

begin
  regs.ax := get_dta;    { request code to proper place }
  msdos(regs);           { get and assign DTA address }
  curnt_dta := ptr(regs.es,regs.bx);
  searchname := 'DATA4*.JNK'+^@;         { what files are we searching? }
  regs.cx := srch_attr; { search attributes }
  regs.ax := srch_first;{ request code }
  regs.ds := seg(searchname[1]);   { let DOS know where the string is }
  regs.dx := ofs(searchname[1]);
  msdos(regs);           { find first occurence }
  if regs.ax = 18 then   { unsuccessful? }
    writeln('No matching files')  { what went wrong? }
  else
    while regs.ax <> 18 do   { while we're successful }
    begin
      writeln;               { for demo, just ship to screen }
      i := 0;                { character array pointer }
      while (curnt_dta^.fname[i] <> ^@) and (i <= 12) do
      begin
        write(curnt_dta^.fname[i]);
        i := succ(i);
      end;
      regs.ax := srch_next;  { repeat the search }
      msdos(regs);
    end;
end.
```

Figure 2 - Search And Display Program For CP/M

```
Program get_cpm_dir;     ( CP/M version 2.2 }

const
  set_dma = $1a;         { won't use default DMA buffer for CP/M }
  srch_first = $11;      { request codes as above }
  srch_next = $12;

type
  fcb = record           { image of what CP/M expects }
    drive : byte;        { drive spec }
    fname : array[0..10] of char;  { filename }
    extent : byte;       { extent (16K block) number }
    s1,s2 : byte;
    rc : byte;           { record count for file }
    map : array [0..15] of byte;  { disk allocation blocks used }
    curr_rec : byte;     { current record for read/write }
    rndm_rec : array[0..2] of byte;   { random I/O record }
  end;

var
  result, i : integer;
  dma : array[0..127] of byte;
  our_fcb : fcb;

begin
  bdos(set_dma,addr(dma));   { make CP/M put it where we want }
  our_fcb.drive := 0;        { default drive }
  our_fcb.fname := 'DATA4???JNK';   { this is what we want }
  our_fcb.s2 := 0;
  our_fcb.extent := 0;
  result := bdos(srch_first,addr(our_fcb));   { is anybody home? }
  if result <> 255 then       { gotcha? }
    while result <> 255 do    { yeah, loop til not gotcha }
    begin
      result := result shl 5; { result * 32 = filename address offset in DMA }
        writeln;
        for i := result + 1 to result + 11 do { 8 char name, 3 char extension }
        begin
          write(chr(dma[i]));
          if i - result = 8 then write('.'); { proper punctuation }
        end;
      result := bdos(srch_next);
    end
  else writeln('No matching files found.');  { sorry about that }
end.
```

user to change a disk and write to the new one), the directory search calls, get/set user code to change to another user area, and the set file attribute function. The CP/M manuals (you should have received them with the operating system) detail them all.

The services MS-DOS provides which are not available in Turbo include file search, system clock access, file attribute get/set, and the like. They are NOT detailed in the manuals you get with MS-DOS, but in a separate, extra cost manual. An excellent alternative is Peter Norton's *Programmer's Guide to the IBM PC,* published by Microsoft Press.

A Concrete Example (Heavy Stuff)

You have written a program that can use any of several data files. To be consistent, these all have names in the range DATA4JAN.JNK to DATA4DEC.JNK. Also, the program is designed for a turnkey (NOT turkey) system, with many users not even knowing what a filename is, much less how it is structured. Within Turbo itself you can prompt for a filename, attempt to open the file, and, if unsuccessful, re-prompt or abort. Not too neat for the naive user. Alternatively, you could input the filename on the command line, but this has the same objections for the beginning user.

Both MS-DOS and CP/M provide system calls to search a disk's directory for files matching an ambiguous name. Turbo, in turn, provides a built-in procedure for making operating system calls, though they are less convenient to use than the higher level procedures.

To use system calls your program has to do all the preliminary setup, and to write this setup code you have to know quite a bit about what the operating system expects.

Figure 1 is a program to search and display all files matching an ambiguous file specification for MS-DOS, and Figure 2 is the equivalent program for CP/M.

First the DOS. All DOS calls are made through software interrupts. Software interrupts can be thought of as two-byte "long calls" (a long CALL is normally five bytes). The Turbo procedure MSDOS(registers) performs an INT 21 instruction after loading the processor registers from the record variable specified in the parameter.

MS-DOS Program

The constants declared in the program define three service requests and one DOS parameter. The service requested is normally specified in the AH register, the high order byte of the 16-bit AX register. The requests we'll need are: get DTA (data transfer area) address, search for first matching file, and search next match. The search attribute constant could be used to qualify the search based on file attributes (hidden, system, or archive).

The type REGSET is an image of the internal processor registers and is used to set up all the information DOS will need. DTATYPE defines what a DTA looks like and will allow us to retrieve the information DOS returns. (Not all system calls return data in the DTA, but the ones we use in this program do.) As you can see, we get back a lot more information than we use. We're interested in the last field, the filename. Don't mess with the first field of the record; it's not really null. It's

(continued next page)

(continued from page 57)

used by DOS for repeated calls for get next match.

The final type is a pointer to a DTA, so we can get at it no matter where in memory DOS has put it.

The first system call we make is to get the address of the current DTA, which has been set up by DOS. This address is returned in registers ES (segment) and BX (offset) and is assigned to the pointer crnt_dta. DOS expects the ambiguous filename to be in the form of an ASCIIZ string, i.e., a character string terminated by a binary zero.

By assigning our name to a "standard" Pascal string, appending a zero (^@), and letting DOS know where to find the first character (segment in DS and offset in DX) we keep everyone happy. DOS will parse a file specifier including path, drive specification, and wildcard characters, which makes things easier.

For the search first operation, we also need to put the attributes in the CX register. DOS returns a completion code in AL with the code 18 meaning "no more matches found." The matching filename can be found in the DTA as an ASCIIZ string.

CP/M Program

The CP/M system calls are set up much the same, but the data is a bit harder to retrieve after the call. The procedure/function BDOS(func, param) will put the function # in C and the parameter (if supplied) into the DE register pair of the Z80. When called as a function, the result returned in the A register is the function return value. For calls expecting a 16-bit return value, there is a related function BDOSHL.

Rather than use the default DMA buffer which CP/M sets up, the program declares a variable for the returned data and then calls the operating system to set it as the DMA area. CP/M also requires that an FCB (file control block) be set up before any related file calls are made.

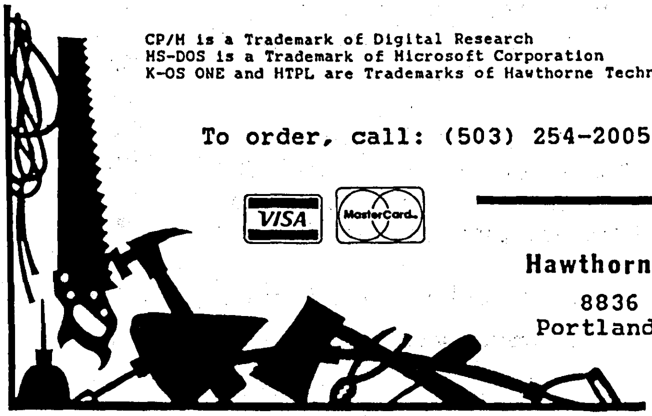
The first byte of an FCB is the drive specifier; zero means the currently logged drive, and 1..15 represent drives A..P. The filename is next, without the period between name and extension. In addition all asterisks must be expanded to strings of question marks. The extent number must be set to zero in order to find the first directory entry for each file.

The file search functions can return several values. Values 0, 1, 2, or 3 represent a successful search, with the filename to be found in the current DMA buffer at position (result * 32). The filename returned has a drive specifier prefix, which the program ignores, followed by the filename and extension without punctuation.

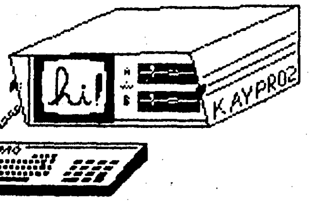
Back To The Example

Each of these programs, with a little massage, can be turned into a procedure to search a specific drive's directory. The calling program could then display a menu of choices for the user — much less error prone than expecting the user to remember what the filename should be.

■ ■ ■



Graphics for the Rest of Us



If you own an '84, '85 or '86 Kaypro CP/M computer, you can quickly and easily draw diagrams and illustrations like those shown here. All you need is the right software: SCS-Draw.

SCS-Draw turns your mild-mannered Kaypro into a powerful drawing machine, with features like:

* True pixel-by-pixel drawing and editing capability.

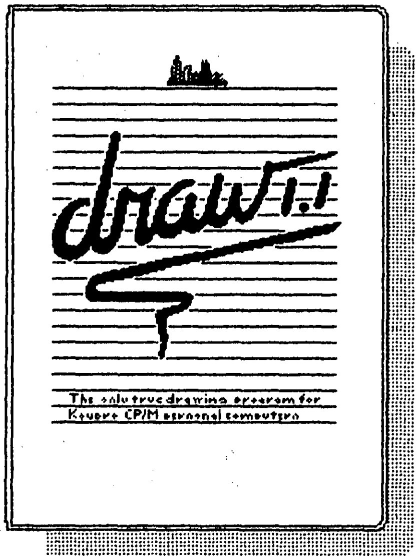
* Multi-screen drawings, for a total resolution of over 120,000 pixels. (336x362)

* Powerful printing options like rotation, enlargement, and mirror image.

* Block move and file merge.

* Context-sensitive help, pop-up menus and a 96-page user's guide. (Illustrated, of course.)

* NEW in Draw 1.1: X/Y coordinate display, vertical text for graph axis labels, print to disk capability, and much more.


draw 1.1
The only true drawing program for
Kaypro CP/M personal computers

"A fabulous program... SCS-Draw permits you to draw to the best of your abilities -- and your computer's."
- Profiles, March 1986

"A great way to fly... the power of this program is unlimited."
- Cornerstone Magazine, June 1986

"Extremely easy to use... excellent for creating logos, bulletins and newlsetters."
- Clergy KUG, July 1986

"The documentation is among the best I've seen."
- MicroTimes, August 1986

"Best use I have seen of the screen graphics."
- SCS-Draw user, Australia

"The best piece of commercial software I have bought... the quality is 4.0."
- SCS-Draw user, Idaho

"A pleasure to use... user-friendly, fun, well put together."
- SCS-Draw user, Michigan

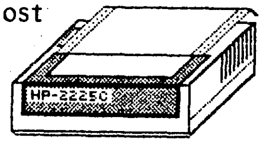
"SCS-Draw is the first program for the Kaypro that I have truly enjoyed."
- SCS-Draw user, Kentucky

Get in on the excitement — get your copy of SCS-Draw today.

[ ] Send me SCS-Draw immediately by UPS 2nd-Day Air. Payment of $59.95 is enclosed.

[ ] I have to see it to believe it -- send me a copy of the SCS-Draw Demo Program. I have enclosed $5.00 for shipping and handling.

(Specify Kaypro model when ordering.)



SCS-Draw supports most dot-matrix printers, including Okidata, Epson, Star Gemini, C. Itoh, and IBM. Daisywheel printers from Juki, Diablo and Brother are also supported, and a custom installation program is provided for printers not listed here.

Second City Software **312-577-7680**
P.O. Box 442
Mount Prospect, IL 60056

This page was created and printed using only SCS-Draw and WordStar.

Writing A Printer Driver For MS-DOS

By Don Fletcher
University of Missouri at Columbia
Rt. 7 Box 120
Columbia MO 65202
(314) 443-7294

Don Fletcher has, in a very short time, become a household word in the Micro C community. After making his name with the logic analyzer series, his "Intro To 8088 Assembly Language Programming" at the SOG V was well attended. (Well attended by everyone, that is, but Don. He wound up grounded by an ear infection.)

Herein, he more than makes up for his ears by covering not only Assembly Language programming but device drivers as well (in his own infectious style). If you're confused by segments or Assembly Language or drivers, you should hear him out.

Boy, that sure was a good deal on a parallel printer, but then a street-corner van doesn't have much overhead. No documentation with it, but after so many printers, no problem — I'll figure it out.

Okay, cable all hooked up, here goes. Hmmm, it won't print. Well, when all else fails, read the manual. Oops, let's see, where's my emergency documentation kit? (A Japanese-English dictionary.) I've never seen a printer that strobes data in a nibble at a time. Well, first time for everything. Time to find the BIOS listing, the Assembly Language reference, and get to work.

Sound familiar? That was the good old days of working with CP/M and external devices. Want a RAM disk, stepper motor driver, simple RS-232 channel with XON/XOFF protocol? Lotsa luck. (Or more realistically, lotsa work.)

MS-DOS Device Drivers

Life improved with the introduction of MS-DOS (or PC-DOS) version 2 or later because the Microsoft programmers borrowed a page from the UNIX manual and supported external device drivers. So in MS-DOS if you don't like an existing driver (such as CON, PRT, AUX, CLOCK, or a disk drive), or if you want to create a driver for a new device (such as a stepper motor controller, or RAM disk), you can write the code, and at boot, tell MS-DOS to use it.

The only difficulty is that you must write the driver in Assembly Language. But if you can modify a device driver in CP/M's BIOS, there's no reason you can't do it in MS-DOS.

This is a two-part tutorial on device drivers. I'll cover a simple character driver (a parallel printer routine) in this installment, and stepper motor and RAM disk drivers (block mode drivers) in the next issue.

Assembly Language (Quick And Easy)

First of all, we'll cover Assembly Language so you won't be totally lost when we start looking at code.

Segments

When Intel developed the 8088, it made the address space one megabyte (20 bits of address bus), but it made the address register (usually called the offset register) only 16 bits (64K). So they added four additional 16-bit registers called "segment" registers. Any one of these segment registers can be added to the offset register to create the full 20-bit address (bits 0 through 19).

The address register holds bits 0 through 15; a segment register holds bits 4 through 19. So the segment register addresses memory in a minimum of 16-byte chunks (called a paragraph), while the offset register can address memory in 1-byte minimums. Again, the 20-bit physical address is formed by adding the 16-bit segment address to the 16-bit offset address. The result is the 20-bit physical address (see Figure 1).

Figure 1 - Creating A 1 Meg Address From Two 16-bit Registers

```
16-bit offset address          0000 0000 1000 0001     =    81H
16-bit segment address   + 1000 0000 0000 1000         = 80080H

20-bit physical address   1000 0000 0001 0000 0001     = 80101H
```

Figure 2 - Example Of Segment Definition In Assembly Language

```
Stack   Segment Para Stack 'Stack'
        DB      1024 Dup (OFFH)          ;1K of stack space
Stack   Ends
;
Data    Segment Para Public 'Data'
;
Test:   DB      'This is a memory test'
;
Data    Ends
;
Code    Segment Para Public 'Code'       ;Code follows
Assume  SS:Stack, DS:Data, CS:Code
```

The four Segment registers have specific uses. The CS (Code Segment) register normally points to program instructions (code), DS (Data Segment) register points to general data, SS (Stack Segment) register is used for stack operations, and ES (Extra Segment) is used as a general purpose segment. The segment registers can point anywhere in memory.

If all four segment registers contain the same value (or point to the same general area), thus limiting the program space to 64K, you have the makings of a COM file if the origin (offset) is 100H, or a device driver if the origin is 0.

The segments are defined within an Assembly Language program using the "Segment" and "Ends" commands, and then referenced using the "Assume" command (see Figure 2).

The command "Para" tells the assembler to start the segment on the next 16-byte boundary. Each segment is named in the label field ("Stack" "Data," and "Code") in case there is more than one segment in your program. The Assume command loads the addresses of the labels (Stack, Data, Code) into the segment registers (SS, DS, CS).

In the case of our device driver, I've named the segment "Code" and referenced all segment registers (except the stack) to the same segment location. This establishes a 64K (maximum) block of memory for our driver code and data.

Procedures

The pseudo-ops "Proc" and "Endp" along with "Near" and "Far" divide your program into logical blocks known as procedures. If your procedure is within the present segment, it's a "Near" procedure (the segment register is unchanged). If it's not within the present segment, it's a "Far" procedure. "Proc" defines the start of the procedure; "Endp" is the end.

You don't know how the segment registers will be set when DOS calls your device driver, so the driver must be a "Far" procedure. Therefore, the printer driver example is defined as Far procedure "Device." The driver procedure itself calls several Near and Far procedures.

Figure 3 - 8086 Addressing Modes

MODE	CODE	DESCRIPTION
Register	Mov AX,BX	Moves the contents of BX into AX
Immediate	Mov AX,5	Put 5 (decimal) into AX
Direct	Mov AX,Test	Put the contents of memory location Test into AX
Indirect	Mov AX,[BX]	Put the contents of the memory location pointed to by BX into AX
Base + Displacement	Mov AX,[BX+3]	Put the contents of the memory location pointed to by BX+3 into AX
Base + Index + Displacement	Mov AX,[BX+SI+5]	Put the contents of the memory location pointed to by BX+SI+5 into AX

Addressing Modes

As you will see in Figure 3, the addressing modes within the 8088 are similar to the 8085.

In almost all cases, when referencing memory in an operand, the segment register used is DS (data segment). If you need to reference any other segment register, simply place it in front of the operand. For example:

```
Mov     AX,CS:[BX]
```

This would use the Code Segment register for the segment address instead of the Data Segment register. In this case you'd be copying the 16 bits of data from memory to the AX register. The memory location is calculated by adding the contents of the CS segment register to the contents of the BX register. (In this example, the BX register is the offset register.)

(Note: The processor's general purpose 16-bit registers are called AX, BX, CX, and DX. If you want to deal with just the high byte (8 bits) of AX you'd call it AH, and the low byte is AL. The same goes for B, C, and D.)

Data Structures

Data structures can help you pass blocks of data between routines. Simply define a base location, then reference the structure from that base using the 'Struc' and 'Ends' commands. You

(continued next page)

(continued from page 61)

move data into or out of the structure by using the [base.offset] format (see Figure 4).

Then within your program, if you want to move the contents of AX into Stat, and Numb into BL, the code would look something like Figure 5.

Since data is transferred in blocks to and from a device driver, the above structure format makes the code much more understandable.

PC Printer Information

The parallel printer driver must communicate with two output ports (data and control out) and one input port (control in). I've assumed in the code example, that the base address of your parallel printer port is 378H (if you have a Parallel Printer Adapter). If you have a Monochrome Display Adapter, you should change the base address to 3BCH.

The port addressing is:

```
Base address       --> Output Data
Base address + 1   --> Input status
Base address + 2   --> Output control
```

And the contents of the output control and input status ports are:

```
           Output Control
Bit Use
7 Unused
6 Unused
5 Unused
4 Enable Int - 0 Int Disabled
               1 Int Enabled
3 Select     - 1 (Always)
2 Init       - 1 neg pulse inits prtr
1 Auto Feed  - 0 no LF after CR
               1 LF after CR
0 Strobe     - 0 pos pulse sends data
```

```
         Input Status

Bit     Use

7 Busy   - 0 Printer busy
           1 Printer not busy
6 ACK    - 1 Normal input
           0 Acknowledge pulse
5 Paper  - 0 Printer has paper
           1 Printer ate the paper
4 Select - 0 Printer off line
           1 Printer on line
3 Error  - 0 Printer error
           1 Printer normal
2 Unused
1 Unused
0 Unused
```

Figure 4 - Defining A Data Structure In 8088 Assembly Language

```
Request equu    ES:[DI]  ;Base address is ES segment and DI register

Reqhdr  Struc            ;Start the structure
;
Numb    DB      ?        ;Numb is a byte with no particular value
Unit    DB      ?        ;Same for Unit
Stat    DW      ?        ;Stat is a word (16 bits)
Address DD      ?        ;Address is a double word (Segment+offset)
;
Reqhdr  Ends
```

Figure 5 - Using A Data Structure To Pass Data

```
Mov     [Request.Stat],AX      ;Move contents of AX into memory
                               ;at Request.Stat
Mov     BL,[Request.Numb]      ;Move contents of memory loction
                               ;Request.Numb into the lower 8 bits
                               ;of BX
```

Figure 6 - Definition Of Device Header Block For Device Drivers

```
Offset  Size    Description

00      4 byte  Link used by DOS to locate the driver within the chain.
                Should be loaded with 0FFFFFFFFH (-1,-1) on entry.

04      2 byte  Device attribute word, tells DOS the type of driver and
                its capabilities.  Its  format is:

                Bit    Meaning

                15       1 if character device, 0 if a block device
                14       1 if I/O Control (IOCTL) is supported
                13       1 if non-IBM format (block devices)
                         1 if output until busy (character devices)
                12       set to zero
                11       1 if open/close/RM supported (DOS 3 or above)
                10-4     Undefined set to 0
                3        1 if current clock device
                2        1 if current NUL device
                1        1 if current standard output device
                         CRT usually
                0        1 if current standard input device
                         Keyboard usually

06      2 byte  Strategy entry point offset.  Should be set to the
                strategy routine starting location within your driver.

08      2 byte  Interrupt entry point offset.  Should be set to the
                interrupt routine starting location within your driver.

0A      8 byte  Name of device padded with spaces if a character device
                (example: 'PRN     '), or number of units (1 byte) followed
                by 7 bytes of reserved space if a block device.

                (This order never varies, and cannot be changed.)
```

The program sends a byte to the Output Data port, then watches the Busy input (Bit 7) until the printer says it's ready for data. The program then raises and lowers (pulses) the strobe output (bit 0). Meanwhile, Init (bit 2), Select (bit 3) are kept high, and Auto Feed (bit 1) is kept high or low depending on your pleasure.

Device Drivers

When MS-DOS boots, it looks for a config.sys file to tell it if you want to make any changes to the operating system.

If a device is added or changed (with DEVICE = [D:][PATH]FILENAME.EXT), DOS adds the driver in FILENAME.EXT into the driver chain after the NUL device but before any other system character devices (CON, AUX, PRN, CLOCK). The user defined block mode device drivers are added after the system character drivers, but before any system block device drivers.

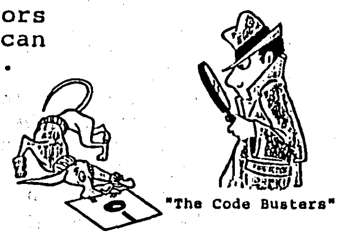
When a call is processed to a device driver, DOS scans the chain of drivers until it finds a match. Since user drivers are always first, they override the default MS-DOS drivers (unless it is a block mode driver — then you can only supplement, not supersede, the DOS driver). This allows you to change the printer, console, or supplement the disk drives to suit yourself.

You pay two penalties for using a device driver. First, there may be a slight increase in the size of MS-DOS, and second, there may be a slight increase in program execution time. Therefore, if speed is important (A/D conversion for example), and you cannot convert the driver to buffered interrupt or block mode control you should leave the code within your calling program, and access the device directly.

However, if you find yourself including the same sequence of code in several programs, or using different devices (printers for example) and having to re-configure your word processor to match, then it's probably time to write a device driver for each printer.

Figure 7 - Device Driver Request Header Format

Offset	Length	Name	Meaning
0	1	Numb	Number of bytes in request header
1	1	Unit	Unit number of this request
2	1	Cmmd	Request headers command code
3	2	Stat	Returned status
4	8	DOS	Reserved by DOS
13	1	Media	Media Descripter
14	4	Address	Data transfer address
18	2	Count	byte or sector count
20	2	Sect	starting sector value



Three Sections

Device drivers have three major sections: the Device Header Block used to communicate with the driver, the Strategy routine, and the Interrupt routines.

Device Header Block

The Device Header Block must start at offset 0 within the driver, and has a strictly defined order shown in Figure 6.

Strategy Routine

The Strategy routine has a very simple task. It must save a pointer to the MS-DOS request header. When DOS calls a driver, it calls Strategy first. Strategy receives and stores the pointer to the data, and DOS then calls Interrupt to perform the task. All Strategy does is transfer four address bytes from the ES and BX registers into memory so they can be used by the Interrupt procedure.

Interrupt Procedure

The work is done by the Interrupt procedure (actually a series of command processors), which uses the pointer passed to the Strategy procedure to locate the Request Header passed by MS-DOS. Figure 7 shows the format of the Request Header.

The character device Cmmd (command) codes are described below. I'll go into the block mode commands in the next issue. Be sure to include some type of command procedure for all the commands in your jump table (even if it's only an exit routine) in case an

(continued next page)

WRITING A PRINTER DRIVER

(continued from page 63)

unexpected command arrives. (See Figure 8.)

Still with me? Good, we're almost done.

Look over the example code for a parallel printer driver to see how all of the above comes together. If you want to try your own, here are some hints:

1. MS-DOS is not reentrant (cannot call itself), and since the driver becomes a part of DOS you cannot use any DOS calls within your code. The only exception is command 0 (Initialization) wherein you may use calls 1 through 0CH and 30H.

2. For the same reason, DEBUG won't work on device drivers unless you write a short program to set up the necessary work areas and call the driver from it. If the driver is installed in DOS, you cannot use DEBUG.

3. You can use BIOS calls, which I've found to be the easiest way to get a driver on its feet. Output a character or data to printer or screen to see if the process made it to the Init code, and so forth.

After you have written the driver, assemble it, then link it. Next convert it to a .BIN file using EXE2BIN. Using an extra DOS boot disk (please don't debug a driver using your hard disk), transfer your driver.bin file and a config.sys file that contains the command line:

```
DEVICE=driver.bin
```

(or whatever the driver is called). Then boot the system on that disk. If you get the DOS prompt your driver should be installed. To find out, try transferring some appropriate data (don't use inappropriate data — it makes too much of a mess) to the driver from the console or a file. The example driver is named SAMPLE so if you install it you should be able to dump a text file to the printer using it like so:

```
A>copy test.txt sample
```

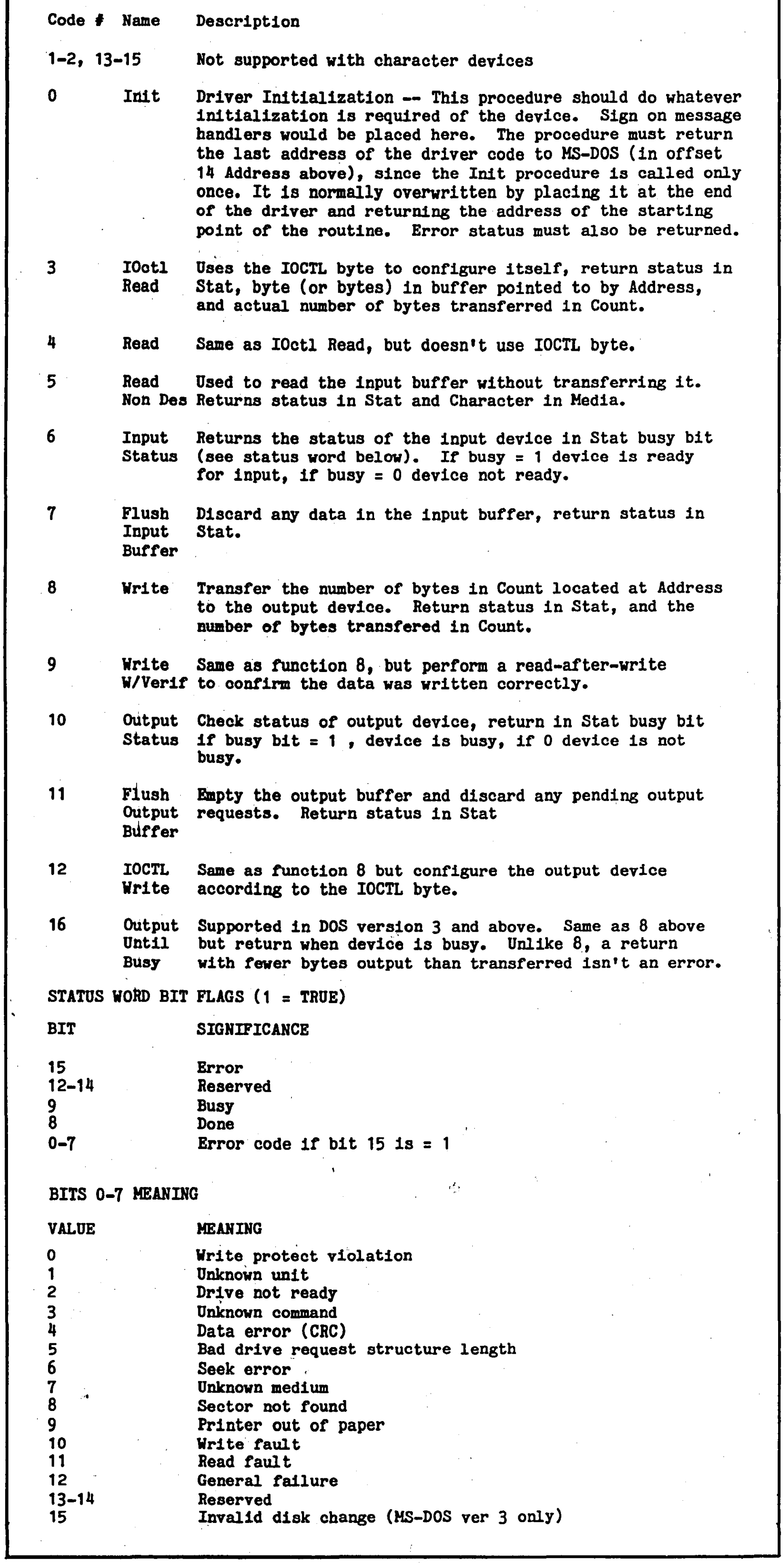
Get the idea? You can now treat your driver just like a system driver, so have fun.

Next time — RAM disk and exotic driver (stepper motor controller).

(DRIVER.ASM is on the Micro C RBBS.)

■ ■ ■

Figure 8 - Character Device Command Codes



Code #	Name	Description
1-2, 13-15		Not supported with character devices
0	Init	Driver Initialization -- This procedure should do whatever initialization is required of the device. Sign on message handlers would be placed here. The procedure must return the last address of the driver code to MS-DOS (in offset 14 Address above), since the Init procedure is called only once. It is normally overwritten by placing it at the end of the driver and returning the address of the starting point of the routine. Error status must also be returned.
3	IOctl Read	Uses the IOCTL byte to configure itself, return status in Stat, byte (or bytes) in buffer pointed to by Address, and actual number of bytes transferred in Count.
4	Read	Same as IOctl Read, but doesn't use IOCTL byte.
5	Read Non Des	Used to read the input buffer without transferring it. Returns status in Stat and Character in Media.
6	Input Status	Returns the status of the input device in Stat busy bit (see status word below). If busy = 1 device is ready for input, if busy = 0 device not ready.
7	Flush Input Buffer	Discard any data in the input buffer, return status in Stat.
8	Write	Transfer the number of bytes in Count located at Address to the output device. Return status in Stat, and the number of bytes transfered in Count.
9	Write W/Verif	Same as function 8, but perform a read-after-write to confirm the data was written correctly.
10	Output Status	Check status of output device, return in Stat busy bit if busy bit = 1 , device is busy, if 0 device is not busy.
11	Flush Output Buffer	Empty the output buffer and discard any pending output requests. Return status in Stat
12	IOCTL Write	Same as function 8 but configure the output device according to the IOCTL byte.
16	Output Until Busy	Supported in DOS version 3 and above. Same as 8 above but return when device is busy. Unlike 8, a return with fewer bytes output than transferred isn't an error.

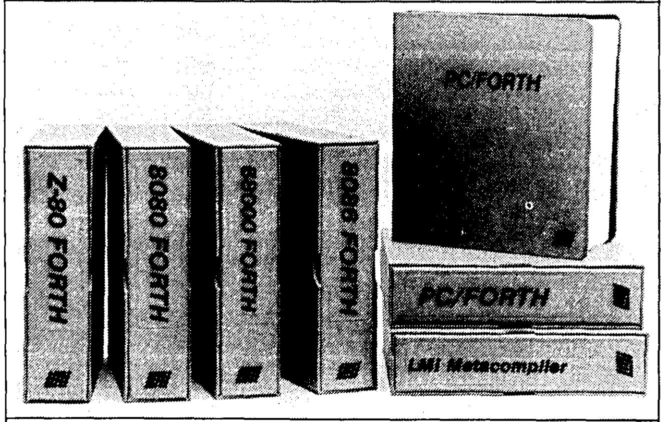
STATUS WORD BIT FLAGS (1 = TRUE)

BIT	SIGNIFICANCE
15	Error
12-14	Reserved
9	Busy
8	Done
0-7	Error code if bit 15 is = 1

BITS 0-7 MEANING

VALUE	MEANING
0	Write protect violation
1	Unknown unit
2	Drive not ready
3	Unknown command
4	Data error (CRC)
5	Bad drive request structure length
6	Seek error
7	Unknown medium
8	Sector not found
9	Printer out of paper
10	Write fault
11	Read fault
12	General failure
13-14	Reserved
15	Invalid disk change (MS-DOS ver 3 only)

TOTAL CONTROL with *LMI FORTH*™



For Programming Professionals:

an expanding family of compatible, high-performance, Forth-83 Standard compilers for microcomputers

For Development: Interactive Forth-83 Interpreter/Compilers

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 400 page manual written in plain English
- Options include software floating point, arithmetic coprocessor support, symbolic debugger, native code compilers, and graphics support

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8086, 68000, 6502, 8051, 8096, 1802, and 6303
- No license fee or royalty for compiled applications

For Speed: CForth Application Compiler

- Translates "high-level" Forth into in-line, optimized machine code
- Can generate ROMable code

Support Services for registered users:

- Technical Assistance Hotline
- Periodic newsletters and low-cost updates
- Bulletin Board System

Call or write for detailed product information and prices. Consulting and Educational Services available by special arrangement.

LMI ***Laboratory Microsystems Incorporated***
Post Office Box 10430, Marina del Rey, CA 90295
Phone credit card orders to: ***(213) 306-7412***

Overseas Distributors.
Germany: Forth-Systeme Angelika Flesch, Titisee-Neustadt, 7651-1665
UK: System Science Ltd., London, 01-248 0962
France: Micro-Sigma S.A.R.L., Paris, (1) 42.65.95.16
Japan: Southern Pacific Ltd., Yokohama, 045-314-9514
Australia: Wave-onic Associates, Wilson, W.A., (09) 451-2946

Up To Your Ears In Alligators?

If that sounds familiar, you need Write-Hand-Man™, the multi-function pop-up desktop organizer that works neatly with existing software for CP/M™ 2.2 and 3.0 systems. Write-Hand-Man eliminates that swamped feeling with tools that will get you organized. Write-Hand-Man comes with a 4-function, floating-point, 14 digit Calculator – Notepad – Two-week Appointment Book, File and Directory viewing – Phonebook with dialing – Cut and Paste – Key Redefinition – ASC II table. Even add your own applications.

Clear the swamp from your desktop.
Order Write-Hand-Man today. $49.95

CA residents add 6.5% tax. Sorry, no credit cards or purchase orders.
Specify: 8" or which 5" format
CP/M 2.2 or 3.0 format
30 day guarantee
™Write-Hand-Man – Poor Person Software
™CP/M – Digital Research

Poor PerSon
Software

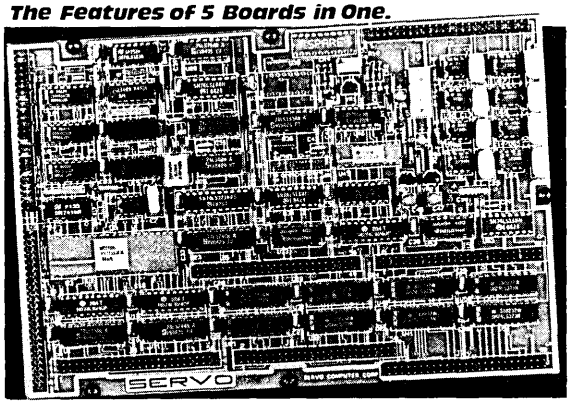
Dept. 206
3721 Starr King Circle
Palo Alto, CA 94306
(415) 493-3735

SERVO INDUSTRIAL PROCESS CONTROL BOARD

14 Bit Lab Accuracy

Industrial Durability

The Features of 5 Boards in One.



The Servo Process Control Board is a general purpose multi-channel analog and digital input/output controller which can communicate with and control most medium-scale industrial processes. Designed to work with the Servo Industrial Microcomputer, (or other computers which support the SASI interface), it offers great accuracy at prices starting as low as $495.

■ American made ■ MILSPEC components ■ 1 year warranty

Maximum configuration includes: ■ 192 analog inputs ■ 64 analog outputs ■ 128 digital inputs ■ 512 digital outputs

■ Based on the INTERSIL 14 Bit A to D converter ■ Communicates with any micro or mainframe computer ■ Easily programmed in high level languages (Turbo Pascal, Microsoft Basic, Forth) ■ PID, FFT, Thermocouple linearization and laboratory DAS software supplied. ■ Custom software and hardware engineering assistance available.

SERVO
SERVO COMPUTER CORPORATION

360-B N. Ellensburg St.
Box 566
Gold Beach, OR. 97444
(503) 247-2021

Recover A Directory By Reading & Writing Disk Sectors

By Gary Entsminger
1912 Haussler St.
Davis CA 95617

A Little Fun With DOS Interrupt 19 (13H)

If you've been using a computer for any length of time then you've no doubt run up against one of those nasty "BAD SECTOR" statements on your screen.

This is one of those frustrating occurences where the more you know, the more frustrated you get. You've got a bad bit somewhere and you're going to lose a day's or week's or month's worth of labor because of that teeny little bit.

In fact, I lost the second-best editorial ever written (we're talking major loss here) to just such a bit. Anyway, this is probably as good an introduction to MS-DOS disk operations and recovering lost data (whether in the directory or elsewhere) as any I've seen. Another winner from Gary Entsminger.

Usually, when we access a disk we're trying to get at files, a chore better left to the high-level control of DOS. But sometimes we need more control — say, we want to copy protect a disk or recover information lost in a bad sector. We can gain control via the ROM-BIOS diskette services, invoked with interrupt 19 (13 hex).

Into The ROM-BIOS

The PC ROM-BIOS offers six standard services (additional services have been added for the AT, but we won't go into them here), numbered from 0 to 5.

These services are —
0 Reset diskette system
1 Get diskette status
2 Read a sector
3 Write a sector
4 Verify a sector
5 Format a track

To invoke a service (or function) put its number in register AH (the high byte of the AX register) and call the ROM-BIOS with interrupt 19 (13H).

For example, to reset the diskette controller and drive in Assembly Language —

```
MOV     AH,0
INT     13H
```

will do the trick.

Resetting the drive this way recalibrates the read/write head (positioning the head on a specific track). This function is important during read sector operations since, often, an error status is returned the first time a read is attempted.

This read error occurs because the disk drive motor takes about a half second to reach usable (full) speed, and the ROM-BIOS doesn't wait for it. To compensate, repeat (or retry) any read function at least three times before accepting an error.

The ROM-BIOS does wait, however, for the motor to speed up on write operations, so retries aren't necessary for writes.

Reading & Writing Sectors

To illustrate the use of interrupt 19 (13H), let's consider the all too frequently occurring problem — BDOS ERROR BAD SECTOR.

When DOS, SWEEP, XTREE, or whatever program you're using re-

Figure 1 - PC-DU: A Program to Read & Write Disk Sectors

```
; PC-DU, a program to read and write disk sectors
; Illustrates disk I/O using ROM-BIOS interrupt 13h
;
;
STACK   SEGMENT PARA STACK 'STACK'
        DB      256 DUP (0) ; 256 bytes for the stack
STACK   ENDS
;
;
DATA    SEGMENT PARA PUBLIC 'DATA'
BUFFER  DB      2048 DUP (0); Enough buffer space for 4 sectors
ERRMSG  DB      'Something isn't working ! '
SUCCES  DB      'Something is    working ! '
DATA    ENDS
;
;
CODE    SEGMENT PARA PUBLIC 'CODE'
START   PROC    FAR
;
; Program prologue
;
        ASSUME  CS:CODE
        PUSH    DS       ; Save PSP Seg Addr
        MOV     AX,0
        PUSH    AX       ; Save Ret Addr offset (PSP + 0)
        MOV     AX,DATA
        MOV     DS,AX    ; Setup Data Seg
        ASSUME  DS:DATA
        MOV     ES,AX    ; Setup Extra Seg
        ASSUME  ES:DATA
;
;
; Read sectors from disk into memory
; In this example we'll read the 4 directory tracks --
; Side 0, Track 0, Sectors 4,5,6,7
```

(continued)

ports a bad sector, it usually means it's unable to access the information on that sector. The sector could be trashed (magnetic interference or the like) and totally unusable, or just slightly garbled. I'll explain.

DOS keeps track of the information it's written to a disk via a CRC. When the disk controller chip writes a sector of data onto the disk, it appends a number at the end of the sector. That number is called the CRC (cyclic redundancy check). The controller creates the number by summing all bytes in the sector (in a cyclic-redundant way) and then keeps only 16 bits of the total.

Later, when the floppy controller reads the sector it creates a new CRC number and compares it with the one appended during the write. If the numbers match, then there is only 1 chance in 65,000 that the data isn't perfect. If they don't match, the floppy controller returns an error message (along with the data).

Sometimes the sector we want to read has been only slightly garbled — a byte or two has gone bananas. If we could look at that sector, we might be able to reconstruct the bad bytes, particularly if those bytes were text. (There are certainly times when I'd accept one or two bad characters in order to save a page of material.)

By using the ROM-BIOS to read and write sectors directly, we can ignore the CRC and take our chances. If you're an old CP/M hand, you might know about DU (on Micro C's Kaypro disk #2) — a program that reads a sector into memory (ignoring the CRC status error) and writes the sector back to disk. The program I'll describe now (PC-DU) will work similarly.

There are disk problems that prevent the controller from reading anything at all (missing disk, disk inserted backwards, completely erased disk, erased sector header, emulsion scraped off the disk...). Obviously these kinds of errors are unrecoverable, but, fortunately, most of the errors are the CRC variety.

PC-DU

PC-DU (see Figure 1) is an Assembly Language shell for reading and writing

(continued next page)

PC-DU Program (continued)

```
        MOV     CX,3    ; Retry 3 times if necessary
RETRY:  PUSH    CX      ; Save retry count
        MOV     BX,OFFSET BUFFER; We'll read the sectors into
                        ; a buffer
        MOV     DL,0    ; DL gets the drive (0=A;1=B)
        MOV     DH,0    ; DH gets the side(0 or 1)
        MOV     CH,0    ; CH gets the track(0-39)
        MOV     CL,4    ; CL gets the sector(1-8)
        MOV     AL,4    ; No. of sectors to read
        MOV     AH,2    ; Service call 2 = read
        INT     13h     ; BIOS disk I/O routine
        POP     CX      ; Restore retry count
        JNC     WRITSEC ; Branch to write, if read ok
        MOV     AH,0    ; Otherwise, reset
        INT     13h
        LOOP    RETRY   ; Try again (up to 3 times)
ERROR:  MOV     BX,OFFSET ERRMSG
        CALL    DISPLAY ; Display the error message
        RET             ; Return control to DOS
;
;
; Write sectors back out to disk
;
WRITSEC:MOV     BX,OFFSET BUFFER; Here's the buffer where the
                        ; read sectors are
        MOV     DL,0    ; DL gets the drive (0=A;1=B)
                        ; Change this to 1 for a copy (see text)
        MOV     DH,0    ; DH gets the side(0 or 1)
        MOV     CH,0    ; CH gets the track(0-39)
        MOV     CL,4    ; CL gets the sector(1-8)
        MOV     AL,4    ; No. of sectors to write
        MOV     AH,3    ; Service call 3 = write
        INT     13h     ; BIOS disk I/O routine
        JNC     SUCCESS ; Branch to ok message, if ok
        JNC     ERROR   ; Else branch to error message
;
; Go get the message we want and display it
;
SUCCESS:MOV     BX,OFFSET SUCCES; Here's where the good message
                        ; is
        CALL    DISPLAY ; Display the good message
        RET             ; Return control to DOS
;
; Subroutine (or procedure) to set up display
;
DISPLAY PROC    NEAR
        MOV     CX,26   ; Display 26 characters
DISP:   MOV     AL,[BX] ; Get next char to display
        CALL    DISPCHAR; Display char
        INC     BX      ; Point to next char
        LOOP    DISP    ; Loop 26 times (value of CX reg)
        MOV     AL,0DH  ; carriage return
        CALL    DISPCHAR;
        MOV     AL,0AH  ; line feed
        CALL    DISPCHAR
        RET             ; Return control to caller of DISPLAY
DISPLAY ENDP
;
; Subroutine to actually display messages
;
DISPCHAR PROC   NEAR
        PUSH    BX      ; Save BX
        MOV     BX,0    ; Select display page 0
        MOV     AH,14   ; Function code for write
        INT     10H     ; Video interrupt
        POP     BX      ; Restore BX
        RET             ; Return control to caller of DISPCHAR
DISPCHAR ENDP
;
;
START   ENDP
CODE    ENDS
        END     START
```

(continued from page 67)

sectors. It's divided roughly into five parts —

1. Standard 8088 Assembly Language prologue
2. Data, stack, and buffer setup
3. Sector read (including retries)
4. Sector write
5. Message display & return

In my example, I've chosen to read and rewrite the directory, located on Track 0, Side 0, Sectors 4,5,6, and 7.

Long note: You could sharpen up this program by permitting a user to interactively input Drive, Track, Side, and Sector at runtime. One obvious (and simple) way to handle this is to set up the user interface in Turbo Pascal and pass the Drive, Track, Side, and Sector parameters to an external assembly language subroutine.

Figure 2 shows the Turbo Pascal shell and write sector portion of such a program. It doesn't include read retries or the interactive details. If anyone's interested, I'll put the complete program on the Micro C bulletin board. Note, in particular, how to pass the parameters from Turbo to the assembly language subroutine. BP is the caller's saved base pointer; BP + 2 is the return address, offset, and segment; BP + 4 is the first parameter (Drive) if you're calling a NEAR procedure (required by Turbo Pascal); BP + 6 is the second (Track), and so on.

The parameters are passed to the stack in the order they appear when the procedure is called. Naturally, the Assembly Language subroutine must remove these parameters from the stack in the proper order. If you're calling a FAR procedure (Microsoft Pascal, for example) add 2 for each parameter.

Briefly, note a few points in PC-DU.

Each sector requires 512 bytes worth of buffer in memory. If you're going to read one sector at a time (see Figure 2), an array of 512 bytes will handle it. To read the entire directory (see Figure 1) of four sectors we need a buffer of 2048 bytes.

Also, note the use of service 0 (reset) and the three-time allowance for retries during the read sector part of the program. If the read is unsuccessful after three retries, we'll assume something is really wrong (an unrecoverable error) and write an error message.

If all is well, we'll go on and write the sectors back out. If that's successful, we'll write a success message; if not, we'll again report an error.

Reading and writing sectors are identical with two exceptions —

Read uses ROM-BIOS service 2; Write uses service 3. And we don't retry writes.

Sum Up & Get Out

I've documented each line of code in Figure 1, so read through it for the details.

Note also that you could copy an entire disk of sectors from one disk to another by changing the drive specification on the read or write. For example, put 0 (for Drive A) in DL during the read, and 1 (for Drive B) in DL during the write to copy whatever's in the sector of choice on Drive A to Drive B.

Just for fun, I used the program in Figure 1 to copy the directory in Drive A to Drive B. Sure enough, when I checked the disk in Drive B, its directory (and nothing else!) was identical to the disk's in Drive A. Surely you innovative guys (and gals) out there can make some totally bizarre use of this capability. (A disk copier that ignores CRC errors?)

That's it; I'm out of here.

■ ■ ■

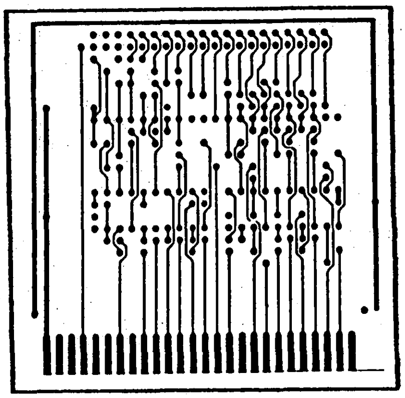


Figure 2 - Turbo Pascal and External Assembler Shell

```
program PC_DU;        {  Turbo Pascal & assembler shell:
                          for interactive PC-DU;
                          does not include retries for reads
                          or interactive details  }
TYPE
        Size_of_Sector = ARRAY[0..511] of BYTE; { set up buffer
                                                  to hold 1 sector }

VAR
        Buffer : Size_of_Sector;
        Drive, Track, Side, Sector : integer;

function SECREAD(
        VAR     B : Size_of_Sector;
                Drive : integer;
                Track : integer;
                Side : integer;
                Sector : integer)
        : BYTE;

        external 'secread.com';  { Secread is an external assembly
                                   language subroutine  }

function SECWRIT(
        VAR     B : Size_of_Sector;
                Drive : integer;
                Track : integer;
                Side : integer;
                Sector : integer)
        : BYTE;

        external 'secwrit.com';  { Secwrite is an external assembly
                                   language subroutine }

BEGIN
  IF SECREAD(Buffer,Drive,Track,Side,Sector)= 0 {If no error}
     THEN
        IF SECWRIT(Buffer,Drive,Track,Side,Sector)=0 {If no error}
           THEN  Writeln('success is ours!')
              ELSE write('Unable to read Track', Track,
                          'Side',Side, 'Sector',Sector);
END.

SECWRIT.ASM              ;subroutine to write 1 disk sector

code    segment 'code'
        assume  cs:code
;
SECWRIT proc    near
        push    BP
        MOV     BP,SP
        PUSH    DS
        POP     ES
        MOV     BX,[BP+12]
        MOV     BL,[BP+10]
        MOV     CH,[BP+8]
        MOV     DH,[BP+6]
        MOV     CL,[BP+4]
        MOV     AL,1
        MOV     AH,3
        INT     19
        MOV     AL,AH
        POP     BP
        RET     10
SECREAD ENDP
code    ends
        end
```

Computers And Education In Central America

By Leroy Searle
Delphi Development, Inc.
6273 19th Ave. NE
Seattle WA 98115

When Leroy flagged me down at SOG and told me what was going on in Central America I was really floored. I've heard of some dumb moves, but emaciating your university system is a pretty quick way to destroy the dreams of your brightest youth. This is especially true for the technical types who, outside of a university, have little or no hope of getting hands-on experience with computers.

War in Central America has not been kind to the universities there. In 1980, the University of El Salvador was invaded and sacked by the Salvadoran military. During the next four years, nearly all the university's equipment and books were sold and many of the facilities destroyed. Although the military withdrew from the campus in 1984, reconstructing the facilities has been greatly hindered by the government's refusal to allot the university its full budget as provided for in the constitution (they are receiving only 20 percent). That 20 percent barely covers faculty salaries.

Until June 1986, the university had one microcomputer serving 6000 engineering students. That computer was a cassette based Commodore 128. Assembly Language, BASIC, and FORTRAN were all taught on that one Commodore.

In June 1986, Michael Hardt (prime mover in the project) and Andrea Stoutland (a graduate student at University of Washington) delivered and installed a Kaypro II and a Morrow Micro Decision. These donated machines were installed in the "computer center" and classes were immediately changed to cover Z80 and CP/M. We've arranged to send another microcomputer and a printer, and hope to keep sending equipment until the computing center is adequately furnished.

Nicaragua Also Suffers

The 1972 earthquake and the more recent strife in war-torn Central America both have devastated Nicaragua's National Engineering University. Its three-computer laboratory serves 4000 students. (Virtually none of the computers donated to Nicaragua are available for educational use, since they've been donated with the stipulation that they not be moved from the agency to which they were given.) To make matters worse for the university, primary schools get first crack at any education funds. Therefore, the university must rely on donations from sources outside the country.

The universities in both Nicaragua and El Salvador desperately need used microcomputers, printers, disk drives, hard disks, books, software, and diskettes. These donations are tax deductible and do not violate the embargo on trade. (The Seattle Council of Churches is a sponsor of the project.) For more information contact:

Michael Hardt
611 10th Ave. E
Seattle WA 98102
(206) 328-4534 (evenings)

■ ■ ■

NOW AT THE SBC MART
COMPUTING SALE-A-THON

THE BEST OF BOTH WORLDS

Run your CP/M programs on any IBM PC or compatible. RUN/CPM transforms your PC's floppy drives into CP/M drives able to directly read, write and format over 100 CP/M disks. RUN/CPM directly executes your CP/M programs. Kaypro, Morrow, Osborne, Cromemco, Heath, many more. Because this is a hardware solution, your programs run up to 30% faster.

★ RUN/CPM only $79 *(Requires* NEC V20 chip below)

PC SPEEDUP KITS

★ NEC's V20 chip will increase the throughput of your system up to 90%. That's almost double your present processing speed. **5MHz kit $24. 8MHz $29.**

ADD-ON CARDS

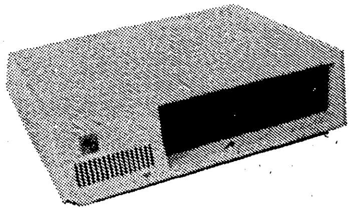
- multifunction card 384K, clk/cal w/battey, serial, parallel, and game ports, with 0K save $50 **Model MFC-4 $135**
- monochrome graphics card runs 1-2-3 graphics, w/printer port, 720×348, **MGC-1 now $135** (Hercules equiv.)
- floppy controller for 1-4 48tpi *or 96tpi* drives, w/cable save $25 **Model FDC-2 now only $74**
- color adapter w/light pen port, RGB and composite outputs reg $140 **Model CC-1 now just $116**

★ **better than the Super 7:** floppy disk controller, clk/calendar, serial, parallel & game ports, spooler & RAMDISK s/w **Sale $145** ideal mate for 640K motherboards

- clock/calendar card reg $58, **CL-1 $149**

★ OMTI hard disk controller card (this card is super fast) reg $220, **Model HDC-1 now $175**, w/cables

CASES

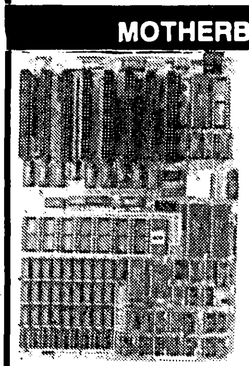


- high quality IBM PC look alike w/sideswitch, flip-top or slide chassis
- any-combo disk drive brackets
- heavy steel, 8-cutout style
- **Model CA-8 reg $95, now $69**

SBC PLEDGE

★ service after sale ★ technical support
★ low prices ★ one year warranty

MOTHERBOARDS



- full IBM PC/XT compatibility
- 8 I/O slots
- runs IBM's PC-DOS 3.1
- BASIC interpreter available
- great foundation for business or personal system
- 1 year warranty

★ **Model MB-1** 256K motherboard with 0K regularly $295, **now $199**

★ **Model MB-3 640K** motherboard with 0K regularly $399, **now $250**

★ **Model MB-2** 640K motherboard, 4.77 *and 7+* MHz clock, w/256K Reg $399, **now $297**, this is a fast board.

ABOUT OUR MOTHERBOARDS

- the most IBM PC compatible motherboards available
- each runs *all* commercially available software
- each works with all commercially available add-on cards

These FOUNDATION motherboards are the most compatible you can buy. All work with V20 chips, have parity checked memory, an 8087 socket, 8 adapter slots, four empty ROM sockets. Super manual includes complete data on how to put together a complete PC system. It even includes schematics.

HARD DISKS

- complete 10MB Kit
- **special, only $398**

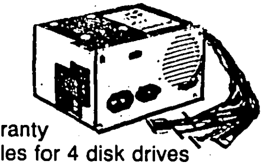
- complete internal ½-high 10MB kit, incl. controller and cables, **special $398.**
- complete 10MB kit with low power 3-½″ half-high drive, fits 5-¼″ slots, **$469.**
- 3-½″ 10MB Winchester drive alone HD-3 **just $380.**
- 20MB half-high drive HD-2 **now $395.**

HARDWARE SPECIALS

- **keyboard** 5150 style regularly $125, **Model KB-3 only $85**
- **keyboard** 5151 style regularly $175, **Model KB-3 only $99**
- **memory** nine 150ns DRAMs **Set ID-256** 256K reg $55, **now $34**
- no better **DS/DD drives** anywhere, 48tpi **F-4 $99;** F-9 *96tpi* **$129**

★ Smarteam modem (fully Hayes equivalent) 300/1200 baud auto ans/dial, reg $399, **now just $249**

POWER SUPPLY



- 135 Watts
- side switch
- top quality
- one year warranty
- standard cables for 4 disk drives

- **Model PS-135** power supply regularly $129, **now only $85**

LAST MINUTE SPECIALS

★ RUN/CPM only $79--see details in upper left corner of this ad

- IBM PC-DOS 3.1 $85 (only $65 with motherboard); Ellis PASCAL, COBOL, FORTRAN, BASIC compilers only $34 ea; MS mouse w/PC Paintbrush $145

★ COMPLETE 100% IBM PC eqiv syst $987

★ 96tpi drive package: over 800K on a single floppy, complete kit with 96tpi drive, software $149

Shipping and handling: drives/motherboards/cases/power supplies $4.50 ea, cards $2 ea, software $2.50 ea, keyboards/modems $4 ea, speedup kits $2 ea, memory $1/set, COMPUTERFACTS $3 first set then $1/set. CA residents add 6% sales tax.

***To order or for further information:* CALL (619) 375-5744**

The SBC MART, P.O. Box 1296, Ridgecrest, CA 93555

AMERICAN EXPRESS

The SBC Mart is part of Computing Technology, 821-B Commercial Street, Ridgecrest, CA 93555

By Dave Hardy
736 Notre Dame
Grosse Pointe MI 48203

Bargains In The Surplus Market

As the prices plummet in the PC arena, the movement in that direction has turned into a stampede. Meanwhile, yesterday's major players, including S-100, are showing up in unexpected places at unexpected prices. Dave helps you weed the gems from the relics.

One of the few good things to come from the inundation of IBM PC compatible hardware is that it has driven a lot of S-100 equipment onto the surplus market. S-100 machines, which are potentially more powerful than their PC brothers, are becoming available for incredibly low prices on the used market.

Many who have "upgraded" to the PC are selling their old S-100 hardware for peanuts, just to get it out of the way. It's sad that these people don't realize the abilities of their S-100 equipment, but, at the same time, it's a great opportunity for die-hard S-100 users to upgrade and expand their S-100 systems without selling the family car.

Good Deals

Which brings us to the subject of the "Good Deal." Having just seen a one-year-old three-user (TurboDos) S-100 system, complete with a Paradynamics frame, Intercontinental Micro-Systems master and slave processors, and lots of other goodies, sold by a local medium sized company for less than $1000, I can definitely say that the deals are out there. The trick, of course, is to find them, and to be able to tell the good deals from the elephant food.

Finding used S-100 equipment, especially IEEE-696-compatible hardware that can be used in modern S-100 frames, can be difficult, but newspaper ads, local computerized bulletin boards, and especially auctions and liquidation sales are often good places to start.

Once you find it, you have to be able to use it, so the following is a brief checklist I've developed (after spending more than a few dollars on some really awful S-100 stuff). If you really do want a boat anchor or a doorstop then ignore my comments.

Things To Watch Out For

COST — It's surprising how many people, especially at auctions, will pay too much money for electronic equipment, including S-100 systems and boards. Unless the equipment is still new (maybe even under warranty), it isn't a good deal unless you can buy it for less than about 25 percent of the new equivalent.

For example, a functional S-100-based Alpha Micro system that was purchased for $20,000 in 1980 (back when Alpha Micro systems WERE S-100) wouldn't be worth much more than $1000 as a "good deal," because it can be replaced today for about $4000 with an S-100-based TurboDos system.

However, a six-month-old CompuPro system with all IEEE-696 boards that originally sold for $4000 still might be a good deal at more than $2000 because all the boards are still usable in modern (that is, IEEE-696) frames.

HARDWARE — The hardware should be functional or easily repairable. Check the equipment for special parts which are available only from the original manufacturer or some other exotic source. S-100 boards using parts which can no longer be obtained, or can be obtained only from a single source, are virtually worthless on the used market. Even established S-100 vendors have destroyed the value of many of their boards by discontinuing special parts (PROMs, personality modules, etc.).

Also, beware of "REV-itis," where something looks like a great buy until the manufacturer's revision number reveals that it's an older version which doesn't work or doesn't have all of the bells and whistles. Some boards may require the dreaded "Factory Modification" to do anything at all.

SOFTWARE — If the S-100 product requires special software make sure you get it at purchase time. This is especially true if the product is a complex SBC (single board computer) or sophisticated I/O device, like a high-resolution graphics display board.

Nothing is more useless than a great piece of hardware for which you can't get software (remember the Sphere 6800?). And even if you can get software, you may pay $500 for the BIOS for your $100 SBC.

DOCUMENTATION — A good deal doesn't have to come with complete documentation, as long as complete documentation is available somewhere. After all, most good deals show up because the seller doesn't realize what he has. If he had all the information, he'd probably be asking a lot more money. Just be sure you can get the information you need to use the equipment.

Good documentation should include full schematics, even if you are not a technically oriented S-100 user (your technician will need them sooner or later), complete set-up and operations guides, and at least a basic theory of operation. If the equipment has PROMs, then a listing of the contents would be nice, although most manufacturers don't make source listings available even to original owners.

CAN YOU USE IT? — The most important question to ask when purchasing used S-100 equipment is: "Do I really need it?" (Not to be confused with "Do I really want it?" to which the answer is always "Yes.") Even an incredibly good deal is a waste of money if you can't use it.

It is usually a good idea not to buy used equipment in the hope of re-selling it to make a profit, unless you have a GUARANTEED buyer. Basements all over America are full of good deals which were purchased with the idea of being resold for a hefty profit.

Bad News For Interfacers

Sol Libes, co-author of the S-100 hardware design reference *Interfacing to IEEE-696 S-100 Microcomputers,* informs me that MicroSystems Journal (of which Sol is the editor/publisher) purchased the last 300 or so copies of that definitive S-100 manual, and that, as of this writing, only eight copies are still available. Osborne-McGraw Hill, the publisher of the book, has no plans for a second edition.

The good news is that future "S-100 Bus" columns will cover all of the major topics of S-100 bus design. Also, the IEEE-696 specification is still available from the IEEE directly, for a small fee, from their main office deep in the heart of Piscataway, New Jersey.

Next Time

Some notes about PC-compatible S-100 machines, S-100 mice, and a mini-construction project for your S-100 frame!

Future columns will also include more S-100 mini-reviews and comparisons. If you would like to see a mini-review of a certain S-100 product, please let me know, and I'll be happy to give it the once-over.

Of course, we'll also have more letters, including feedback, S-100 tips, and hints.

As always, I encourage reader feedback, and welcome questions, suggestions, comments, and ideas for future "S-100 Bus" columns. You can contact me here at Micro C, on CIS 70150,102, Source TCH054, or via GEnie mail at DJHARDY.

By Stephen M. Leon
200 Winston Drive
Cliffside Park NJ 07010

SIG/M Takes On New Line

He may be the late Stephen Leon, but he wasn't too late. The blockbuster this time is that SIG/M, that bastion of CP/Mism, is now co-mingling volumes of MS-DOS. Steve also takes on one of those fancy extended graphics adapter (EGA) cards and wins (sort of).

I did the unpardonable. I was late with copy for this column. Usually, I'm way ahead of schedule, but this time when the material was due, I was so tied up with Project Turbo that I just plum forgot about it.

I had seen the NEC Multisync monitor and decided I had to have one. Once it arrived, I needed an EGA card. One of the members of the users group had a clone EGA card, so I tried it. It turns out the people on Taiwan don't know that Microsoft's Word recognizes an EGA card by looking for the IBM copyright notice. (Using DEBUG enter "D C000:0". If you don't see the letters "IBM," Word will not know you have an EGA.)

Since we occasionally use Word, I wound up buying an Everex EGA card after they assured me on the phone that it was fully compatible. Everex makes a real nice card — BUT! I have some UCSD P Systems software, i.e., Charles Schwab's Equalizer which lets me lose money in the market without even talking to a broker. I therefore have to boot up in CGA mode because the Pascal program can't be configured for the EGA.

Everex And Quadram Don't Mix

Would you believe it. If you boot up the Everex EGA card in CGA mode and have a Quadram multifunction card in the system, the screen goes into double vision. I wound up taking out the calculator and determining that with the equipment I had in the house, I could put together another system with a turbo board for about $200 — and I did just that.

Building a turbo clone is a tale in itself. Suffice it to say, I went through three boards (including one fried by a power supply) before I was satisfied with the results. Even then, every BIOS I tried gave up something to go turbo. I finally wound up with a real IBM ROM and a hardware switch, but even then, I have yet to get the JRAM 3 to work in turbo mode. That extra speed is handy for compiling, but I ran into trouble a few times with Newword.

I have been promised a chance to test an AT board that fits inside an XT. That should be interesting.

Why tell this story? In the first place, without the people in the users group, I never could have resolved some of the problems that came up. Second, a lot of this kind of information is in the public domain libraries and on the BBS systems. Last, the extra PC is now a part of the Amateur Computer Group of New Jersey bulletin board system. You can download any of the SIG/M or PC/Blue software we discuss in Micro C by calling (201) 886-8041 without having to go through any preliminary registration procedure.

What's New At SIG/M

While the SIG/M library has traditionally featured CP/M and related software, some of the foreign libraries, such as the Australian, UK, and Japa-

nese, have tended to mix CP/M and MS/PC-DOS. We recently advised that the SIG/M library was being made available in a variety of 5" formats as well as the standard 8". Now that we are making 5" formats available, we decided to take the next step and include some MS-DOS and PC-DOS material.

Editor's note: This is a real break with tradition for SIG/M, but it's obviously a matter of survival. Don't forget that Micro C is also releasing new software into the public domain arena, so keep us in mind. However, we are pickier about program quality and usability than PC/Blue or SIG/M. But we don't exclude software just because it's shareware.

Micro C readers who would like to donate PC/MS-DOS software to the public domain are welcome to send it to SIG/M. We will not publish shareware or any software that asks for a contribution of any sort. Demo programs for commercial software are definitely thumbs down. We also prefer that source code be included. With these standards, we don't see any overlapping with the PC/Blue library, which contains mostly "send-me-money software." We are also looking for software that works in the MS-DOS orphans.

Browse For Clipper

Last issue (Micro C #31), our esteemed editor Dave Thompson explained how he discovered the wonders of Clipper. At the same time he bemoaned the lack of a Clipper equivalent of the dBASE Browse. Well Dave, the first of the SIG/M PC-DOS releases (SIG/M 284 and 185) contain the ACGNJ Mailing List program in dBASE3+ and on the disk is a Clipper compiled version. It contains a full-featured Clipper Browse.

This version does just about everything you might want a mailing list program to do. (A much simpler dBASE 2 version is available on SIG/M 110.)

I wrote the mail program, Harry Van Tassel wrote the Browse, and the disk contains both the source and the executable program. Harry's Browse does everything the real Browse does except APPEND.

The same disks tell how to add the Clipper SAVE SCREEN to dBASE3, how to have more than 20 open files in dBASE, and include an automatic .CLP and .LNK file maker for Clipper. If you're addicted to BRIEF as a code editor (as I am), Harry has also written a how-to-add-a-Time-Date-stamp to BRIEF and how-to-add-line-graphics-code to your programs using BRIEF.

Irv Hoff's IMP

Irv Hoff and Harry Van Tassel are probably two of the most prolific contributors to the public domain. While Harry always makes the source code available, Irv does not. We think Irv is wrong on that score, but we understand why he wants only one "official" version of his software.

We have now released Irv's Improved Modem Program for CP/M as a three volume set (SIG/M 272 to 274). Included on the disks are overlays for just about any computer or modem you can think of. Also in the set is Irv's version of XMODEM, called KDM17.

Another three volume set (SIG/M 275 to 277) is from the Tesseract RCPM+, Dural, New South Wales, Australia. Included in the set is a smart archive backup utility, a full screen C text editor, a volume of ZCPR3 utilities, a Z80 assembler, and a program to read MS-DOS disks in CP/M and CP/M Plus.

SIG/M Volume 280 contains some interesting Turbo Pascal programs from Jon Dart. Volume 279 has another Irv Hoff program — XIZI — a Z80 to 8080 and 8080 to Z80 translator. Also featured on 279 is a disk catalog program, a Z80 disassembler, a file dump/patch utility, and an RPN calculator.

Speaking of calculators, the Summer 86 issue of the Nantucket News (Nantucket is the publisher of Clipper) contains a program by Guenther Daubach called Das Calc. Using a function key, you can call up a calculator during a Clipper program and output the result into your program. Harry Van Tassel rewrote it and improved the graphics. It will be out with the

(continued next page)

(continued from page 75)

next batch of SIG/M PC-DOS software.

Volume 278 contains Z80DIS, a Z80 disassembler written in Turbo Pascal by Kenneth Gielow. Unfortunately, he has not released the source code, but we have released the program because of its value. The major feature of the program is an extensive analysis of the memory usage by the program being disassembled. Z80DIS produces a break table based upon this memory usage analysis.

It also gives you an address listing which itemizes all addresses referenced by the code, distinguishes types of label references, marks subroutines and jumps, and produces a fully annotated cross-reference table.

More From PC/Blue

The updated RBBS-PC (version 14.1b) is out on PC/Blue volumes 221 and 222. The Capital PC User Group deserves much credit for their maintenance of this program. It's an excellent BBS system that installs with a minimum of effort. (You can even run two BBS systems on a single 640K PC using RBBS-PC and DESQ.)

Volume 223 contains a unique how-to-do-it program — a Cantonese tutor from Louis Crew of the Chinese University in Hong Kong. For those of you with limited spreadsheet needs who cannot justify the purchase of Lotus 123, Freecalc on PC/Blue 224 may help.

I never could understand why anyone would want to do software encryption. It would seem to me that the danger of messing up your data is greater than the value of the encryption. However, if it interests you, take a look at PC-CODE on PC/Blue 225.

A general ledger program called PC-BOOKS is on 226. Volume 227 contains a single entry accounting system. Volume 228 has a double entry accounting system. Volume 231 contains a general ledger, accounts receivable, and payroll program. How good or bad these programs are I can't say. I do know that unless you know what you're doing, one of these programs can really cause trouble for a small business. All my accountant friends tell me they still find a substantial void in current micro accounting software. So, user beware.

A Must Mortgage Program

Every banker, lawyer, accountant, or real estate broker with a PC should pick up a copy of PC/Blue 226. Ronald C. Pakanowski has written MI-Analyst, a mortgage investors' forecasting tool. Some of it is dated because of the new tax law, but it has the best display of a mortgage amortization schedule I have ever seen. It does a lot more than that, and it does ask for money — but in a nice way.

Not being an engineer type, I didn't quite understand PC/Blue 229. It contains a program called D'SCOPE, which at first I thought made your PC into an oscilloscope. It doesn't. Instead, it converts the PC into a power line monitor, if you add some hardware.

Volume 230 has a Lotus learning adventure, a spline calculator, and a graph program. Templates of Doom is stated to be an interactive adventure game tutorial for 123 versions 1a and 2. Unfortunately, this is nothing more than a demo of a commercial game which you are told you can purchase for $39.95. The spline and graph program look good.

Church Management User Group

ACGNJ is contemplating setting up a Church Management User Group to help religious institutions with their computer hardware and software problems. Both the PC/Blue and the SIG/M libraries already have some church management software, but as far as we know this would be the first major effort to unite this group.

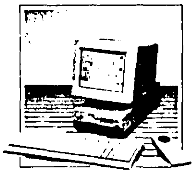
As part of this proposal, a BBS system would be set up as a clearing house for information and software. The group would probably try to develop or seek some specialized software which could also go out on SIG/M and or PC/Blue.

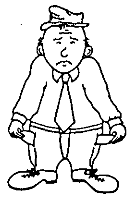
If you're interested in this program, or know someone who might be interested, please have them contact Burke Mawby at (201) 566-7445.

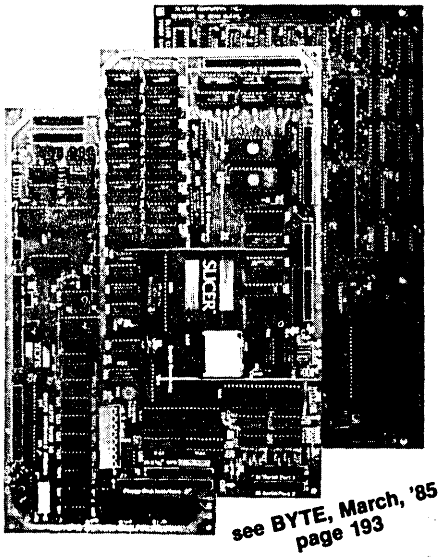
How To Order

SIG/M volumes are available on 8" SSSD disks for $6 each ($9 foreign) directly from SIG/M, Box 97, Iselin, NJ 08830. They are also available in most 5" formats. The charge for 5" disks is $7 per volume. However, for SSSD formats, or any format which requires more than one disk, please add another $2 per volume. Printed catalogs are $3 each ($4 foreign). PC/Blue volumes are $7 each ($10 foreign). The printed catalog is $5. Both are available from the New York Amateur Computer Club, Box 100, Church Street Station, New York, NY 10008. Both groups have 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.

■ ■ ■







(continued from page 2)

already seen substantial use (and abuse) and they've been great. Mike suspects that they have plated media.

Last issue we reported that the Micropolis 85 Meg was probably the most reliable drive on the market with its MTBF (mean time between failures) of 50,000 hours. No one has put 50,000 hours on one of the 20s yet (there haven't been 50,000 hours to put), but Portland folks are expecting them to make it.

Unfortunately, Micropolis drives haven't been generally available on the user market because the company prefers to deal with OEMs.

Mike also reported that his group likes the Adaptec 2010A controller. He says it's very fast (2 to 1 interleave with an XT) and will format anything with platters. The group was using the DTC before they discovered the Adaptec. (They've spent the past few months checking out keyboards, and the two which haven't worked with some of their clones are Cherry and Microswitch. They'll let us know if they find a keyboard that works with everything.)

By the way, for those of you in the Portland area, the group meets the second Wednesday, 7 p.m., in the Multnomah Art Center (an ex-grade school in Multnomah).

See the letters column for more information on hard drives.

QEX

The American Radio Relay League has recently begun publishing QEX, a journal for radio experimenters. At 20 pages it's not large, but it has quite a bit of information for those of you interested in packet radio, amateur satellites, and other new communications technologies. The June issue of this monthly magazine included the first of a series of articles on the Xerox 820-1. They are covering the 820 because it's used as a packet controller.

QEX
225 Main St.
Newington CT 06111
$12 per year, 12 issues
($6 for ARRL members)

Power Supplies

I've been working, on and off, on the next servicing article. The next is to be about power supplies, those low tech versions of high tech that can wipe out a complete system, drives and all, in mere milliseconds.

The current supplies (pun intended) are all switchers. They all take AC, rectify it and filter it into a reasonable facsimile of DC (about 300V), run it through high speed switching transistors, a toroidal transformer, and then through regulators to the system.

That's a very simplified look, but it gives you an idea.

The problem with most power supplies is that there is no protection for the computer should the supplies fail. All the supplies I've seen have spike arresters built in (same as the ones in those expensive power line filters), but spikes account for only a small percentage of the problem. Most of the damage to supplies and most system crashes are caused by dropouts (or holes) where the power goes away or sags for a few milliseconds.

Also, the input (300V) side of these switchers will occasionally self- destruct without any help from outside.

Anyway, the ICs on the main board are literally at the mercy of the power supply. They get fried by voltages that are 20 percent or more over spec.

So when Dean Klein mentioned that PC-Tech had found a better power supply, I was very interested (we've had two fail so far, and both caused significant damage to the computer system). The company is Phihong, and it's been making power supplies for Japanese manufacturers for many years. Dean was impressed with the quality of the parts and the fact that the supplies had over voltage protection built in. (Hooray, no more sizzled parts.)

I just purchased one of the 135 watt supplies. It was nearly $90 (including shipping), but it has the best quality assembly I've seen, and the fan is very, very quiet. If you purchase in quantity, prices get very competitive, so this may be a good group purchase.

Phihong USA
204 Los Gatos Blvd
Los Gatos CA 95030
(408) 395-9222

Automatic Transmission

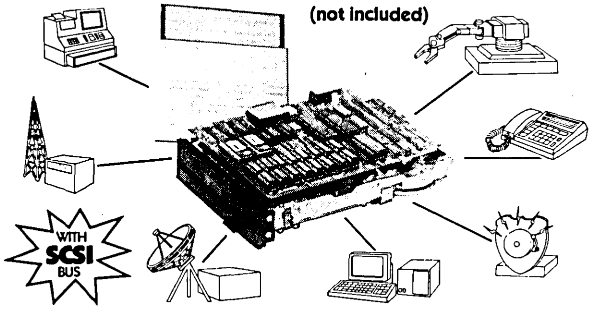
Micro C has been called the Hot Rod magazine of the computer world. OK, we'll match our speedups with anyone in the industry. We'll also match fuel economy with the slow pokes.

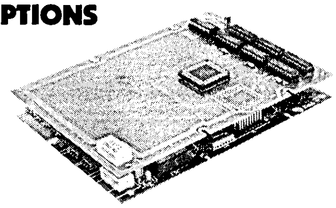
However, we're still crystal controlled. We can't speed up a system until it dies, and then back off 10 percent or 15 percent so it'll keep running. At least not without 25 or 30 crystals.

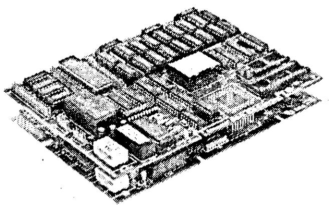
However, a small outfit in Beaverton, Oregon called Data Environment has designed and produced a variable speed clock which replaced the 8284 clock chip in the XT. Thad Young, the designer, took a VCO (voltage controlled oscillator) chip and hooked a potentiometer to its input. The VCO outputs anything from 4MHz to well over 10MHz.

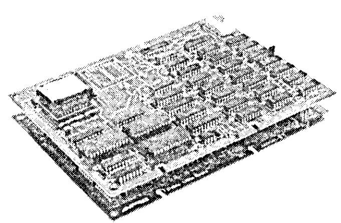
The output of the VCO goes to the 8284 on their daughter board. The daughter board plugs into the 8284 socket on the XT.

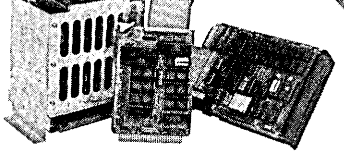
(continued next page)











AROUND THE BEND

(continued from page 79)

They brought out a switch (high speed, low speed) and a reset button on a phone cable so you can switch speeds externally. The pot is mounted on the daughter board inside the computer.

Data Environment includes a really nice software routine with the board. The software shows the current speed as well as a rectangle for each 64K bank of RAM on the board (four banks per 256K set). If one of the RAM chips fails while you're increasing the speed, its rectangle begins to blink.

The program crashes when the processor dies and when the program's bank of RAM goes to lunch.

I fired up several Sky High clones and found that the floppy controller quit between 7.5 and 7.8MHz. The processor (or kernel related ICs) died at 8.2 to 8.7MHz. The winchester interfaces were working when the kernel died. In that mix there was an 8088 (not -2) that went to 8.4MHz, a 5MHz V20 that went to 8.2MHz, and several 8MHz V20s that died at 8.2 to 8.7MHz (120ns RAM).

At 8.36MHz, the V20 gave me a Norton speed index of 3.2 (4.77MHz 8088s have an index of 1.0).

This is a fun little product. And it would be ideal for a group or shop. You could speed up a system until it died and then install a crystal that's slightly slower. You could also use this system to test boards or RAM chips.

The package is $199 retail (dealer orders are encouraged).

Data Environment
Box 5191
Beaverton OR 97006
(503) 627-9626

Cal-Tex

We no longer have any Big Board IIs at Micro C. It's not that we didn't purchase any; we had five as recently as June. It's just that we've been getting calls from desperate corporation types looking for BBIIs.

The calls began in early spring and went something like:

"Do you know what happened to Cal-Tex? We've been calling them and writing to them and no one answers. Are they out of business?"

Our five boards have gone out to those businesses which had designed products around the BBII and now have no source of boards.

I talked to Jim Ferguson, designer of the board, and he told me that Bill Siegmund (head honcho, chief technician, marketing director, and sole Cal-Tex employee) had gone to Hawaii. Temporarily.

I understand that Bill is now back in the San Jose area, the Cal-Tex office is still there, mail still arrives at the door, hundreds of populated boards gather dust on the shelves, but no one goes to the office.

They Don't Write Right

I attended a writers' conference last weekend and rubbed shoulders with some major league novelists, technical writers, romance writers, play writers, and crime writers (romance writers who love violence).

All of the technical writers used word processors. Very, very few of the other writers used computers at all (maybe 10 percent).

An Apple dealer conducted a forum on desk top publishing and demonstrated his Macintosh and Apple laser printer package (complete with software) for a mere $10,000. Later, Computerland showed up with their clone package for $2,000 (two floppies). The average professional writer makes $8,000 per year, working full time.

Anyway, these poor writers were being told that they should invest anywhere from $2,000 to $10,000, and, of course, most of them couldn't afford it. But even those who could were put off by the treatment they received from computer salesmen.

One attendee mentioned that he had found a store in the Portland area (after trying over a dozen) where the salesperson worked with him for more than an hour figuring out what he wanted to do and how she could help him do it. This guy is no dummy — he's a well-known film and TV commercial producer, but he was most impressed with the fact that she didn't make him feel like a dummy. That salesperson talked to him in writers' language, and carefully defined computer terms (byte, memory, RAM, drives, word processor) before using them.

He wound up purchasing a Leading Edge system with Word Perfect from the gal, and is delighted with the package.

"I don't know why I waited so long. If someone offered to trade a new Porsche for this computer, and stipulated that I couldn't buy another system, I'd turn him down. I can't go back. But if I hadn't found that computer store (Abacus) I'd still be scribbling on parchment."

One attendee who was knowledgeable about computers had set himself up as a computer consultant. For $20 per hour he'd work with prospective computer owners helping them decide what they needed and where to get it. He's writing his experiences for an "On Your Own."

256K Fix

We've had a few calls recently from frustrated folks attempting the 256K Kaypro upgrade from issue #30. After all that work, they would end up with only 16K of memory.

Dr. Liddle (the author of the upgrade) called from England the other day with the answer to the 16K blues. Refer to figure 1 of that article and make the following changes. On the 74LS393, pin 2 should be grounded. Jumper it to pin 7. On the 74LS153, connect pins 10 and 11 to A14b and pin 12 to PIO1.

Ray Rizzuto came up with the same answers and added another warning. He was picking up noise at the input to U3 which was dependent on the position of the board. Ray stifled the noise by connecting up three 4.7K pullup resistors: one between plus 5V and each of PIO1, PIO2, and PIO3.

Check the Micro C bulletin board for Dr. Liddle's software. It's in the KAYPRO area under the name 256KAY.LBR.

Portable EPROM Eraser

Have you've been out camping and suddenly realized that you needed to erase an EPROM? If so, here's your project. This EPROM eraser came about because I couldn't find a standard fixture for a 6 watt (9") ultra violet fluorescent tube. (For erasing EPROMs.)

I tried all the local hardware and home centers with no luck. Finally, I read the back of the package of a standard 6 watt fluorescent tube. It said: "For use in battery powered lanterns."

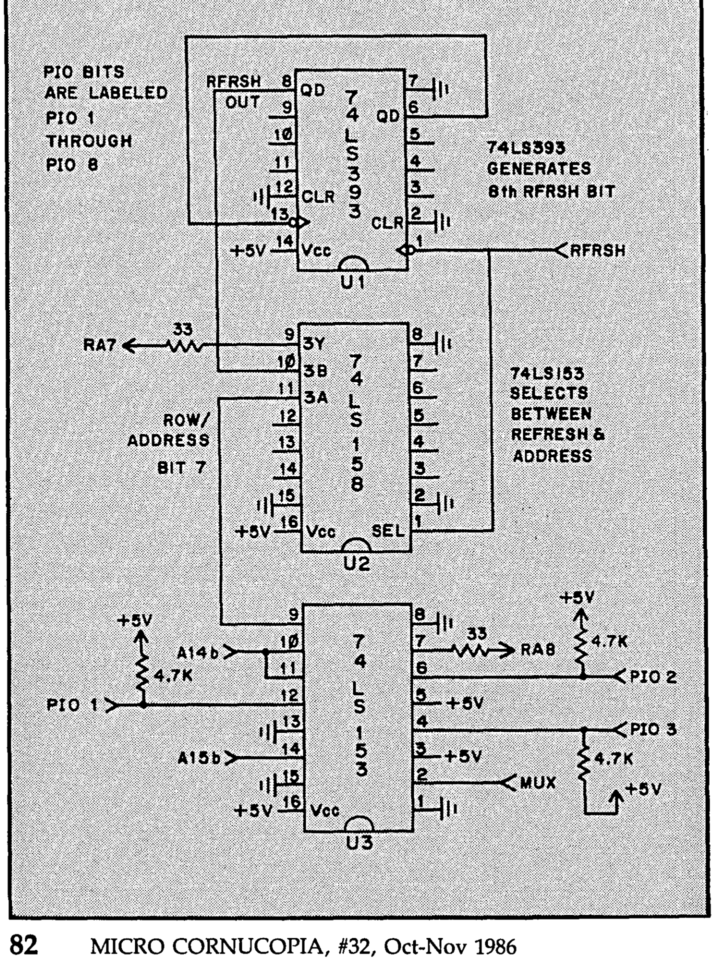
I now have an EPROM eraser and a camping lantern. Lanterns cost from $10 to $15 and are available at any department store with an automotive or camping section. The 6-watt ultra violet light costs about the same. A good source for the lights is:

Ultra Violet Products
5100 Walnut Grove
San Gabriel CA 91778
(818) 285-3123

It takes about 15 minutes to erase an EPROM under the ultra violet. Be sure not to look at the tube any longer than necessary — severe eye damage can result. (Black lights or bug lights are totally useless on EPROMs.)

Robert R. Carroll
19860 Kingswood Lane
Huntington Beach CA 92646

Figure 1 - Changes For 256K Kaypro Upgrade


PIO BITS ARE LABELED PIO 1 THROUGH PIO 8
RFRSH OUT
74LS393
U1
74LS393 GENERATES 8th RFRSH BIT
RFRSH
RA7
33
ROW/ADDRESS BIT 7
74LS153
U2
74LS153 SELECTS BETWEEN REFRESH & ADDRESS
+5V
4.7K
A14b
PIO 1
A15b
74LS153
U3
RA8
PIO 2
PIO 3
MUX

PC Speedup Tips

A couple of potential technical problems which your readers might encounter when attempting to install Larry Fogg's PC speedup which appeared in Micro C issue #31:

1. Thermal problems: The 8284A is a very high power device, dissipating as much as 1 watt under normal operating conditions. The stacked mounting scheme shown in the article does not provide adequate air flow and can cause device failure.

2. Hardware incompatibilities: Some expansion hardware such as Quadram's Quadboard expect the 14.318MHz oscillator signal to be speeded up to match the processor's clock.

3. Boot problems: While the author's system boots properly at high speed, many won't. Also, the high speed boot up will not allow users to run either IBM's standard or extended diagnostics.

4. Installation: The installation necessitates tacking wires on the motherboard and mounting the 74LS04. If removal is necessary, the wires will need to be removed.

Also, a minor note regarding RAM speed. In the article the readers are advised to check RAM for "150 microsecond parts or faster" The correct access time specifications for RAM devices are in nanoseconds.

Many systems will run fine at 7.37MHz with 200ns devices. There is never any need to use the more expensive 120ns devices; 150ns devices meet all worst case timing specifications at 7.37MHz. Alternatively, users with 200ns RAM and memory timing problems can spare themselves the chore and expense of RAM replacement by replacing the 22.11MHz crystal on the high speed oscillator with a 20.0MHz crystal which will run the system at 6.67MHz.

John K. Martinelli
Director of Engineering
Microspeed, Inc.
5307 Randall Pl.
Fremont CA 94538

Editor's note:

First: we haven't had any heat problems with 8284s and yours is the first report.

An unscientific test for overheating is to lay your hand firmly on top of the suspected chip. If the chip is hot enough to burn your hand, the chip's probably too hot. Since we have two chips stacked, you'll have to deduce the temperature of the lower. (Touch the power supply first to discharge any static.)

Second: the boot problem shows up around 7.5 MHz. It's a limitation of the floppy controller board (see the Editorial in this issue).

Third: as for wires on the board, I guess I've been playing with hardware too long to see that as anything less than a sign of maturity. A system that's reached its potential will have more than a few grey hairs. (And that goes for people too.)

Fourth, I didn't catch the ns/us error during proofreading. My fault.

As for the 120ns vs. 150ns vs. 200ns parts, a conservative rule of thumb is: Memory access time should not be longer than 1 clock cycle (though processors often leave extra time during memory access). Anyway:

8 MHz times 120ns = .960 cycle

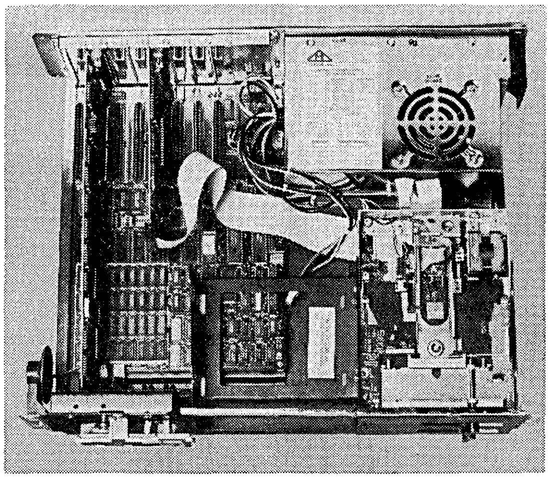
6.67 MHz times 150 ns = 1.00 cycle

Since 200ns parts will often pass 150ns or 120ns tests, it's not too unusual to find them working fine in fast machines.

Finally, most commercial speedup products and turbo boards add wait states to memory and I/O accesses, so a system will run dependably with cheap memory. For that reason, the '8 MHz' super systems we've seen have benchmarked out at about 6.6 MHz.

■ ■ ■

BUILD YOUR OWN PC XT/XT TURBO



EXPANDABILITY
8 IBM Compatible I/O Slot bus. Works with all Hardware for IBM PC/XT

VIDEO CARD
Monochrome graphic card or color graphic card

CUSTOM LANGUAGES
EPROM & ROM sockets, accepts 2764-27128 chips for custom BIOS applications.

RANDOM ACCESS MEMORY
640K RAM on board. Utilizes 41256-150 N.S. RAM chips. 250% higher reliability factor based on 2 5ths amount of memory chips required with IBM PC/XT for 640K RAM using 64K RAM chips. Memory versatility, single 256K or 640K RAM on board selectable jumper.

OPTIONAL PROCESSOR
INTEL 8087 Math CPU socket for volume numerical calculations.

ENCLOSURE
Convenient flip-top style or slide-in style metal case. Scratch resistant enamel surface.

POWER SUPPLY
150 Watt output, 110 or 220 volt operation. Quiet cooling fan. Ample current for high wattage consumption accessories.

FLOPPY DRIVES
Model Teac-55BV 5¼" ½ Height, Double Sided, Double Density, 48 TPI, 360K Storage, Direct Drive, Quiet mechanism, low power consumption.

HARD DRIVE UPGRADES
Room for two optional half height accessories. Auxiliary cooling fan mounting capability.

PC/XT COMPATIBLE ADD ON & ACCESSORIES

MONITORS

RGB Color/amber/green (TVM)	$359.00
TTL Monochrome amber/soft white (Mctek)	109.00

SYSTEM COMPONENTS

XT Motherboard 0-640K (2 layers) 4.77 mhz	119.00
XT Motherboard 0-640K (4 layers) 4.77 mhz	149.00
XT Turbo-motherboard (4 layers) 8 mhz	179.00
XT Case (flip-top)	45.00
XT Case (slide-in)	49.00
XT 150W power supply	89.00
XT 5160 keyboard (AT style) FCC approved	85.00
Floppy controller w/cable	39.00
Multi I/O card	89.00
384K Multifunction card (OK)	85.00
I/O Plus II	65.00
Color graphic card	65.00
Monochrome graphic w/printer port I	85.00
Teac 55BV Floppy drive	109.00
Assembly & Test	25.00

HARD DISK SYSTEMS:

Internal SEAGATE ST-215 10MB hard disk drive with controller	410.00
Internal SEAGATE ST-225 20 MB hard disk drive with controller	430.00
Internal 30MB hard disk drive with controller	655.00
V20-8	25.00
EasyData 1200H (half size internal modem) half size internal modem card for IBM Portable and PC/XT/AT and compatibles, same features as 1200L but without the additional serial port.	149.00
XT Assembly Manual	9.00

HARDWARE FOR IBM AT

a. AT compatible motherboard 8 slots 1 MB memory on board, expandable to 1 M	$650.00
b. Combined floppy/hard disk controller	239.00
c. Printer port/serial port card	115.00
d. 200 watt power supply	139.00
e. 1.2MB floppy disk drive	155.00
f. 2MB multifunction OK expandable to 2M, parallel port, 2 serial ports, game port	239.00
g. 3MB multifunction OK expandable to 3M, parallel port, 2 serial ports, game port	259.00
h. AT type case	89.00
i. AT compatible keyboard	89.00
j. 20MB hard disk, SEAGATE 4026	595.00
k. 30MB hard disk, SEAGATE 4038	739.00
l. 40MB hard disk, SEAGATE 4051	899.00

ACCESSORIES

Surge Suppressed outlet strip	9.50
Surge Suppressed outlet strip (UL approved)	15.95
Parallel printer cable/serial cable	12.95
Disk storage w/lock (holds 100 pcs)	14.50
Disk storage w/lock (holds 50 pcs)	10.95
Roll Top Disk Storage (holds 100 pcs)	29.50

McTEK SYSTEMS, INC.

2316 FOURTH STREET
BERKELEY, CA 94710
(415) 549-3472

DEALERS: Identify yourself as a dealer and we'll send you dealer information. We're reliable, efficient, and price competitive.

Turbo Pascal, An Almost Complete Language

By Mortimer Nerd

All you Pascal, and non-Pascal, programmers should note that there's a super bargain in the computer language field. It's Turbo Pascal. Turbo Pascal is the really neat brainchild of Philippe Borland, an old prospector who comes down from the mountains to visit computer shows.

I'd recommend that you all program in BASIC except that it's expensive, at least for a compiler and compilers make your code run faster, but faster isn't always better. I've found that the compiled Pascal code is taking as long to run as the BASIC interpreter.

Pascal

Pascal is a computer language like no other, except that it's somewhat like C and Modula II. It's also like RATFOR, which is similar to C and Modula II, too.

I learned structured programming in BASIC, and I've found that Pascal gives me everything I need to write structured code except GOSUBs. It's hard to be really structured without GOSUBs. If you don't use GOSUBs you're missing something.

A Letter

Anyway, I wrote to Mr. Borland and told him that if he were going to pretend to be structured then he had to add GOSUBs. I won't read you the letter he sent me, but it's obvious that he's been to too many computer shows (or his bear has a foul mouth). Outside of the language, his letter was pretty interesting; you should read it someday.

Fortunately, I did figure out how to add line numbers. Because of this I was able to translate my favorite BASIC program into Pascal in only two weeks. The Pascal program follows.

```
Program hiworld;
label: 1,2,3,4;
1: begin
2: writeln('hello world');
3: goto 2;
4: end.
```

This program compiled very quickly, but it's taking as long to run as the BASIC version. I checked the computer just last week and the program was still running. The BASIC version is running on my other computer. It has tied up that computer for over two months. I may have to speed up both computers to get them to finish. I'll let you know which one wins.

Next I think I'll learn another language, but I'll have to save up for another computer first. (Maybe I need hardware floating point.)

P.S. I'd like to thank the following people for helping me translate my program: Peter Casey, Larry Fogg (those Micro C people are great), Sheila Silverman, Jim Warren, Captain Crunch (unfortunately he doesn't have a phone), Mike Weisert, George Morrow, the entire staff of the Dead Pines Computer Center and Tanning Salon in Tucson, and my mother. (Hi mom.)

The Technical Torture Winners Are . . .
FIRST PLACE
Nancy Rogers
SECOND PLACE
Eric Johnson
THIRD PLACE
Sandy & Earl Brabandt
SOG Entries

Solution to last issue's Technical Torture puzzle.

S	T	A	C	K	■	B	A	S	I	C	■	N	E	W	M	O	D	S	■
M	A	C	H	I	N	E	L	A	N	G	U	A	G	E	■	G	A	■	■
A	R	T	I	S	T	■	O	■	H	■	■	Z	■	D	■	R	T	S	■
L	D	■	P	S	■	■	G	R	A	P	H	I	C	S	T	E	A	M	■
L	Y	R	E	■	■	F	I	■	R	■	O	■	■	■	H	■	B	■	H
■	H	E	X	A	D	E	C	I	M	A	L	D	A	M	E	■	A	L	E
■	A	D	A	■	R	E	N	■	■	M	E	O	W	■	O	■	S	E	A
■	R	O	M	■	A	■	O	K	R	A	■	M	■	E	R	■	E	N	D
I	D	■	S	I	M	O	N	■	■	T	H	E	O	R	Y	O	F	O	P
B	Y	E	■	■	B	A	S	E	T	E	N	■	■	S	O	M	E	■	A
M	■	S	Q	U	A	R	E	C	P	U	■	B	I	T	F	I	E	L	D
■	F	T	U	■	K	■	Q	U	A	R	T	I	C	■	O	T	S	■	D
P	I	■	A	■	I	O	U	S	■	L	■	T	■	O	P	■	■	H	I
A	R	O	D	■	N	■	I	■	■	O	F	F	■	R	E	G	R	I	N
L	S	■	D	I	G	I	T	A	L	G	R	I	P	E	R	■	E	G	G
S	T	O	R	K	■	R	E	P	E	■	E	D	S	■	A	■	N	H	■
■	B	R	I	E	F	E	R	■	■	R	A	D	I	A	T	I	O	N	■
P	A	■	V	■	I	■	■	P	I	■	K	L	■	L	I	■	S	O	S
I	N	T	E	R	F	A	C	I	N	G	■	E	R	S	O	B	I	O	S
O	K	■	S	I	O	■	■	■	S	O	G	R	E	U	N	I	O	N	S

ON YOUR OWN

By Dave Thompson

The Secrets Of Consulting

There are lots of books for consultants, serious books, funny books, readable books, and useful books. I found a book recently that was all the above and I thought you should know about it.

I was digging around for a subject for this issue's "On Your Own" when I heard about a book titled *The Secrets of Consulting* written by Gerald Weinberg, author of such technical tomes as *The Psychology of Computer Programming* and *An Introduction to General Systems Thinking.*

"He should have a computer perspective, and secrets too," I thought, and ordered it. (The bookstores here have lots of books, but the interesting ones I have to order.) I have a natural curiosity about secrets, especially published secrets.

First Three Chapters

Anyway, I climbed into the book, and an hour later found myself three chapters in and fascinated.

What I had started out hoping to find were the secrets, those hard facts everyone else had missed, key information that would mean automatic success in the consulting field. Then I'd distill those secrets into this column and make you all rich and famous consultants.

I didn't find many secrets in that first hour, but Weinberg did show me a philosophy I hadn't encountered before.

I'd already heard that businesses don't succeed or fail because of their products; they succeed or fail because of their people. Generally, I agree with that (after leaving space for a few short-term hi-tech exceptions).

Anyway, Weinberg says that business problems are really people problems. Businesses call in consultants to solve software problems, or hardware problems, or accounting problems, or financial problems, or whatever. But deep down, those problems are really people problems, and in the first three chapters Gerald Weinberg made it clear that he's at home dealing with people.

Next Six Chapters

I read three more chapters; then a fourth, fifth, sixth. It was fun. I was fascinated by this guy's ideas. Gerald was speaking to me, as though I were sitting in on one of his lectures. It was easy listening, punctuated by stories of his own life, and curious principles:

"Clients always know how to solve their problems, and always tell the solution in the first five minutes."

"The chances of solving a problem decline the closer you get to finding out who was the cause of the problem."

"Repeatedly curing a system that can cure itself will eventually create a system that can't."

"If what they've been doing hasn't solved the problem, tell them to do something else."

"Most of the time, for most of the world, no matter how hard people work at it, nothing of any significance happens."

"The best way to lose something is to struggle to keep it."

Hey, just a crummy minute. If something isn't working then people should do something else? Seems a bit obvious. And what about curing a system, or making significant change, or find the person who's causing a problem? If consultants can't (or shouldn't) do that, it doesn't leave much.

Plus, if clients tell their consultants the solutions to the problems in the first five minutes, what are consultants going to do with all their free time? And, what does any of this have to do with secrets and computers and success and the universe?

A lot.

And the further I got into the book, the more I understood why he'd been playing hardball with my assumptions. Those off the wall comments had stretched my thinking so I could handle it when the game really warmed up.

Chapter 11

In Chapter 11 he talks about marketing your services. If any chapter contains secrets, this one does. A few secrets:

"A consultant can exist in one of two states: Idle or Busy." (There is no in-between.)

"The best way to get clients is to have clients."

"Spend at least one day a week getting exposure." (While you're busy!)

"Clients are more important to you than you can ever be to them."

"Never let a single client have more than one-fourth of your business."

"Give away your best ideas."

Really interesting stuff, and it really makes sense when you follow his reasoning all the way through.

In this chapter he also states that you should spend at least a quarter of your time doing nothing. This means not marketing yourself, not bookkeeping, not smiling at the folks in the office — nothing. This is how you keep your mind fresh, how you have a little margin if an emergency arises (personal or otherwise), and how you avoid burnout.

So you're only going to spend half your time doing billable work, even when you're busiest. There are also those times when you're not working, so he figures that you should be charging five times what you need to make. For example, if you need $10 per hour, then charge $50. If you need $100 per day charge $500. If you need $1,000 a day, find another profession.

Finally

Can I recommend this book? Yes. Especially if you are a consultant or are thinking about becoming a consultant. It really does contain those little secrets that I had expected, but it also has something much more important: a way of understanding your role as consultant that should make you far more effective for your clients, and far more comfortable with yourself.

The Secrets of Consulting
By Gerald Weinberg
Dorset House Publishing
353 W 12th St
New York NY 10014
ISBN 0-932633-01-3
228 pages, about $25

■ ■ ■

2 Megabytes On The 68020

By Sigi Kluger
Definicon Systems, Inc.
21042 Vintage St.
Chatsworth CA 91311
(818) 341-5654

The Definicon folks have always kept us abreast of their latest boards and the latest of mods to their latest boards. The latest board is the 68020, the latest mod is the following memory upgrade. (Sort of a two-for-one special.)

When we designed the DSI-32 (a 32032 co-processor for the PC) we figured it should support up to 2 Meg of RAM, but because of the limited space available on a PC main board, 256K RAM chips had to be piggybacked in order to get those 2 megabytes.

That board had traces and pin holes for the "upper" RAS signal already in place, so you could simply install a wire leading from the RAS pin of the upper chip to the hole directly beneath the pin. (Incidentally, two customers insisted on getting 4 Meg of RAM that way. While it worked at first, it wasn't reliable. So please refrain from expanding memory past two meg using piggybacking.)

On To The DSI-020

Definicon's latest project, the DSI-020 (a 68020 controlled board), was designed with low cost in mind. Although intended to run at 12.5 MHz, it's worked reliably at 16.67MHz. (If you use 120ns RAM chips, an NS8419 DRAM controller instead of the 8409, and the higher speed CU/FPU.) I've even cranked one up to 20MHz and am using it for software development.

As far as memory space is concerned, however, the DSI-020 was NOT designed to be expanded past one meg.

Despite that, I figured it must be possible to piggyback chips the same way it's done on the DSI-32.

Checking the schematics, I noticed that the DRAM controller has four RAS outputs. Only the first one is used by the DSI-020; it's connected to all RAS pins via a 220 ohm resistor. So what's to keep us from using the second RAS output for the second megabyte? Nothing!

About Those Warranties

At this point, let me warn you that "following along" will make all warranties on the DSI-020 NULL AND VOID. This procedure is not recommended by Definicon Systems, Inc. So, if you want to play it safe, restrain your memory hungry programs. But if you're the hardware hacker I know you are, then let's get started!

You'll need the following:

— 1 DSI-020 co-processor board (see your local hardware store)
— 1 220 ohm resistor, 1/8 watt
— 1 ft. wire-wrap wire (30 gauge)
— 40 (36 plus 4 spare) 256K dynamic RAM chips (150ns for 12.5MHz, 120ns for 16.67MHz)
— a 20-30 watt soldering iron
— solder
— screwdriver
— wire cutters

Start by removing the DSI-020 from your computer. (Remember to turn the power off first.)

Now follow these instructions:

1. Remove all RAM chips from the DSI-020. There'll be 36 chips if you have parity, 32 if you don't.
2. Take one of the RAM chips you bought and stick it on top of one of those you removed from the board. You may have to devise some means of holding them together for a while. (Make doubly sure the pin 1s are together.)
3. CAREFULLY solder all leads together, EXCEPT FOR PIN 4. Do NOT allow any excess solder to remain on the pins.
4. Carefully bend pin 4 of the top chip (the only one that's not soldered to the bottom chip) all the way up. Note that if you bend the pin back down, it WILL break!
5. Insert the piggybacked chip into the DSI-020.
6. Repeat steps 2 through 5 for all remaining RAM chips.
7. Cut the leads on both sides of the 220 ohm resistor to about 1/8 inch.
8. Solder one end of the resistor to pin 29 of the DRAM controller. The DRAM controller is the large 48-pin chip to the right of the RAM array. The resistor goes on the component side, sticking straight up in the air.
9. Now run a wire from the other end of the resistor to every pin 4 sticking up from the RAM array. Use your own ingenuity, but be sure the wire doesn't short out anything, and use as little wire as possible.
10. Plug the board into the computer and smoketest it (i.e., turn it on).

Troubleshooting

Not much can go wrong if you work carefully. You will, of course, have observed all static electricity precautions, so it's unlikely you'll have zapped a chip. Nevertheless, it is possible that a chip or two won't work.

If the card doesn't work, but you've run a memory test on the 8086 side (see below) and the card passed, you might try moving the delay line tap.

The delay line, located just below the clock crystal, has a set of four jumper pairs located to its left. There's one jumper installed, which you can move left or right until you've found a satisfactory setting. When the jumper is set correctly the board should run.

You can now run the 68020 memory test, MEMTEST.HEX, to see if the RAM can be accessed by the 68020. MEMTEST should indicate that it's testing memory up to 1FFF00 or some such. When you load the monitor it should come up with a message saying it's loaded at 001F0000.

More Disclaimers

Let me remind you again that upgrading your DSI-020 to 2 Meg, or attempting to do so, will void warranties. The procedure isn't approved by Definicon. Forget about the fact that this was written by a DSI employee. As far as you're concerned, the instructions herein were written by an anonymous hardware hacker.

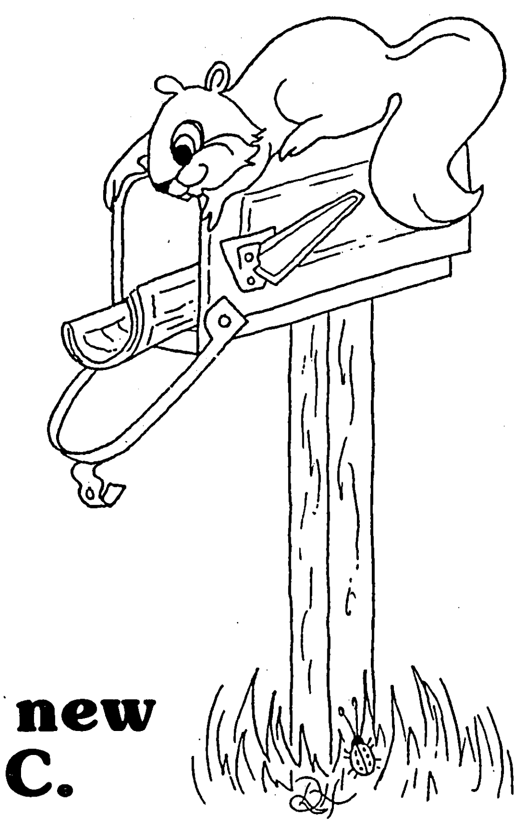
And, again, be careful. Don't break the board beyond repair. Don't modify your board without being able to thoroughly test and troubleshoot it! Write (if you need to) memory test routines BEFORE touching the board.

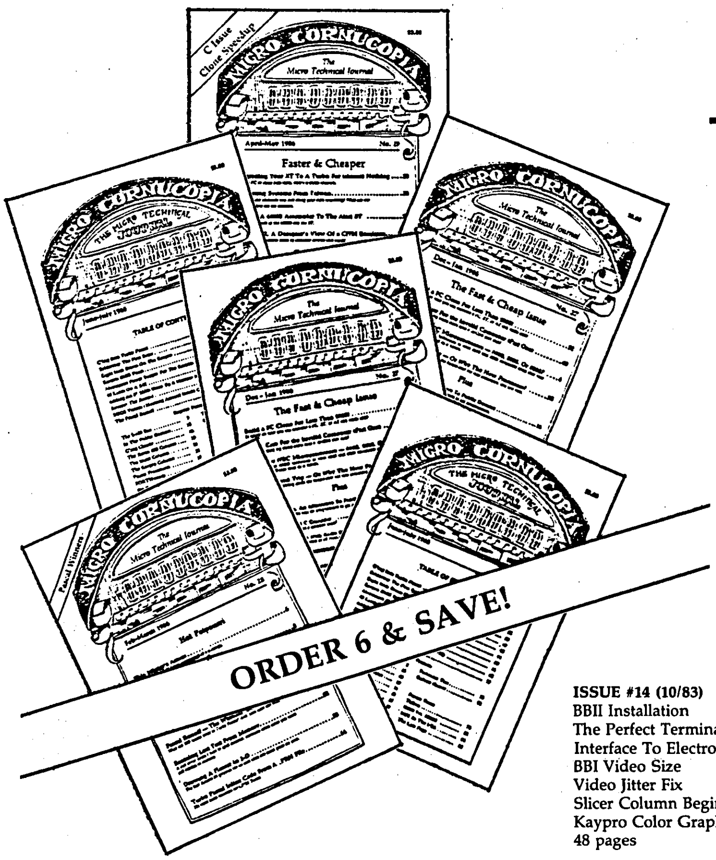
I would consider RANTEST.A86 essential for testing RAM on the PC side. RANTEST is available in source and object form on Trevor Marshall's BBS (805-492-5472 or 805-493-1495, directory path is 68020\dsi020) and in source only in the "dsi.32bit" conference on BIX and on the Micro C bulletin board (503-382-7643).

Finally, be aware that you can minimize trouble by buying your RAM chips from a reputable source. I don't wish to publicly endorse anyone, but if you write to me I'll tell you a good source.

Good luck!

□ □ □





Complete Your Education

If you're not working with a full set, fill out your collection of Micro C today.

BACK ISSUES OF MICRO C.

Only $1.50 each when you order 6 or more
All Foreign $2.00 each for 6 or more
(includes surface postage)

ISSUE #1 (8/81)
Power Supply
RAM Protection
Video Wiggle
1/2 PFM.PRN
16 pages

ISSUE #2 (10/81)
Parallel Print Driver
Drive Motor Control
Shugart Jumpers
Program Storage Above PFM
1/2 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 SIOs
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
Self-Loading ROM
Terminal In FORTH
24 pages

ISSUE #7 (8/82)
6 Reviews Of C
Adding 6K of RAM
Viewing 50 Hz
On Your Own Begins
24 pages

ISSUE #8 (10/82)
Drive Maintenance
Interfacing Drives
Installing A New BIOS
Flippy Floppies
C'ing Clearly Begins
Xerox 820 Begins
28 pages

ISSUE #9 (12/82)
BBII EPROM Program
Relocating Your CP/M
Serial Print Driver
Big Board I Fixes
Bringing Up WordStar
Cheap RAM Disk
32 pages

ISSUE #10 (2/83)
Saving A Flakey Disk
Hooking Wini To BBII
The Disk Doctor
JRT Fix
Serial Keyboard Interface
Pascal Procedures Begins
36 pages
SOLD OUT

ISSUE #11 (4/83)
BBI Expansions
BBII Details
Dyna, RAM Disk Review
Easier Reverse Video Cursor
Plannercalc Review
Kaypro Column Begins
36 pages
SOLD OUT

ISSUE #12 (6/83)
256K For BBI
Bringing UP BBII
dBase II
Look At WordStar
Double Sided Drives For BBI
Packet Radio
5 MHz For Kaypro
40 pages

ISSUE #13 (8/83)
CP/M Disk Directory
More 256K FOR BBI
Mini Front Panel
Cheap Fast Modem
Nevada COBOL Review
BBI Printer Interface
Kaypro Reverse Video Mod
44 pages

ISSUE #14 (10/83)
BBII Installation
The Perfect Terminal
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 Adventrue
SBASIC Column Begins
Upgrading Kaypro II to 4
Upgrading Kaypro 4 to 8
48 pages

ISSUE #16 (12/84)
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 C0-Processor
DynaDisk For The BBII
Serial Printer On BBI Sans SIO
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

BACK ISSUES

U.S. Regular Price	**$3.00 each**
6 or more 1/2 price	**$1.50 each**
Regular Foreign Price (air mail)	**$5.00 each**
6 or more (including Canada)	**$2.00 each**
	(surface postage)

WANT ADS

The following folks are reaching you for only 20 cents per word. If you would like to reach the same audience, send your words and 20 cents for each to Micro Cornucopia.

Public Domain UG Software Rental: CP/M UG Vol 1-92 on 46 8" flippies, $45, SIG/M UG Vol 1-270 on 135 8" flippies, $175. PICONET Vol 1-34 on 17 8" flippies, $25, Pascal-Z UG Vol 1-30 13 8" flippies, $25, KUG (Charlottesville) 74 disks, $80, IBM-PC SIG 1-454 PC-DOS, $475, PC-Blue 1-220 MS-DOS, $230, directory disks $5 PP. Also available on 209 5.25" formats. Rental is for 7 days after receipt with 3 more days grace for return. Credit cards accepted (preferred). Downloading-disk format conversions. Call. User Group Software Automatic Update Service, $3.00 per disk. 619-727-1015 info. 24 hrs. 619-749-7534 9-5. National Public Domain Software Center, 1533 Avohill, Vista, CA 92084.

"Getting Started With CBASIC" - Tutorial workbook for beginners/intermediates in this powerful language. Fundamentals through file handling, user functions, etc. Gets you started and takes you up to advanced features and functions. Ends confusion, frustrations! $25/U.S. Fourcorners Press, 222 Washington St., Hanover, MA 02339.

DDraw 3.0 gives you total control of Kaypro graphics! Flowcharts, circuit diagrams, business graphs and more! Epson compatible printer required for printouts. Works on all 1984+ Kaypro CP/M computers. For your copy with extensive manual, send check or money order for $49.95 to: Titan Software, 39 13th St. #2, Troy NY 12180-4203.

For $12 postpaid, I will send you the neatest "gadget" you can have for your small-screen Kaypro. MAGGIE! is a flat plastic lens that slips into place in front of your screen (without tools) and gives you the effect of a 12 inch screen. Virtually no distortion, and no more glare from ambient light than your regular screen. Money back guarantee of satisfaction. Kay-Pal, 107 Hudson Dr., Hendersonville, NC 28739. Allow three weeks for delivery.

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.

Flight Plan Program with NAV-AID database covering entire U.S. "Install" for your airplane's performance. CP/M or IBM. Supports user entered "custom" waypoints. Completely menu driven. Thoroughly documented. $29.95. For program description send SASE to NAVIGATE, 3051 Shirley Dr., Newbury Park, CA 91320.

CP/M Emulates PC-DOS! Innovative utilities let CP/M reassign drives, autofind files (even overlays!), transform user areas into "disk drives," create smart batch jobs, "window" the BDOS, format text columns, and much more. Inexpensive, copy-enabled, 30-day trial. LOGIC ASSOCIATES, 1433 Thome, Chicago, IL 60660, (312) 274-0531. Ask for free catalog, reviews.

$120 Value: Micro C Pro-884 MAX ROM (Normal version) and PLUS-4 Decoder Board. $55 postpaid. Richard Brewster, R.D.1, Brackney PA 18812, (717) 663-2408.

Organize Your Hard Disk Tired of wading through confusing manuals? Step-by-step instructions show you how to manage your hard disk. Begin immediately with our MS-DOS help system and library of batch files. On 5 1/4" disk. $12.50 PP. Gordon Thomas, 759 S. State St. #125-A, Ukiah CA 95482. Free information sheet.

BBI System For Sale BBI, 2 Shugart drives, Odd Ball Electronics cabinet, keyboard, Zenith monitor, lots of software. $450 or best offer. David Lubkeman, 4131 The Oaks Drive, Raleigh NC 27606. (919) 737-2336 daytime.

DDraw 3.0 gives you total control of Kaypro graphics! Flowcharts, circuit diagrams, business graphs and more! Epson compatible printer required for printouts. Works on all 1984+ Kaypro CP/M computers. For your copy with extensive manual, send check or money order for $49.95 to: Titan Software, 39 13th St. #2, Troy NY 12180-4203.

Version Control System - TLIB (tm) stores ALL versions of your source code in ONE compact library file, even with hundreds of revisions. Updates (deltas) 5 - 7 times faster than UNIX SCCS. Date and comments with each version. Easy retrieval. LAN-shared libraries. Synchronized control of related source files. Free public domain MAKE (with Lattice C source) by Landon Dyer. MS/PC-DOS 2.x/3.x $99.95 + $3 s/h. VISA/MC. Burton Systems Software, POB 4156, Cary NC 27511-4156. (919) 469-3068.

640K Mother board Upgrade 640K memory without using an expansion slot! Allows easy insertion of 256K chips on the system board. 100% compatible. 30 day $$ back guarantee, 2 year limited warranty. COMPAQ Portable & Plus - $29; $99 with memory. IBM XT, 3270 & Portable - $39; $109 w.m. IBM PC - $50; $129 w.m. $4 s/h. Site discount. Dealers welcome. Product of Innoventions, Inc. Aristo, 16811 El Camino #213-S, Houston TX 77058, (713) 480-6288, 800/3ARISTO.

DataCOM Super Systems (tm) is your one-stop headquarters for all the latest public domain and shareware programs. We support CP/M, CP/M+, CP/M86, MS-DOS, PC-DOS, and TurboDOS users with over 100MB of online files. DataCOM is a remote multi-user TurboDOS database with 2 incoming phone lines and Courier 2400 modems. We have an international reputation as the Tampa Bay Kaypro Users Group and have been online since July 1982. Call (813) 791-1454 by modem (300/1200/2400 baud) and download an application or send $35 to: DataCOM Systems, 2643 Cedarview Ct., Clearwater FL 33519. DataCOM also has over 120 disks of CP/M and 150+ disks of IBM software available by mail for $10 each. Catalogs available if you send a disk in a reusable mailer and include appropriate return postage. Specify Kaypro SSDD, DSDD, or IBM DSDD format.

Xerox 820 w/Emerald DSDD Upgrade. Has SSDD drives. Complete w/CP/M. Like new condition. $450. Juki 6200 daisy wheel, 18 cps on 4th ribbon, $325. Ken, Sonoma CA (207) 935-1001.

Science Software for the Commodore 64 and C-128 (in C-64 mode). Includes hi-res color graphics and capability to save to most popular printers. Programs for amateur astronomy, earth satellites and aeronautics. $19.95 - $34.95. Demo disk and catalog available for $5. Science Software, 7952 W. Quarto Dr., Littleton CO 80123.

All New & Unused: Big Board II A&T with CP/M $200; Morrow MD-1 $500; MD-2 $550; MD-3 $600; Smith-Corona TP-1 daisywheel $200; Cromemco CS-3 $1000; Otrona Attache $1200; Wicat 150-1FS $1500; Wicat 150-3WS $2500. Polygon Industries (504) 486-2154.

Turbo Budget! and Turbo Cheques! CompuTech announces the release of Turbo Budget!, the financial companion to Turbo Cheques!, the most comprehensive personal checkbook management program available for the IBM PC. Turbo Budget! provides budget management tracking capabilities. Update budget expenses automatically as you write checks. Both programs are professionally developed, easy to use, support color, hard disks, contain no copy-protection, print checks, password controlled, extensive screen displays, reports, and Turbo Budget utilizes window technology. 140+ page users manual. FREE demo disk available with self-addressed, .54 stamped mailer. For the IBM PC/XT/AT and high PC compatibles. Price: $49.95 for both programs. Send check or M.O. to: CompuTech, Box 7000-309, Redondo Beach CA 90277. Call me, Ralph Freshour, for more information at (213) 377-7198.

■ ■ ■

TIDBITS

By Gary Entsminger
1912 Haussler St.
Davis CA 95617

Secret Computers & Turbo Tutor

Gary looks at a new version of Turbo Tutor, and the Heel computer (it has a leg up on the roulette wheel). If you've been looking for a no-risk gamble, or wondered how to make an easy million using parts out of an old Apple, then read on.

You might have noticed a new address in my byline. Dave's charged me with the care of Micro C's California office (what, you didn't know Micro C had a California office?), so I'll be writing and editing from Davis while my wife, Alison, attends graduate school at the University of California.

Editor's note: Our only other office is in Turkey, and Laine is staffing that one.

If you have ideas for articles you'd like to see in Micro C, or questions about articles we've already published, write me here. I'd love to hear from you.

A Piece Of The Pie

I've just been treated to an outrageous reading experience — "The Eudaemonic Pie" by Thomas A. Bass.

Bass, a science writer, was part of a group of Santa Cruz crazies who designed, built, and tested a computer system for predicting the results of spins of a roulette wheel. As if the notion itself weren't enough, consider the computer — it fits into the sole of a shoe.

In Bass's Terms

"We are slipping into our soles a CMOS 6502 microprocessor with 5,000 bytes of RAM. Apple computers are made with the same chip. (*Editor's note: But let's not hold that against it.*) We carry another 4,000 bytes of memory crafted into a program smart enough to beat roulette at a 44 percent advantage.

"The program — a set of mathematical equations similar to those used by NASA for landing spaceships on the moon — tracks a ball in orbit around a spinning disk of numbers. During the ten to twenty seconds in which the game is played from beginning to end, the computer calculates coefficients of friction and drag, adjusts for changes in velocity, plots relative positions and trajectories, and then announces where in this heavenly cosmos a roulette ball will likely come to rest on a still-spinning rotor.

"Its predictive power lies in the fact that the computer in our shoes can play out in microseconds a game that in real life takes a million times longer."

The payoff in roulette is 35 to 1, so a 44 percent advantage, according to Bass's calculations, can produce a return of as much as $2200 per hour if only a few hundred dollars are invested. Not bad, not to mention the pleasure one would derive from beating up on a heavy favorite (gambling losses for bettors are figured in the billions per year).

In the beginning, there was no venture capital, so each member of the group donated time, skills, ideas, etc. to the project. Eventually, they planned to beat the casinos in Reno, Las Vegas, and Tahoe with their computer-in-a-shoe and divide the winnings (the pie).

"The Eudaemonic Pie" is the funny, skillfully written story of the design and implementation of one of the wildest computer systems I've ever heard of. Read it for an entirely new experience.

From Vintage Books, it's in paperback for $5.95. (Don't try to purchase the computer from your local Apple outlet. At least not just yet.)

Turbo Tutor

Borland International continues to produce pleasant surprises — this one a rehash of an old Borland product, the Turbo Tutor (which I wrote about back in issue #23).

The manual and program have been extensively revised to include many advanced Turbo Pascal topics —

Stacks, queues, deques, lists, trees and graphs, assembly language interfacing, accessing parameters from external subprograms, libraries & more.

For example, the little shell in Figure 1 illustrates how an external program accesses the parameters of the calling program (or procedure).

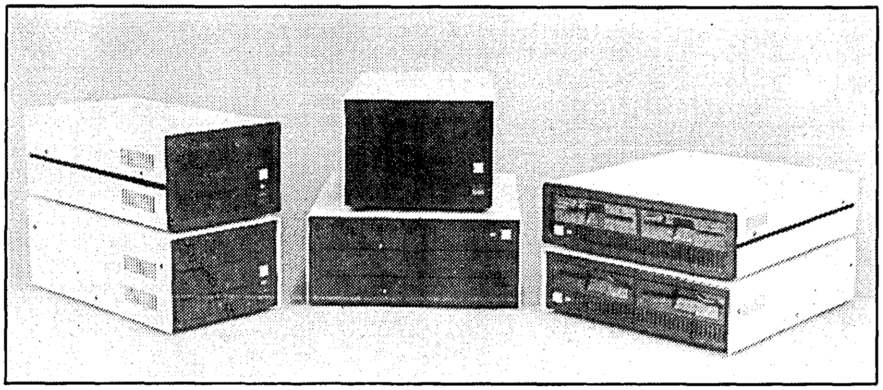
To clear things up a little, picture the stack after executing the first two instructions of SIMPLE (PUSH BP and MOV BP,SP) —

```
                High Memory
                ____________
     BP + 8  | SEG (X)     |
             |-------------|
     BP + 6  | OFS (X)     |
             |-------------|
     BP + 4  | Y(integer)  |
             |-------------|
     BP + 2  | RET ADDRES  |
             |-------------|
     BP      | OLD BP      |
             | 2 BYTE WD   |
              -------------
                Low Memory
```

"The stack grows downward from SS:SP to SS:00, which means it starts at high-memory addresses and moves toward low-memory addresses. Everything is pushed onto the stack in 2-byte quantities.

"The parameters are pushed onto the stack immediately before the call to SIMPLE, and the call itself uses the return address to be pushed as well. To access the parameters, the subprogram can use the BP register. Once BP has been loaded with the previous value of SP, it points the stack segment at a location relative to the parameters."

Since Y is an integer, it occupies two bytes. X, a VAR parameter, consumes four bytes. All addresses, regardless of the size of the object they reference, take up four bytes of stack.



"When a subprogram is completed, it returns control to the calling routine with a NEAR RET instruction. If the program is called with parameters, they must be popped from the stack. In addition, if the subprogram is a function, a function-result variable may need to be popped as well."

The RET instruction performs this task automatically. In SIMPLE, the RET 6 instruction deallocates six bytes — two for Y and four for X. If a procedure has no parameters, just use RET without specifying a number of bytes.

Perhaps the most significant feature of the new Tutor is the program itself which provides a split screen — descriptive text below and running program above — to create a very effective learning environment.

If you're just starting out, or in need of some really indepth information about Turbo Pascal, you'll have to look long and hard for a better tutorial and reference. Highly recommended!

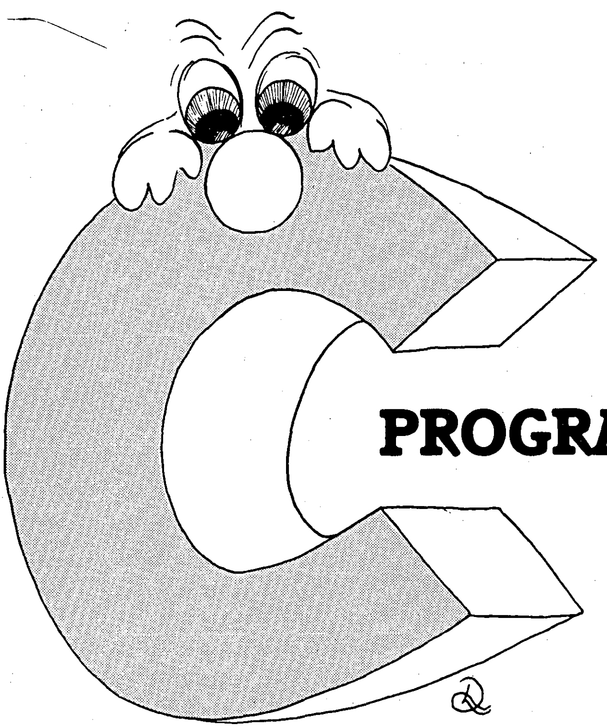
And that, friends, is Tidbits.

■ ■ ■

Figure 1 - Assembler Code To Add Two Numbers

```
CODE    SEGMENT
ASSUME  NOTHING
; procedure Simple(var X : integer; Y : integer);
;
; Simply adds X to Y and returns the sum in X.
; Note that X is a var parameter (address passed
; on the stack) and Y is a value parameter
; (value passed on the stack).
;
SIMPLE  PROC    NEAR
        PUSH    BP      ; Save old BP,
        MOV     BP,SP   ; Load in SP so
                        ; BP can be used
                        ; to address
                        ; parameters.
        LES     DI,DWORD PTR [BP +6]
                        ; Move X's address
                        ; into ES:DI
        MOV     AX,[BP+4]
                        ; Move Y's value
                        ; into AX
        ADD     WORD PTR[DI],AX
                        ; Add Y's value to X's
        POP     BP      ; Restore BP
        RET     6       ; Return, popping 6 bytes
                        ; of parameters from stack
                        ; 4 for X (a var parameter)
                        ; 2 for Y (an int value
                        ; parameter)
SIMPLE  ENDP
CODE    ENDS
        END
```

MICRO CORNUCOPIA'S C CONTEST



If you've been going so hard you just can't C straight, then drop everything, dig out the old compiler, brush off the cobwebs, and C what you can C.

Any kind of C. From Small-C to large, from a hundred lines up, this is your chance to C'cure fame and glory (plus valuable prizes).

PROGRAMMING CONTEST

GRAND PRIZE:

- DSI-32, 1 meg, 10 Mhz 32032 computer board with assembler/linker from Definicon Systems
  21042 Vintage St., Chatsworth, CA 91311
- Optimizing C Compiler (large model)
  from Manx Software, Box 55, Shrewsbury, NJ 07701
- Essential Software's C Library
  (graphics, I/O, everything)
  from Essential Software, Inc.
  Box 1003, Maplewood, NJ 07040
- 3 year subscription to Micro C

SCORING — The programs will be judged by the Micro C staff on the following:

0-15 for ALGORITHM
0-25 for CODE ORGANIZATION & READABILITY
0-40 for PROGRAM FUNCTION
0-20 for USER DOCUMENTATION

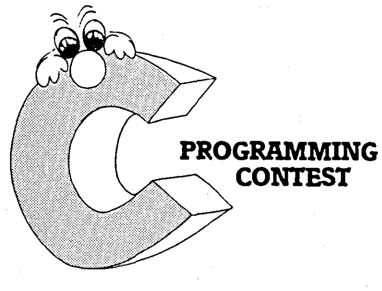
1st & 2nd Place

- Optimizing C Compiler (large model)
- Essential Software's C Library (graphics, I/O, everything)
- 2 year subscription to Micro C

3rd - 5th Place

- Aztez C Prime Package
  (interpreter & compiler)
  from Manx Software
- 1 year subscription to Micro C

--------- CONTEST DEADLINE November 1, 1986 ---------



ENTRY FORM

Program Title ________________________________

Purpose ________________________________

Which C compiler? ______________ Target system(s) ____________

Please include source and object files on disk.

NOTE:
I hereby release this program to the public domain and give Micro Cornucopia the right to print the listing.

Signature ________________________________

Name ________________________________

Address ________________________________

City ______________ State __________ Zip __________

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

(continued from page 5)

The workmanship looked solid, and after a few minor problems, I got the disk formatted and everything running smoothly. Then I decided to test the keyboard lock on the front panel. I turned the keyboard lock to disconnect the keyboard, and the machine died. It turns out that the switch had been miswired to short the power supply. I returned it to Compuadd for repair and was surprised to get it back just four days later.

Now the story gets interesting. The new unit had a new power supply which consistently died after several hours. I sent it back. Same fast turnaround, but they sent me a totally new system. This one ran fine when it was warm, but the hard disk was dead. When I sent this second unit back, I made one of the repair people promise to keep it on his desk until he was absolutely sure it was working. My latest machine works!

The downside of this story is obviously the questionable quality control. Another of Compuadd's problems is that they put too much emphasis on fast repair turnaround. The upside is their fast warranty repair, the inexpensive price, and the solid feel of the base unit and keyboard. Performance is impressive. The CMI disk is dependable, so far, although I've logged only about 40 hours on it.

W.Y. Crutchfield
5503 Yarwell
Houston TX 77096

Back To The Future?

Thanks for nothing, John P. Jones! Three days ago my Kaypro locked up. Nothing short of unplugging it helped. Even when I plugged it back in, nothing but an unblinking cursor appeared on the screen. Until yesterday I thought it was dead; now I don't know.

Then I received Micro C issue #29 — and a package from Berlund Extranational. I had never heard of them, but I now suspect that my computer has been running Chrono Pascal for some time. Careful excavation of my desk revealed mysterious printouts at a strata that indicates the work was probably conducted four to six weeks ago. Underneath the printouts I found handwritten notes (my handwriting) on the niceties of Chrono Pascal syntax and error checking. Spooky! There's no telling how much software might be in progress, or even completed, on my disk. The computer, of course, is locked up.

I phoned Berlund to see what they could suggest. They answered before the phone rang and refused to talk to me "again".

The article mentioned that Berlund's mail order processing runs under Chrono Pascal, but if my experience is indicative they are shipping products as much as six weeks before an order is received, not two days. This is unethical, arrogant, and, in my case, disastrous. My heavily modified Kaypro 4-84 has obviously gagged on some hardware-specific code in the Berlund compiler. Now I'm contemplating a "processor lobotomy" to get my machine back.

Darrell Bethune
Box 1632
Fairview, AB Canada T0H 1L0

Stuck At 200 Points

About a year ago I bought a copy of your disk K4, the 550 point Adventure game. I've enjoyed playing it, but I need some hints.

Specifically, I need help getting by the ogre near the golden eggs, crossing the reservoir north of the mirror canyon, opening the safe below the hall of the Mountain King, understanding the clue about "stand where the statue gazes and use the right tool," and finding the pirate's chest in the maze. I have found the vending machine maze and have mapped the maze south of the west mist hall, assuming it to be the pirate's, without locating the chest. Is the pirate's a third maze? HELP!

Thomas L. Adams
SAPL Pgm Ctl
POB 30167
Yanbu Al Sinaiyah
Saudi Arabia

■ ■ ■

ADVERTISER'S INDEX

THE LAST PAGE

By Gary Entsminger
1912 Haussler St.
Davis CA 95617

Various Video: Interrupt 16 On The PC

If I may interrupt this column for a second... Actually, I don't want to interrupt but you might. And if you like interruptions, read on.

You can improve (i.e., make more general) your MS-DOS application program by making it work in various video modes — for example, monochrome and color/graphics.

Memory Addresses & The Limbo

The monochrome and color graphics adapters have different memory addresses. If your program uses color, for example, it must send the color information to the c/g's memory which begins at physical location 0B8000H. The monochrome adapter's memory begins at address 0B0000H.

The point is — if you're sending information to the wrong memory location for the current adapter, your program will find itself in limbo (in other words, doing a Chubby Checker dance).

But (and here's the happy part), you can design your program to check the video mode before it takes a chance. If your program is really prepared, it will adjust to the current adapter, or exit gracefully — far better alternatives to the limbo.

Video Modes

On the PC, 12 video modes define the display characteristics corresponding to the capabilities of the various video adapters —

Modes 0-3 are text modes for the color/graphics adapter

Modes 4-6 are the graphics modes for the c/g

Mode 7 is for the monochrome adapter

Modes 13-16 are used by enhanced graphics adapter

These modes are controlled by the ROM-BIOS, and we can check or set them via ROM-BIOS interrupt 16 (10H). Service (or function) 15 checks and returns the current video mode. Service 0 sets the video mode.

At Interrupt 16

To check the current video mode —

```
MOV     AH,15
INT     16
```

On return —

AL = current display mode
AH = number of characters per line
BH = active display page

If your program isn't prepared for what it sees in register AL, it should branch appropriately — either to a subroutine designed for the particular adapter (which sets the mode before it writes to memory), or exit.

For example —

```
MOV     AH,0     ; for service 0
MOV     AL,1     ; for mode
INT     16
```

will initialize the c/g adapter for a 40 by 25 color image.

More Interrupt 16

Interrupt 16 offers 16 main services for the PC. Each is selected by placing the service number in the AH register. Some services require other information which your program places in registers BX, CX, or DX before calling interrupt 16.

For example, if you want to set the cursor size —

```
MOV     AH,1     ; for service 1
MOV     CH,12    ; for the top of
                 ; the scan line
MOV     CL,13    ; for the bottom
                 ; of the scan line
INT     16
```

to get the default underline for a monochrome adapter.

If you want to set a cursor position —

```
MOV     AH,2     ; for service 2
MOV     DH,5     ; for row 5
MOV     DL,1     ; for column 1
MOV     BH,0     ; for graphic mode
INT     16
```

defines a cursor on the c/g adapter at (2,5).

Here's a complete list of interrupt 16's services (in decimal) —

0 = Set video mode
1 = Set cursor size
2 = Set cursor position
3 = Read cursor position
4 = Read light-pen position
5 = Set active display page
6 = Scroll window up
7 = Scroll window down
8 = Read character & attribute
9 = Write character & attribute
10 = Write character
11 = Set color palette
12 = Write pixel
13 = Read pixel
14 = Write character as TTY
15 = Get current video mode
19 = Write character string

The Beauty Of Interrupts

One of the beauties of the MS-DOS operating system is the range and ease of use of its interrupts. Once you get the hang of calling the various services an interrupt offers, 8088 (8086, etc.) assembly language becomes not only an amazingly powerful but also a very easy-to-use programming language.

And much can be learned from the study of these versatile ROM-BIOS services.

If you want an in-depth look at assembly language programming for the PC (including excellent studies of ROM-BIOS interrupts), I suggest —

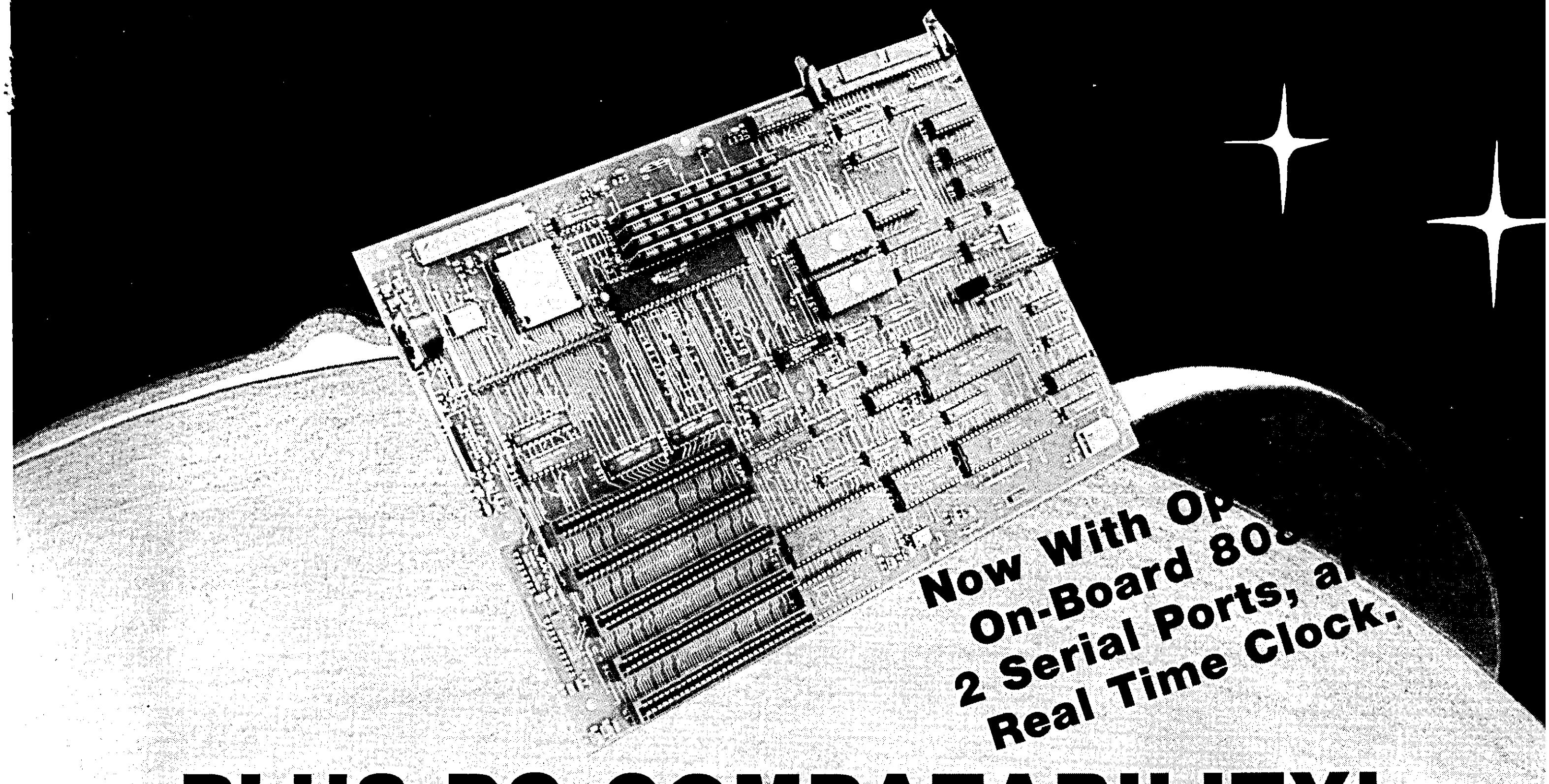
"8088 Assembler Language Programming: The IBM PC" by Willen and Krantz

"Programmer's Guide To The IBM PC" by Peter Norton

There's beauty in both of these, and they work perfectly (in my programming experience at least) on generic PCs, including the X16.

NEW LOWER $

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



PLUS PC COMPATABILITY!

Features:

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

Designed, Manufactured and Serviced by:

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

COD, VISA, MASTER CARD, CERTIFIED FUNDS

Kits - All Wave Soldered!!

NEW LOWER PRICES

• Full Kit W/O SCSI, serial ports, clock		$ 550.00
• Assembled and Tested		
W/O SCSI, serial ports, clock	1-4	$ 599.00
	5-9	$ 525.00
SCSI Port	add	$ 20.00
Serial Ports	each - add	$ 12.50
Real Time Clock and Battery	add	$ 12.50
• ½ MEG Add-On		$ 105.00
• XT Style, Enclosure		$ 75.00
• 150 Watt Power Supply, Tested		$ 105.00
• 8087-2 Coprocessor		$ Call

NEW

Total Solutions:

Assembled and Tested Systems

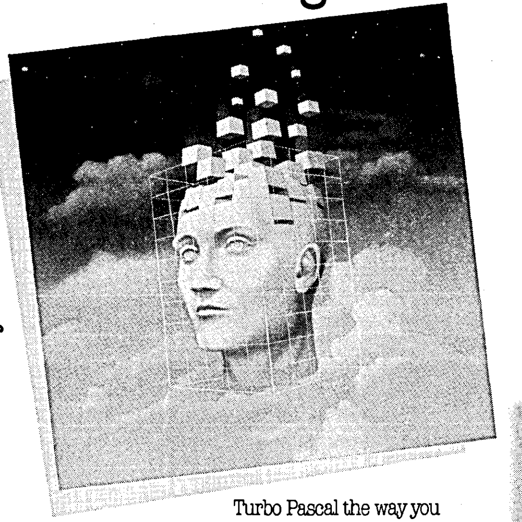
- Mono-Chrome System $1,499.00
  Includes: 512K, Power Supply, Enclosure, Keyboard, Monitor, Two Drives, and Printer Port, 1 Serial Port, Real Time Clock.
- Quantity Discounts Upon Request
- Call For Info on Add-Ons, Hard Disks, Etc.

Step-by-step tutorial, demo programs with source code included!

Borland's new Turbo Prolog is the powerful, completely natural introduction to Artificial Intelligence

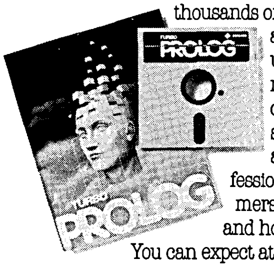
Prolog is probably one of the most powerful computer programming languages ever conceived, which is why we've made it our *second* language—and "turbocharged" it to create Turbo Prolog.™

Our new Turbo Prolog, the natural language of **A**rtificial **I**ntelligence, brings supercomputer power to your IBM® PC and introduces you step-by-step to the fascinating new world of **A**rtificial **I**ntelligence. And does all this for an astounding $99.95.



Turbo Prolog is to Prolog what Turbo Pascal® is to Pascal!

Our Turbo Pascal astonished everyone who thought of Pascal as "just another language." We changed all that—and now Turbo Pascal is the de facto worldwide standard, with hundreds of thousands of enthusiasts and users in universities, research centers, schools, and with professional programmers, students, and hobbyists.



You can expect at least the same impact from Turbo Prolog, because while Turbo Prolog is the most revolutionary and natural programming language, it is also a complete development environment—just like Turbo Pascal.

"Turbo Prolog offers generally the fastest and most approachable implementation of Prolog.

Darryl Rubin, AI Expert "

Even if you've never programmed before, our free tutorial will get you started right away

You'll get started right away because we have included a complete step-by-step tutorial as part of the 200-page Turbo Prolog Reference Manual. Our tutorial will take you by the hand and teach you everything you're likely to need to know about Turbo Prolog and artificial intelligence.

For example: once you've completed the tutorial, you'll be able to design your own expert systems utilizing Turbo Prolog's powerful problem-solving capabilities.

Think of Turbo Prolog as a high-speed electronic detective. First you feed it information and teach it rules. Then Turbo Prolog "thinks" the problem through and comes up with all the reasonable answers—almost instantly.

If you think that this is amazing, you just need to remember that Turbo Prolog is a 5th-generation language—and the kind of language that 21st century computers will use routinely. In fact, you can compare Turbo Prolog to Turbo Pascal the way you could compare Turbo Pascal to machine language.

You get the complete Turbo Prolog programming system for only $99.95

You get a complete Turbo Prolog development system including:

- The lightning-fast Turbo Prolog incremental compiler and the interactive Turbo Prolog editor.
- The 200-page reference manual which includes the step-by-step Turbo Prolog tutorial.
- The *free* GeoBase™ natural query language database including commented source code on disk—ready to compile. GeoBase is a complete database designed and developed around U.S. geography. It includes cities, mountains, rivers, and highways, and comes complete with natural query language. Use GeoBase immediately "as is," or modify it to fit your own interests.

So don't delay—don't waste a second—get Turbo Prolog now. $99.95 is an amazingly small price to pay to become an immediate authority—an instant expert on artificial intelligence! The 21st century is only one phone call away.

Turbo Prolog 1.0 Technical Specifications

Programming System Features

☑ *Compiler:* Incremental compiler generating native in-line code and linkable object modules. The linking format includes a linker and is compatible with the PC-DOS linker. Large memory model support. Compiles over 2500 lines per minute on a standard IBM PC.

☑ *Interactive Editor:* The system includes a powerful interactive full-screen text editor. If the compiler detects an error, the editor automatically positions the cursor appropriately in the source code. At run-time, Turbo Prolog programs can call the editor, and view the running program's source code.

☑ *Type System:* A flexible object-oriented type system is supported.

☑ *Windowing Support:* The system supports both graphic and text windows.

☑ *Input/Output:* Full I/O facilities, including formatted I/O, streams, and random access files.

☑ *Numeric Ranges:* Integers: −32767 to 32767; Reals: 1E−307 to 1E+308

☑ *Debugging:* Complete built-in trace debugging capabilities allowing single stepping of programs.

YES! I want the best

Turbo Prolog at only:

$99.95

To order by phone,
or for a dealer nearest you,
Call (800) 255-8008
in CA call (800) 742-1133.

Send me ___ Turbo Prolog at $______
Outside USA add $10 per copy
CA and MA res. add applicable sales tax $______

Amount enclosed: $______

This price includes shipping to all US cities

Payment: VISA MC Bank Draft Check

Credit card expiration date: ___/___

Card # |_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|

†You must have an IBM or true compatible running DOS 2.0 or later.
My computer's name and model is:
____

The disk size I use is: □ 3½" □ 5¼"

NOT COPY PROTECTED
****60-DAY MONEY-BACK GUARANTEE***

Name: __________________________

Shipping Address: ______________________
____

City: ___________________________

State: ____________ *Zip:* ____________

Telephone: ________________________

CODs and purchase orders WILL NOT be accepted by Borland. Outside USA make payment by bank draft payable in US dollars drawn on a US bank.

**YES, if within 60 days of purchase this product does not perform in accordance with our claims, please call our customer service department and we will gladly arrange a refund.*

†Minimum system requirements:
IBM PC, XT, AT, PCjr,
and true compatibles;
384K RAM.

SU8

BORLAND
INTERNATIONAL

4585 SCOTTS VALLEY DRIVE
SCOTTS VALLEY, CA 95066
(408) 438-8400 TELEX: 172373

Vive la différence

Borland products include Turbo Prolog; Turbo Pascal; Turbo Tutor; Turbo Editor Toolbox; Turbo Database Toolbox; Turbo Graphix Toolbox; Turbo GameWorks; Turbo Lightning; Lightning Word Wizard; Reflex: The Analyst; Reflex Workshop; SideKick; SideKick, The Macintosh Office Manager; Traveling SideKick; and SuperKey—all of which are trademarks or registered trademarks of Borland International, Inc. or Borland/Analytica, Inc.

IBM, XT, AT, and PCjr are registered trademarks of International Business Machines Corp.