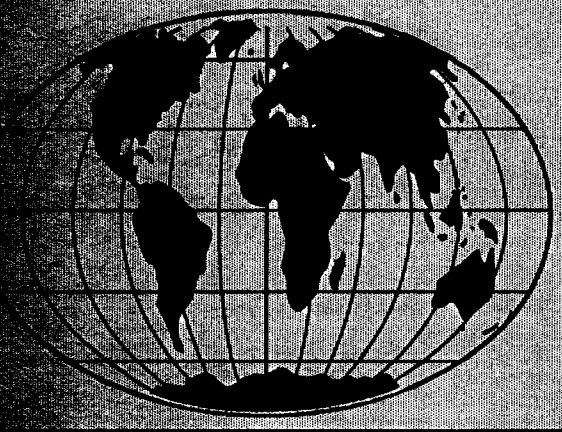
Providing Support Around The World



The Computer Journal

Issue Number 76 November/December 1995 US$4.00

Alternatives to the XT

Small System Support

PC/XT Corner

The European Beat

Real Computing

PC Security System

Dr. S-100

PC Time Clock

Floppy Disk Problems

Centerfold - Jade Bus Probe

The Computer Corner

ISSN # 0748-9331

TCJ - For Having Fun With Old Computers!

Cross-Assemblers as low as $50.00
Simulators as low as $100.00
Cross-Disassemblers as low as $100.00
Developer Packages
as low as $200.00(a $50.00 Savings)

A New Project
Our line of macro Cross-assemblers are easy to use and full featured, including conditional assembly and unlimited include files.

Get It To Market--FAST
Don't wait until the hardware is finished to debug your software. Our Simulators can test your program logic before the hardware is built.

No Source!
A minor glitch has shown up in the firmware, and you can't find the original source program. Our line of disassemblers can help you re-create the original assembly language source.

Set To Go
Buy our developer package and the next time your boss says "Get to work.", you'll be ready for anything.

Quality Solutions
PseudoCorp has been providing quality solutions for microprocessor problems since 1985.

BROAD RANGE OF SUPPORT

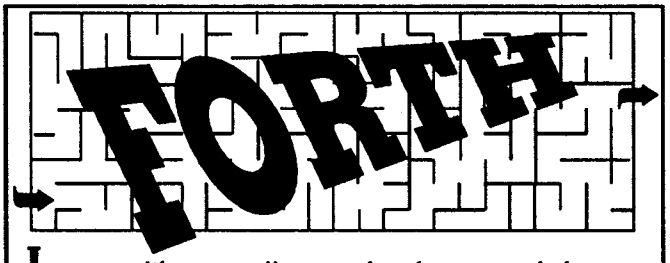
- Currently we support the following microprocessor families (with more in development):

Intel 8048	RCA 1802,05	Intel 8051	Intel 8096
Motorola 6800	Motorola 6801	Motorola 68HC11	Motorola 6805
Hitachi 6301	Motorola 6809	MOS Tech 6502	WDC 65C02
Rockwell 65C02	Intel 8080,85	Zilog Z80	NSC 800
Hitachi HD64180	Motorola 68000,8	Motorola 68010	Intel 80C196

- All products require an IBM PC or compatible.

So What Are You Waiting For? Call us:

PseudoCorp
Professional Development Products Group
921 Country Club Road, Suite 200
Eugene, OR 97401
(503) 683-9173 FAX: (503) 683-9186 BBS: (503) 683-9076



Journey with us to discover the shortest path between programming problems and efficient solutions.

The Forth programming language is a model of simplicity: In about 16K, it can offer a complete development system in terms of compiler, editor, and assembler, as well as an interpretive mode to enhance debugging, profiling, and tracing.

As an "open" language, Forth lets you build new control-flow structures, and other compiler-oriented extensions that closed languages do not.

Forth Dimensions is the magazine to help you along this journey. It is one of the benefits you receive as a member of the non-profit Forth Interest Group (FIG). Local chapters, the GEnie™ Forth RoundTable, and annual FORML conferences are also supported by FIG. To receive a mail-order catalog of Forth literature and disks, call 510-89-FORTH or write to:
Forth Interest Group, P.O. Box 2154, Oakland, CA 94621.
Membership dues begin at $40 for the U.S.A. and Canada.
Student rates begin at $18 (with valid student I.D.).

GEnie is a trademark of General Electric.

Kibler Electronics

Serving the
Industrial Electronics Community
since 1978

Specializing In
Hardware Design and
Software Programming

Previous Projects include:
PLC ladder programming (15,000 lines)
8051 Remote I/O using MODBUS
6805 Instrumentation Controller
68000 Real Time Embedded Operations
NETBIOS programming and Debugging
Forth Projects and Development
HTML Design and programming
Articles, Training, and Documentation

Bill Kibler
Kibler Electronics
P.O. Box 535
Lincoln, CA 95648-0535
(916) 645-1670

e-mail: kibler@psyber.com
http://www.psyber.com/~kibler

SAGE MICROSYSTEMS EAST

Selling and Supporting the Best in 8-Bit Software

Z3PLUS or NZCOM (now only $20 each)
ZSDOS/ZDDOS date stamping BDOS ($30)
ZCPR34 source code ($15)
BackGrounder-II ($20)
ZMATE text editor ($20)
BDS C for Z-system (only $30)
DSD: Dynamic Screen Debugger ($50)
4DOS "zsystem" for MSDOS ($65)
ZMAC macro-assembler ($45 with printed manual)

Kaypro DSDD and MSDOS 360K FORMATS ONLY
Order by phone, mail, or modem and use Check, VISA, or MasterCard. Please include $3.00 shipping and Handling for each order.

Sage Microsystems East
1435 Centre Street
Newton Centre MA 02159-2469
(617) 965-3552 (voice 7PM to 11PM)
(617) 965-7259 BBS

The Computer Journal

Founder
Art Carlson

Editor/Publisher
Bill D. Kibler
Dave Baldwin

Technical Consultant
Chris McEwen

Contributing Editors
Herb Johnson
Charles Stafford
Brad Rodriguez
Ronald W. Anderson
Helmut Jungkunz
Ron Mitchell
Frank Sergeant
JW Weaver
Richard Rodman
Jay Sage
Tilmann Reh

The Computer Journal is published six times a year and mailed from *The Computer Journal*, P. O. Box 3900, Citrus Heights, CA 95611, (916) 722-4970.

Opinions expressed in *The Computer Journal* are those of the respective authors and do not necessarily reflect those of the editorial staff or publisher.

Entire contents copyright © 1995 by *The Computer Journal* and respective authors. All rights reserved. Reproduction in any form prohibited without express written permission of the publisher.

Subscription rates within the US: $24 one year (6 issues), $44 two years (12 issues). Send subscription, renewals, address changes, or advertising inquires to: *The Computer Journal*, P.O. Box 3900, Citrus Heights,CA 95611.

Registered Trademarks

It is easy to get in the habit of using company trademarks as generic terms, but these trademarks are the property of the respective companies. It is important to acknowledge these trademarks as their property to avoid their losing the rights and the term becoming public property. The following frequently used trademarks are acknowledged, and we apologize for any we have overlooked.

Apple II, II+, IIc, IIe, Lisa, Macintosh, ProDos; Apple Computer Company. CP/M, DDT, ASM, STAT, PIP; Digital Research. DateStamper, BackGrounder ii, Dos Disk; Plu*Perfect Systems. Clipper, Nantucket; Nantucket, Inc. dBase, dBASE II, dBASE III, dBASE III Plus, dBASE IV; Ashton-Tate, Inc. MBASIC, MS-DOS, Windows, Word; MicroSoft. WordStar; MicroPro International. IBM-PC, XT, and AT, PC-DOS; IBM Corporation. Z80, Z280; Zilog Corporation. Turbo Pascal, Turbo C, Paradox; Borland International. HD64180; Hitachi America, Ltd. SB180; Micromint, Inc.

Where these and other terms are used in *The Computer Journal*, they are acknowledged to be the property of the respective companies even if not specifically acknowledged in each occurrence.

TCJ The Computer Journal
Issue Number 76 November/December 1995

Editor's Comments 2

Reader to Reader 3

Real Computing 7
MINIX and more.
By Rick Rodman.

PC/XT Corner 10
Article by Kirk Lawrence,"Super-Charge that old XT!..
By Frank Sergeant.

The European Beat 15
10 years of support.
By Helmut Jungkunz.

Alternatives to the XT 18
Options and the PT68K-4.
By Bill Kibler.

Dr. S-100 22
GIDE and Jade Bus Probe.
By Herb Johnson.

Center Fold 25
Jade BUS Probe.

PC Time Clock 29
Improving Accuracy.
By Terry Hazen.

PC Security System 30
House security system in BASIC..
By Michael Krabach.

Small System Support 36
C and assembly language tutorial.
By Ronald W. Anderson.

Floppy Disk Problems 41
Trouble shooting floppy design problems.
By Claude Palm.

Support Groups for the Classics 46

The Computer Corner 50
By Bill Kibler.

EDITOR'S COMMENTS

Well here is issue 76 and my last as editor. Dave is almost stepping on my heals, trying to get a start at producing the next issue for you. He comments later on this page. Have no fear number 76 is good as usual and I am sure number 77 will be too, I know I will help make sure that comes true.

This issue is support of XT and PC systems and beyond. After letters to the editor, comes Rick Rodman with MINIX is alive and well? Can this be true, read and see for yourself.

Frank Sergeant has comments on XT/PC's and a little insert by Kirk Lawerance on supercharging XT's. Helmut Jungkunz is next with his report on 10 years of Amstrad support through their User Group. I follow with my shortened but interesting article on Alternatives to using XT/PCs. Herb Johnson gives his long awaited discussion of the Jade Bus Probe. I follow that with the schematics in the centerfold.

Using XT or AT's for guarding your house is next. Michael Krabach sent me this article some time ago and I have been trying to fit it in, so here it is now. Remember the entire code is on the *TCJ* BBS and home page. For those still learning assembly or C, Ronald Anderson keeps trying to make us experts, well at least able to program in those languages. Speaking of trying, Claude Palm sent me an article in hard copy which I finally got into computer form for you to read. It explains what can happen in designing systems.

Lastly is my column, The Computer Corner, where you will be seeing my words for many more issues to come. I don't plan to do any big articles right away, as I know I will be helping Dave get up to speed on producing the next few issues. So what does he have planned?

Dave's own words

Here we go again. *The Computer Journal* has a new editor, so the first question is 'Are there going to be changes?'. Of course there are. Big changes? Probably not. I'm going to 'play' around with the format and I'm talking to a couple of new people about becoming authors. We now has a BBS and a fax number and the Web page is attracting a lot of attention on the Internet. I've been updating the *TCJ* files and messages on Genie and posting some of the info files on CompuServe.

Of course, our current writers will still be here and some new ones. David McGlone of the *Z-Letter* has agreed to write an article about his experiences trying to get CP/M software and someone (I don't know who yet) is going to write a CP/M and Z-System column. Another regular item I'm going to try to get going is a 'Program This Chip' article series. I'm going to do one on the Z80 SIO which seems to give people fits.

I'm going to have as much fun as I can with *TCJ*. I'm going to try out some little changes here and there and see who notices. It's also going to be an 'educational' experience. Bill says that in six months or so, I'll have found out what I should have known to start with. What's Bill going to do? He's going to continue to write the Computer Corner and be the *TCJ* Technical Consultant. He lives right up the road from me and I have his home and work phone numbers, just in case.

Well, what's the 'new' *TCJ* going to be like? A lot like the 'old' one. I'll quote from our Web page on the Internet <"http://www.psyber.com/~tcj/"> (which I wrote):

The Computer Journal has articles and reviews on both new and old hardware and software. In the last year, there have been articles on most of the popular microcontrollers, reviews of a new Z180 system for CP/M, modifications for older systems, software articles and tutorials on Forth, 'C', and assembly language, and the 'Centerfold' schematics for older computer circuits.

In general, we cover hardware and software that one person can work with, where you can 'do it yourself'. This means boards and systems where you can identify (and get) the parts and get code to make it work. In particular, this means the Kaypro, S-100 boards, Z80/180/280 and CP/M systems, microcontrollers like the 8048, 8051, and 68HC11, and software articles that include source code. This is also why we started covering the PC-XT clones made with identifiable TTL parts. BIOS code is also available for them now so you can make them do what you want.

On the other hand, if a project or system requires an engineering team or access to custom IC's, you probably won't read about it in *TCJ*. The exception to that is when our authors and/or readers get together for a project and can provide all the necessary resources.

TCJ is subscriber supported rather than advertiser supported. Your suggestions, requests, and complaints are listened to and acted on when we can. You have my name, number, address, email address, BBS number, fax number. If you never suggest, request, or complain, I'm going to assume you either love what I'm doing or you're not paying attention.

Thanks Dave and BYE! Bill Kibler.

(800) 424-8825 or (916) 722-4970
Email: tcj@psyber.com
BBS: (916) 722-5799
FAX: (916) 722-7480
Dave Baldwin: dibald@netcom.com
Bill Kibler: kibler@psyber.com
Home page "http://www.psyber.com/~tcj"

READER to READER

Letters to the Editor
All Readers
MINI Articles

Dear Bill:
I have never had any kind of EPROM programmer, so a number of projects have been started but not completed. This summer I was about to begin again, starting to build a Motorola 6805 design-kit that copied an EPROM into internal memory. I have been planning to build a programmer by myself using switches on the address bus and data buses, with a binary and HEX display for both data and address using HP Hex displays from Hosfelt and blue discrete LEDs, but made my decision to buy Needham's Programmer and clean house for as I said, I have stacked up a number of projects and the price was only a little more than Motorola's Design Kit. Sorry Motorola, but maybe if your Kit came as a bare board with ICs I might have sprung for it, and still might as money becomes available. Fortunately, I have been aching to make some sort of IBM Front Panel with LEDs and switches, so the planned Eprom Programmer parts list was easily changed into a new project. There is no soldering to do with Needham's PB-10, it is complete with 5.25" 1.2M floppy.

What Comes With The PB-10?

The PB-10 is an 8-bit ISA card to plug inside the computer, with a ribbon cable(supplied) that leads out the back of the computer to your Programming Socket that you get to mount where you want, the cable is about long enough. Mounting tape for the socket is supplied, and it holds the socket well but can be moved if need be. A 5.25" 1.2M diskette holds the "EMP" program for the PB-10: the same software they use with their other programmer, the more expensive stand-alone.

First, a person ought to back up their new diskette. I think DISKCOPY is the easiest way to do that. Then INSTALL the program on your HardDisk, mine is on C: and I had no problem. I suppose it could be INSTALLed to a 360k diskette in the same way, for smaller computers. I had wanted to install the whole thing in my XT(XtraTiny) Magnum until I found out the PB-10 can handle *Big* Eproms on the order of 27010, 27020 and 27040. Being able to program these big EPROMs made the decision to put the card into my x486 with 8M clear: that's where it is now. The Magnum will continue to serve as my test bed for plug-in home brew projects.

Some EPROMs and the 8748 and the 8752 types are programmed with the purchase of an adapter. Not immediately possible, but not impossible. I'd like to build a kit to do those devices, but that isn't how its done. Just soldering one socket and one plug on a tiny pcboard sounds like fun. Volunteers to design adapter kits? Another difficulty lies in the big EPROMs, the 27040 needs special code modifications to the EMP program, they must supply it with the adapter I bet.

That's all I have for the PB-10 Programmer by Needham's for only $139.95 from Digi-Key. What have I done with it? I have unplugged the 2764 EPROMs from the Magnum, the only ROM inside, and read it into the buffer then saved the buffer to a file and begun to debug it to learn how a computer boots. I have reloaded the file to the buffer and programmed a blank 2764 which I've plugged back into the Magnum and it works, testing the whole thing from read to program. I can easily buy 2764s by the tube of 13 in Portland, and this was one way to test what I did with it. The Magnum booted, the Eprom passed the test. This is a work in progress.
Regards,Marshall Montchalin <Obiwan@habit.com>

Dear Bill:
I bought a ten-key computer keyboard from St Vincent's for only $2 and since it's connector wasn't standard, I pulled out my Philip' and opened it up. I found a proto-pack with a 2716 plugged in!

If the keyboard works, it could be a good protopack to desolder for other purposes or it could be a keyboard. Having the memory already wired for you adds a degree of confidence and ease to any project.

Mostek Mk 38P73/D2 D

What is it? (and I want to program it and use it, ha-ha; I saw them introduce the F8 back in the beginning at The Jantzen Beach Red Lion Thunderbird and fell in love with the Proto-Pack version they showed flash LEDs or something. It was also at the WesCon in Las Vegas.) The keyboard was factory dated August 1983 but my most recent Fairchild catalog is January of 1978. So it's an F8 between an F3872 and an F3874 in the MicroMachine Series shown in my catalog.

I read the F3870 is 2k ROM, F38E70 2k PROM, the F3872 is 3k ROM/64 bytes RAM, the F3874 is 4k ROM.

Therefore the 38P73 must be 3.5k I *wonder* : A close inspection of the ProtoPak reveals the holes for more machine-pin socket pins above the 2716 plugged into it, as if *bigger* JEDEC 28-pin memories could be plugged in com-

patibly, had there been four more pins installed, making it easy for the foundry to add/subtract features offered, and save them money on one style of ceramic. Surely I am dreaming, but maybe they added their options when soldering pins to the dip. What other F8 ProtoPaks did they make?

So here I am, staring at this technological marvel. If it works, I could just hook it up to the S/M or my 486 and not desolder any protopack. Until I find the right books...

Right now its untested, untraced. Got any good stories to tell about Mostek or Fairchild? Why didn't their style cpu make the big time? I'd bet there's a few good tales to hear. In this catalog Fairchild has both the F8 and 6800 listed. Now, that doesn't sound right, does it?

All for now, Marshall

Thanks for both messages Marshall. Well it sound like you are about to have plenty of fun learning to use your new programmer. I have been using Needhams for many years and since they are just across town from us (and past advertisers) we hope to see them back in TCJ pages soon.

The F8 never really took off and I think Fairchild was just a secondary chip producer in the case of the 6800. Motorola had plenty of agreements with other makers, that way people would use the chip knowing the could buy it from more than one source.

The chip carrier idea never caught on. The idea allowed you to prototype an actual system using a separate ROM that later would be burned into the single chip. I think this was before EPROMs where built into CPU's, or at least when those chips were $100 in price and the F8 and EPROM could be had for less than $30. The idea died of course as soon as the other chips came down in price.

Well that is all I know about those items, but if you learn more, we would love to have some articles telling more about them old days... Bill.

Dear Editor,
I am looking for the following Fairchild books (original or photocopied).

F8 User's Guide
F8 Microprocessor Family Data book
F8 Guide to Programming
FairBUG User's Guide
OCM-1 User's Manual

I urgently need then to repair and reforming of U.S. made microprocessor controlled radio sets. Unfortunately F8 family itself and F3850 family ICs were used very rare here in Japan, so these books are absolutely unobtainable at present. To my surprise that the library of Diet of Japan has no information on those titles.

I also looking for some ICs, F3850 (MK3850), F3853 (MK3853), 11C90 and uA757, If you have any information on above items would you please advise me where can I purchase them?

Looking further to hearing from you.
Sincerely, Kato Higuchi,
1-16-8 Minami-Senzoku,
Ota-ku, Toyoko 145, Japan

Well Kato, sorry your note to me fell so far down in the pile that I only found it now. If you have internet access you might try contacting Marshall above, it seems you both have things in common. I remember now that MK stands for Mostek and I am not sure but think that Mostek designed the F8, but it could be the other way around.

Any other readers out there who can help Kato and Marshall get straight on what these chips do.? How about the documentation? Thanks for writing Kato. Bill.

TCJ: Your advertisement came to my attention. Your magazine appears to be potentially helpful to me in maintaining my PC/XT computer system.

I am presently trying to add a 3.5" floppy drive with PC-DOS 3.2 to help me to format 3.5" diskettes to 720K. For some unknown reason, I either get a 360K format or an error message.

Very truly yours, Keith E. Watkins, Schenectady, NY.

Well Keith, I see you need a little help understanding the drive arrangement with PC's. First off you might want to buy one of the XT-AT Handbooks sold by Annabooks (800)462-1042. They run about $9 and are often given away at conventions. On page 86 of mine is tables of floppy disk formats. I see that they were first supported by DOS 3.2, earlier DOS's would not handle anything other than 360K drives.

The table also says you need to use the /F:720 as part of the format command line. The DRIVER.SYS switch is /F:2 as in add this line to your CONFIG.SYS file "DEVICE=\dos\DRIVER.SYS /D:1 /F:2" and you should then be able to properly access the drives. You will get a message that tells you which drive letter to use for the new drive. The drive is defined by the /D:1, 1 being the second physical drive or B:. The drive gets redefined so to speak, to talk both the new disk parameters and the old ones. By that I mean, put a floppy in the B: drive and format it (format B:), it will be 360K. Reformat it now as the new drive (the prompt displayed a "D") (format d:) and it will be 720K.

Any DOS manual, from 3.2 on will have a section on CONFIG.SYS parameters and explain the above again with more samples. I think in version 3.2 config came into existence. Before 3.2 you were pretty much stuck with BIOS defined drive support. After 3.2 you could then reassign, create logical drives, add ramdisks, and many other functions through the config.sys options.

So thanks for asking Keith and hope you get a better start on upgrading your system. Bill.

To: *TCJ*: Enclosed is my check for a 2 year subscription. I've been looking for a source like this for data on the PC/XT's for a long time. My current project is a SCSI device for ATE work working with DTC 3250C controller, if it works out I'll try to cobble up a decent article for publication.

Disregarding my 486 for the moment, I am using a LASER XTSL, NEC V30, 8087, 640K, and an INTEL ABOVE BOARD. I discovered the motherboard had some built in options that I have enabled, also it has DIP locations for what appears to be 2 MEG of Expanded Memory, if I can get my hand on the necessary driver software.

With your PC/XT owner's readership, have you accumulated any kind of database on the configurations and options of versions of the PC/XT and clones, I'm interested in sources for hardware and software and possibly corresponding with fellow owner's of the LASER series. Are there any plans for a BBS in the future.

I program in Qbasic and MASM, I'm working on C on my way to C++, I'm interested mostly in hardware, I've got a MOD-EMUP programmer and I love to tinker if I can help anybody out please drop me a line.

Thomas Rumpf, 1st Floor,
1621 W. South St.
Allentown, PA 18102.

Thanks Thomas for your renewal and offer to write an article. Well we certainly could use more hands on information on SCSI as it pertains to PC/XTs. A friend just dropped by asking about his SCSI problem. Seems all his software is set up for certain drives and he just went to IDE drive from MFM. All is fine until he puts the SCSI drive in to. Then the drive letters get all out of whack and his software will not work.

I told him how most BIOS code checks for drives, then checks for logical drives. That means if you have only one drive and set it up as three, they will be C, D, E. Add a second physical drive and it will become D, pushing the two old drives out to E and F. I remember coming across this problem and using Disk Manager to load drivers that swapped drives around to be in proper order.

Now in your case, it sounds like you need to just add HIMEM to your config.sys file. High memory is suppose to be the drivers to use the expanded or extended memory in your motherboard. That assumes that it was built using one of the memory management standards which the software knows about. If it doesn't follow the standards, then we need to find other LASER users for you. But try the HIMEM switches, you will find them explained in any DOS manual. If I remember right LASER is also called Acer machines and "DEVICE=HIMEM.SYS / machine:acer1100 ..." might work.

We are interested in getting resource listings and group support indexes setup, but for now, we do it mostly on the TCJ BBS or WEB pages. Just stop by either and we will try and get you in contact with other users. Since both resources are new to TCJ, it may take some time to get more readers corresponding by these methods. So thanks and keep watching TCJ pages for more XT support. Bill.

Dear Chuck,
I recently purchased a KAYPRO II computer (sometimes referred to as Darth Vader's lunch box). I think it is probably the original "portable" or laptop version of computers. Anyway what I need is a source for the software which brings the beast to life. The manual indicates that it originally came with a package of 10 diskettes but the only one I really need is the basic one which brings it to life and makes it ready to receive other software. I think this same disk probably holds the utility software for the system. I am quite sure the machine operates basically in CP/M. I already have a full set of software formatted for the HP 125. However, I am told that Kaypro software was very proprietary and will only operate on the one system for which it was designed. If I can obtain the software I'll try and have fun with the beast, otherwise I guess it needs to go to a museum or be converted to a very large paper weight.

Thanks in advance for any assistance you may be able to provide.

Sincerely, Jerry D. McAtee, Rapid City, SD.

Well thanks for this letter Jerry, and hopefully Chuck has helped you out by now. For our readers, you are a typical inquiry that we get here at TCJ. First off Kaypro were not the first lunch boxes made, I think Osbornes get that honor. The Kaypro was certainly a very popular system and they were made for a very long time. You will see them often and there are several places to get software for them. I usually send people to Lambda Software in Oregon, his number is on the back page.

On operating systems, CP/M is NOT proprietary in any real sense. Kaypro's use CP/M, considered the first PC Operating system and mother to PCDOS. What makes systems non-compatible is how they talked to keyboards and video screens. CP/M has the BIOS being part of the operating system and is loaded from the boot disk. Later PC/XT machines, separated out the BIOS, placing it on the motherboard in ROM. Unlike CP/M where you must boot from a disk set up by the manufacturer of the machine (to get the correct BIOS code), PCDOS knows nothing of hardware, other than where and how to call routines in the ROM BIOS that DOES know about the hardware.

Another problem was early vendors could not decided on a standard disk format. Each company found a way to setup their disk and ideally give them some advantage over the competition. The opposite happened, people just went to the PC/XT when they came out, since IBM set standards. You can get for your "BOX" a program that can read many different disk formats. That way can copy programs from your HP125 to Kaypro disks and use them.

Using the Kaypro should be fun, once you get past that initial learning curve. Later you might even want to consider ZCPR, which can give you features still not found in some version of PCDOS.

Thanks for the request Jerry. Bill.

From: PhilMarkDavis@medio.net
Mr. Kibler:
Up here in Seattle, we have been very pleased to be able to receive and read the publication *TCJ*. Our group is PNHUG (Pacific Northwest Heath Users' Group).

And now you've made everything even better with the new web page and your new e-mail address. The page is simply great and with so much reference information plus even a reference to our little group, as is in the magazine. The Tilmann Reh e-mail address in Germany is new to me also. We're all interested in the IDE expansion project. We would be most happy if it all fit in, as a package, to the old Z-100 Heath/Zenith Computer. Maybe someday?

May I request an update to our PNHUG entry? You have our secretary-treasures old address which was P.O. Box 12214. He has a new one which is as follows:
Jim Moore,
1554 - 16th Avenue East,
Seattle, WA 98112-2807
e-mail: be483@scn.org

If you wish to publish, the other officers can be reached as follows:
Phil Mark Davis (Co-President),
3250 Portage Bay Place East,
Seattle, WA 98102-3893
e-mail: PhilMarkDavis@medio.net

Don Viele, Vice-President & BBS Sysop,
2732 S. W. 312th Place,
Federal Way, WA 98023-7811
e-mail: DonV820047@aol.com

Our BBS probably contains the now largest collection of PD and Shareware for the Heath/Zenith Z-100 computer. Nearly 15 MB! We have just started a long range project, in which several of the members, are going to organize a good collection of CP/M software and try to produce a CD-ROM of same. This may turn out to be only for our edification or may grow in other directions. This is a collaboration of one member who is the sysop of one of the oldest CP/M boards in the Seattle area, Mr. Ludovic Van Hemelryck of "Adam's RiBBS". The telephone number of his BBS is (206)481-1371 and that of PNHUG is (206)860-8472. It is our desire to eventually establish a web page and possibly a telnet link to our BBS. We are some way from this point at this time. Please feel free to publish any or all of the above information.

Thanks again for the great job and the reminder to renew. Will use your 800 number at the earliest time.

Sincerely,
Phil Mark Davis (Co-President)

Thanks Phil for the update. I am now going to work on a TCJ CD-ROM and am looking for your type of material to include. I want to save forever BIOS and Utility code for the older systems. I guess since the Z-100 is no longer supported, I wonder about problems publishing it's BIOS source code, got any ideas about that. I have Helmut's AMSTRAD programs and tons of old 8 inch CP/M disks to go through, but hope to have something by Spring of next year. Maybe we should work together? Thanks. Bill.

From: schang@huey.csun.edu
Subject: PC/XT: How to read ROM BIOS FDD info?
Hello! Just ran into your article about $10 PC/XT in the hopes of getting info on how to read FDD type info from XT's ROM BIOS. I've got MCT BIOS (c)1985 E.S. Oneal (whatever that means, attached on the BIOS chip). I tried Alt-Ctrl-S, Alt-Ctrl-Ins, and few other keys at random, including <Del> by itself, <F1>, etc. No luck so far. Is there any way to view that info? I picked up a 486DLC board a couple days ago, put in the old MFM HDD control card. The HD is about 20 MB big, but can't figure out the type info like # heads, # cylinders, # sectors, #LZone, etc.

I hope you can help me. Thanks! SS

OK SS, here is the information, get one of the Annabooks XT-AT Handbooks, making sure it has the list of hard drives. They cost about $9, but watch out for the free ones, they do not have the full drive tables. There is also a hard drive information text file on many BBS's that will most likely have your drive in it.

What you are trying to do by hitting <DEL> during boot up, only works with AT type machines that have a battery backed up drive and other information option. The XT's drive information is fixed in the BIOS, although most MFM controllers saved the information on the hard drive itself and changing controllers often means reformatting the drive and losing all the old data. Another problem is the BIOS sounds like it was modified or custom made and as such, I have no idea what it may do for you. IBM produced books on their machines, which contained the source of their BIOS. Look for these at swap meets, they are great for knowing what goes on inside the machines.

Look on the side of the hard drive at the model number and then look that up in hard drive table for values. If you can't find a book, e-mail me and I will look it up. Thanks. Bill.

Subject: DEC Rainbow and other
From: fritz_chwolka@AC.CyberCity.de (FRITZ CHWOLKA)
Hi....
I just got the last issue and it's funny. As you know I'm collecting old systems (not real, I don't throw them away) and now I get an DEC Rainbow with a segate ST506. The system run but I miss the installation disk for the Harddisk. I've some other ST506 and want to test them for replacing the original if it's necessary.

So I need information, software etc. for the Rainbow. Another problem are my Zilog Systems MCZ-10/15. I've got the Zilog- system disks for them (Hard sectored) but no CP/M disk. The manuals are complete. I've a nice Zilog MCZ 1/05, a multiuser system but without some boot disks it wont run. Maybe there in US is someone who can help. I need all about the MCZ-1/05 because I've nothing for it.

I've a NASCOM Computer. This is a german product an sold in similar ways. As a complete system or a kit. Never the less I've got no system disks for it . Maybe that someone has these disks and can make a copy for me. So this are my wishes and the other systems I have run well. Thanks for *TCJ* and it's a nice picture on page 51.

Greetings from Germany..Fritz Chwolka / collecting old computers just for fun.

Contact Rick Rodman, Thanks. Bill.

Real Computing

By Rick Rodman

32-Bit Systems

All Readers

Rick Moved

Minix is Back

The immediate question is, is Minix 1.7 a contender? I think so. Although it's acquired lots of new features and lots more utilities, it's still fairly simple at its core. For the intrepid Real Programmer who wants to understand what's under the hood, Minix 1.7 is definitely worth considering.

I loaded the Minix-86 1.7 beta on an ISA-bus 386SX machine with 2MB of RAM and an 80MB MFM hard drive. Notice that I used the "86" version. Minix 1.7 comes in an "86" version, for 8088s and 286s, and a "386" version for the bigger chips. I'm more interested in the "86" version, because it's smaller and simpler.

Minix 1.7 includes a driver for an Adaptec 154x SCSI host adapter, which is expensive but common, and, more significantly, TCP/IP networking. The networking support is for SMC/Western Digital WD8003 or 8013 boards, which are fairly common. If you've got a different board, well, you have all the source, write your own driver.

Every software review has to discuss the installation program, because that's one of the most important parts of any package. Minix 1.7's installation program is excellent. You log in as "root" - no "Geheim" any more - and type "instdist". From then on, it's smooth sailing.

Especially nice is the new "part" partitioning tool (compare to FDISK). Move to the first line, over to the "size" column, and type "m". Then type "w" and "q" and you're done with what used to be a daunting task.

That's really about all you have to do, besides putting in floppies. A "vol" program automates floppy changes. The "instdist" script doesn't install the source disks; you do that by entering just one command (admittedly cryptic, but that's Unix):

```
cd /usr ; vol 720 /dev/fd0 | uncompress | tar xvfp -
```

Source to the whole OS and most of the utilities is included. The old compiler is no longer used. Now, the ANSI C compiler, which also compiles Pascal and Modula-2 and used to be an extra-cost item, is supplied. Everything is written in ANSI C.

Other new things that are included include rz and sz (Zmodem), bawk, a new shell called "ash", and the m4 macro processor.

Glitches? Thus far in the beta, hard disk controller incompatibilities and booting seem to be the main problem areas. I couldn't get it to work on a PS/2 model 60. Unlike 1.5, it doesn't appear possible to run a system from floppies and RAM-disk alone. Of course, once you have it running on some machine, you could generate a floppy-only system or a driver for any disk controller you happen to have.

KA9Q TCP/IP

I've gotten numerous inquiries regarding using Tiny-TCP on various platforms, usually PCs. Now you can certainly do whatever you want with Tiny-TCP, but as for me, I intend to keep it very small and simple. For PCs, small and simple is not the way people like to do things.

The definitive TCP/IP on PCs is a package written by Phil Karn and others called by his call sign, KA9Q. Originally, this was a fairly simple package not too much bigger than Tiny-TCP, but it grew and grew through the years as more and more features accreted. Once upon a time, it actually compiled and ran under CP/M.

Several versions of KA9Q came out over the years. The oldest I have dates to 1987. The version I will describe here is from the Simtel CD-ROM and is dated June 1993.

Now, there are several ways to run TCP/IP on a PC. One way is to Start Up Windows 95. Another is to run Windows NT, OS/2 Warp Connect, Linux, or FreeBSD. If you have a Western Digital ethernet card, you could run Minix 1.7. You can also use Wolverine TCP/IP with Windows for Workgroups, or buy Chameleon, Acadia, Frontier, LAN Workplace, or one of many other fine Windows- and/or DOS-based TCP/IP packages.

But if you want to run TCP/IP on a PC for transferring files or Telnetting into other computers, and don't feel like buying an entire convenience store just to get a Twinkie, there's KA9Q.

So many people have asked me about it that it must be one of the Deep Dark Secrets of the PC World (like most free and/or good things). I've tried it out and it works just fine.

If you have the Simtel CD, you'll need E920603.ZIP from the \MSDOS\KAQTCPIP directory, and DRIVERS.ZIP from the \MSDOS\PKTDRVR directory. The manual is in MAN_9106.ZIP in the KAQTCPIP subdirectory. (If you don't have the Simtel CD, but you do have a

PC, go get it. It isn't worth **not** having.)

Unzip E920603.ZIP to obtain NET.EXE. The DRIVERS.ZIP is an archive of various network card drivers. To use SLIP or PPP on the serial port, you don't need any of them. I used a 3Com 3C523 board in an IBM PS/2-60 (286 machine), so I got 3C523.COM out of DRIVERS.ZIP.

The 3C523 driver is what's called a "Crynwyr Packet Driver". It's a TSR, like a Novell ODI driver. It sits in memory, and you use a software interrupt to talk to it. The software interrupt must be selected between hex 60 and hex 80. When you use this port from KA9Q, you specify the same software interrupt, and you have a connection. All of the packet drivers use the same method. Packet drivers are sometimes even supplied by card manufacturers with their cards. I don't think this is purely for KA9Q, however; I think some other NOSs (network operating systems) use this interface as well.

The following is how I run KA9Q with 3C523. The underlined text is what I type:

```
c:\ka9q>3C523 0x60 3 0x300 0xC000
c:\ka9q>net
net>hostname zinc
net>attach packet 0x60 eth0 8 1500
net>ifconfig eth0 ipaddress 129.212.32.8
net>ifconfig eth0 netmask 255.255.0.0
```

At this point I can telnet, ftp, ping, etc. To leave NET running and run a DOS program, the command "!" or "shell" can be entered. However, NET won't be able to do much while you're doing something else.

The first line above is the command to load the 3C523.COM packet driver, with 0x60 (hex 60 in C syntax) as its software interrupt, 3 as its hardware IRQ, hex 300 as its I/O address, and segment C000 as its memory address (0C0000H as seen from the Reference Diskette). The second line loads NET.EXE, which is referred to as the "NOS" in the documentation. The third line specifies the host name, AKA the machine's name; here at KPCF machines are named for chemical elements. The fourth line attaches the packet driver through the hex 60 software interrupt, calls it "eth0", and specifies a transmit queue of 8 packets with a maximum packet data size (MTU) of 1500 bytes. The latter two parameters are standard for Ethernet. The last two lines specify the IP address and network mask for the interface "eth0". By IP addressing rules, every interface has to have a different IP address.

By the way, we've talked a lot about TCP/IP in Real Computing. The IP addressing scheme is discussed in detail in *TCJ* #71. Detailed information on TCP, IP and FTP is also presented in *TCJ*s #64 and #66.

To run a SLIP or PPP connection with KA9Q, you don't need a packet driver, because there is an async driver built in to NET.EXE. Here's how you do that:

```
C:\KA9Q>net
net>hostname zinc
net>attach asy 0x3F8 4 slip sl0 1024 256 9600
net>ifconfig sl0 ipaddress 192.9.201.2
net>ifconfig sl0 netmask 255.255.255.0
```

The third line above is the configuration of COM1, as you'll recognize from the hex 3F8 I/O address and IRQ 4. The next word, "slip", can be either "slip" or "ppp", depending on what protocol you want to use; "sl0" is an arbitrary name for the interface; 1024 is the buffer size, 256 the MTU, and 9600 the baud rate.

KA9Q is loaded with features, as it ought to be considering that NET.EXE is almost 200K bytes in size. It supports FTP either as client or server, telnet, UDP, bootp and bootpd for dynamic address assignment, and on and on. You can put all the "net>" commands into a file called AUTOEXEC.NET to save all that mistake-prone typing every time you run NET.

When some folks think about networking, they immediately jump to mapping drives. In the TCP/IP world, this is usually done with NFS. KA9Q doesn't include NFS. Neither does Windows NT, for that matter. Neither does TinyTCP... so you aren't going to be able to see network drives on your Osborne. If this is what you want, you'll need to write an RSX of some kind. A network redirector, which is what we call the piece that makes the virtual drive, is heavily OS-dependent and complicated to write. Maybe someone will write one for the Z-System.

Other folks think of the World Wide Web. You'll need a little more than KA9Q or Tiny-TCP to get you on the Web... but not **much** more. There are text-only browsers, such as Lynx, with which we may be able to cobble together a CP/M or Minix Web browser.

On the other hand, suppose you want to write a program to transfer data over the TCP/IP network. The problem in the DOS environment is that there is no standard socket library interface. However, all the source is there on the Simtel CD. Take what you need and use it.

Building a LED panel for a PC

A *TCJ* reader named Marshall (AKA obi-wan of the Internet) has started construction of an LED panel along the lines of what I talked about in *TCJ* #70. He writes: "The front panel should display the ports used to boot the computer and the keyboard. The address bus could latch on every fourth clock or something, with the data bus. I think it should be binary, except the data bus should be hex... That makes a long row of address LEDs, 20+ and four hex data digits, 10 0-port switches and 10 1-port switches for 20 total switches, 16 port-1 LEDs, and 16 keyboard LEDs of two colors... 20 red address, 16 yellow, 16 blue, and 16 green LEDs would display well." Sounds really great!

Marshall accesses the Internet from a coffee shop in Portland, Oregon called "The Cafe@Habit.Com". He supplied a lot of interesting WWW addresses: http://www.digikey.com, same with marshall.com (no relation, I presume), cirrus.com, and cypress.com. Zilog has a book ordering page on theirs at http://www.zilog.com. Also, the following less obvious addresses are from Motorola and supply good information for hardware experimenters:

http://freeware.aus.sps.mot.com
http://Design-Net.com

Many modern PC BIOSs write "POST Codes" during boot up to an I/O port. I don't know what the port is; in fact, different manufacturers might use different ports. (Isn't it strange, everyone making identical boxes, yet each fanati-

cally guarding their identical secrets?) For interface debugging I think it'd be nice to be able to set the I/O port to be monitored by writing a port number to a latch. The address or data busses could be just two of many options.

Some PC manufacturers have recently taken timid steps toward more visually attractive machines, adding subtle curves or slightly less bland colors to their cheap-looking plastic cases. Also, accessories are available for PCs such as garishly-colored mouse pads and covers, "screen frames", and even horribly ugly keyboards in childish motifs - ghastly, but again with the same cheap plastic look. And they wonder why these changes make little difference.

Next time

Let's see, I was supposed to review FreeBSD, but I ran out of space and time. Oh, and Windows 95 came out. The view at KPCF? Other than the new user look & feel, it's just Windows for Workgroups and DOS 6 shrink-wrapped together. If you've got to use a Microsoft OS, take a look at Windows NT instead. It's a lot more stable, is really multitasking, and if you can't live without the new look & feel, you can get that too. Windows 95 is mostly 16-bit code. If you've got 32 bits, use 'em.

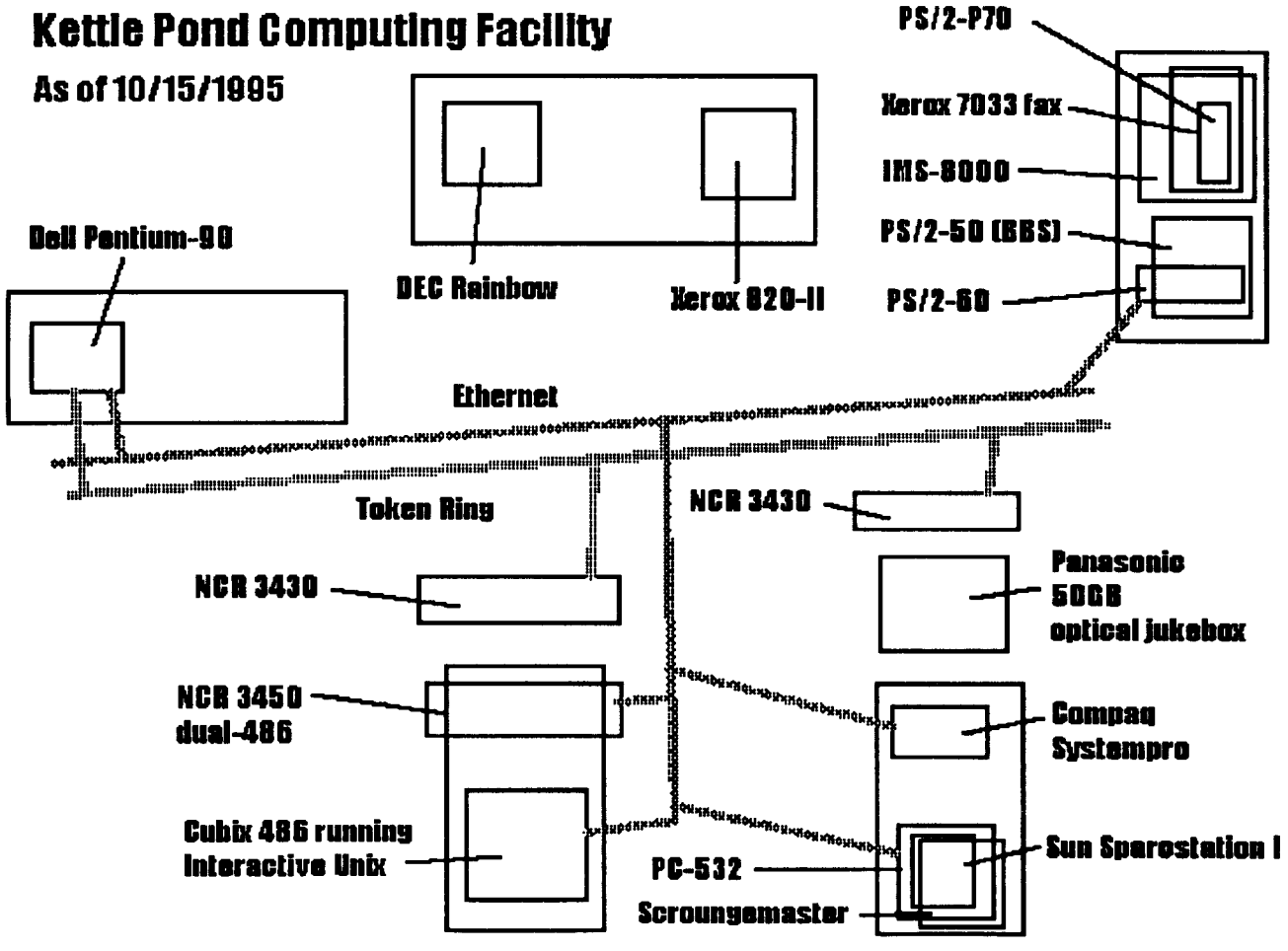
For more information

Real Computing BBS or Fax: +1 703 759 1169
E-mail: **Changing! To Be Determined**
Mail: 1150 Kettle Pond Lane, Great Falls VA 22066-1614

Walnut Creek CD-ROM (Manufacturer - Simtel/CPM/LINUX)
1547 Palos Verdes Mall Suite 260, Walnut Creek CA 94596
+1 510 676 0783

LINUX $49.95
Slackware Pro 2.3
Includes 4 CD-ROMs and a 450 page manual
A ready-to-run multitasking Unix clone for 386 and higher PC compatibles. TCP/IP, C, C++, X Window, complete Source Code, and much more!

JUST COMPUTERS!
(800) 800-1648
Fax (707) 586-5606 Int'l (707) 586-5600
P.O. Box 751414, Petaluma, CA 94975-1414
E-mail: sales@justcomp.com
Visa/MC/Int'l Orders Gladly Accepted
For catalog, send e-mail to: infor@justcomp.com
Include "help" on a single line in the message.



Regular Feature

Hardware Support

XT's and More

PC/XT Corner

By Frank Sergeant

Bank Switching on PCs

I thought I'd say a few more words about how PCs work. PCs from the beginning have had a Byzantine memory addressing system. It could be so simple! The actual address is just the collection of 0s and 1s on the physical address bus. A PC or XT has only 20 address lines. The 386/486/586, when in real mode, only uses the first 20 address lines. This gives a maximum physical address range of 2^20 = 1 MB (one megabyte). No matter how you slice it, 20-bits only let you address within the first megabyte of RAM. This is where all our troubles come from. Windows and DOS users get "out of memory" errors whether they have 1 MB or 4 MB or 8 MB of RAM. Without a great deal of trouble, the extra several megabytes don't help at all because they cannot be accessed in real mode. Further, less than 1 MB is available because the upper 384K of it is used for various system ROMs and video memory. Thus, programs must fit into the lower 640K of the first 1 MB. This is where the infamous 640K DOS memory barrier stems from.

This limitation is built into the architecture of the CPU. The 8088/8086 (and higher, when in real mode) only have 20 bits! This is a damn shame, because the 20 bits come from a 16-bit segment register and a 16-bit offset register. If one were the upper half and the other the lower half of an address, you could have a full 32-bits, giving an address range of 4 GB (gigabytes). But no, Intel doesn't do it this way. To calculate the physical address from a segment register and an offset register, the segment register's value is multiplied by 16 and added to the offset register's value. To determine how many locations can be addressed, each address line counts as a power of two. Thus, if you only had 3 address lines, you could address 2^3 = 8 locations. So, here is how the PC's address bits and address lines limit us:

```
(2^16 * 16) + 2^16  = (2^16 * 2^4) + 2^16 =   (2^20)   + 2^16.
= approximately 2^20 = 1 MB.
```

So, only 20 bits. Boo!

However, protected mode on '286 or better CPUs gets around this, even when running in 16-bit protected mode. Here is how it does it. 16-bit protected mode still uses a 16-bit segment register and a 16-bit offset register. The difference is this: in protected mode, the segment register's value is not multiplied by 16. Instead it is used as an index into a table of starting addresses. Each entry in the table holds a 24-bit (in 16-bit mode) or a 32-bit (in 32-bit mode) starting address for a segment. To this, the 16-bit (or 32-bit) offset register's contents are added, giving a 24-bit or 32-bit address. In the simplest protected mode, this 24-bit or 32-bit address is the physical address. In more complicated modes, it may be a "linear" address which undergoes further translation to convert it to a physical address.

In a nutshell, as far as addressing memory is concerned, the difference between real and protected mode is whether a segment register contains the physical address of a block of memory or whether it contains an index into a table that supplies the address of the block.

The Internet

The Internet can be a great waste of time. Reading usenet newsgroups is addictive and hypnotic. Hours pass with nothing to show for them. On the other hand, I've *got* to have my comp.lang.forth and various others. The occasional gem makes up for the rest of the rocks.

Email

For me, the internet offers three things: email, newsgroups, and file transfer. Of these, the most important is email. I so prefer to respond via email than to write a letter, print it, address the envelope, pay the post office. A phone call may interrupt someone, but email is always convenient for both parties. If you don't have email access, at least phone MCIMail at 800-444-6245 and consider their $35/year service. They even let you email people who don't have email addresses: they print and mail a hardcopy.

Newsgroups

Perhaps 15,000 world-wide news groups exist. These are essentially publicly-posted email messages, separated by topic, discussing nearly every subject imaginable. I faithfully follow a few groups related to my computer programming interests. I occasionally browse through others as the mood strikes me. Email saves time, but newsgroups waste time, but the right newsgroup article might supply just the answer you need to solve a tough problem. Together, we are a co-operative technical support self-help group.

SUPER-CHARGE THAT OLD XT!

By Kirk Lawrence

I find it difficult to think of a turbo XT-class machine as an "old" computer. Heck, I still use mine every day! While the XT might technically qualify as an "old" computer, this venerable workhorse has a lot of life left in it. With a few software tweaks, and possibly a hardware change or two, the turbo XT will continue to serve you well right into the 21st century.

One of the most common moans I hear about XTs is that they can't run the faster IDE or SCSI hard drives. Not true! 8-bit, XT-compatible IDE and SCSI hard disk host adaptors are widely available at a modest cost. Check the smaller, non-color hardware ads in Computer Shopper magazine; you'll find'em. The controller's on-board BIOS on these cards provides total support for the hard disk, and no AT-style system CMOS is needed.

Another complaint about XTs is the lack of high-density floppy drives. Again, this is an easy fix. If you feel the need for HD floppies, just grab yourself an 8-bit, XT-compatible high-density floppy drive controller card, or a multi-I/O card with built-in high-density drive support. These are also widely available ... either new (from Computer Shopper ads), or used. If there's an independent, local computer store in your town, ask'em to let you go through the boxes of "obsolete" used hardware in the back room. Almost every independent computer shop has a stock of such stuff which they've accumulated over the years. They're usually glad to get rid of it at extremely reasonable prices. Digging into this "old" stuff often turns up some really good finds.

Video? Easily upgradeable to VGA, if you so desire. Any 8-bit VGA card will work just fine in an XT. Simply plug it in, and you're in business.

Okay, that takes care of the hardware. But what about overall system performance? Well, let's face it: compared to a Pentium, a turbo XT is slow! The good news is, there are several easy software tweaks you can do which will greatly increase your XT's performance. In fact, you'll be amazed at the difference.

The first thing we can do is increase the processor speed. Yes, that's right, we CAN speed up the existing processor, and we can do it with software! Here's how: the XT's 8253 timer chip controls the rate at which the system RAM is refreshed. In a normal generic XT, the RAM is refreshed every 15 microseconds. These 15-microsecond intervals require the processor's attention, thereby consuming chunks of computing time. But the RAM in an XT really doesn't need to be refreshed that often. So if we change the DREQ0 interrupt from its normal 15 microseconds to a much longer period of time, the processor won't have to spend nearly as much of its time refreshing the RAM ... and this effectively increases processing speed. This approach will speed up the processor by 5 to 10 per cent. There are a number of freeware utilities, such as SPEEDUP and TOAD8253, which will alter the RAM refresh rate. Run one of them from your AUTOEXEC.BAT file at boot-up. If you wish, you can compile your own program from the following source code (use any shareware ASM compiler, such as A86). It's a tiny program that'll make a BIG difference in your XT's system performance...and it's not a TSR.

```
; Increase XT processor speed by altering the RAM refresh rate.

        org     100h
        mov     al,074h
        out     043h,al
        mov     ax,00D0h
        out     041h,al
        xchg    al,ah
        out     041h,al
        int     20h
```

The next thing we can do to boost XT system performance is to increase the speed of video writes. The mechanics involved here have to do with speeding up certain INT 10h functions, and are somewhat complicated. Fortunately, the shareware program QCRT Version 2.0 ($5.00 from Glenn K. Smith, RADON Software, 809 South Missouri, Weslaco, Texas 78596) takes care of this for us. Load QCRT from your AUTOEXEC.BAT file, and the speed of your screen writes will increase by as much as 80 per cent. QCRT can be found on many local BBSes and shareware CD-ROMs as QCRT20.ZIP.

Another performance enhancement is to increase the XT keyboard's repeat rate. While not strictly necessary, increasing the typeamatic rate makes a world of difference, particularly if you do a lot of word processing or other types of data entry. Once you've tried it, you'll never be able to go back to that pokey old default keyboard rate! The only catch is, you'll have to find a utility that's designed specifically for the PC and XT. Most utilities of this type are designed for AT-class machines, which support Interrupt 16h, Function 3 (an XT doesn't). I recommend the freeware program HOTKEY.COM, which can be found on many local BBSes and shareware CD-ROMs. Load HOTKEY from your AUTOEXEC.BAT file, and your keyboard will sizzle.

One final suggestion: resist the temptation to run DOS version 5.x or 6.x on your XT. The memory overhead required by these versions is too high. Use DOS version 3.2x or 3.3x, instead. When you're limited to 640K of lower DOS memory, as most XTs are, you need every extra bit of memory you can get. Almost all of the fancy "frills and extras" which were included with the higher versions of DOS (such as MOVE, FIND, etc.) are external commands, and have been available for years as third-party freeware. So if you need'em, find and use the freeware versions of these utilities, rather than loading a memory-hogging "newer" DOS version.

Even though it'll never be able to run WINDOWS 95 or DOOM, the turbo XT is a wonderfully useful and versatile machine. There's still a ton of DOS-based, XT-compatible productivity software programs out there — text editors, word processors, databases, spreadsheets, even desktop publishing programs — which will allow you to do just about anything you need to do on a computer. By implementing a few software tweaks, and maybe a few minor hardware upgrades, your XT will remain vital and viable for many years to come. And the nicest part is, the price is right!

Enjoy. Kirk Lawrence (klaw@3rd1000.com)

File Transfer

Lots of information is stored on the Internet in the form of files that can be read on-line or copied to your computer to be read off-line. It may not be easy to locate and you might waste more time looking than it is worth. The traditional method is 'anonymous ftp' (file transfer protocol), where you "ftp" into a remote system, browse its file structure with the 'dir' command, and transfer the files you want with the 'get' command. Slick user interfaces are taking over, so more and more the files are transferred by clicking on the file you want as you "surf the Web." Those with only email access can receive files, even binary files, via email.

My Internet Solution

I finally decided what to do about Internet access and a permanent email address. The two best deals I've come across are Eskimo North and Pobox.

Eskimo North

Eskimo North in Seattle offers a full-service internet account for as low as $6/month for dial-in or $3/month for telnet-only, when you pay for 5 years at once. Even though I live in Texas, I just opened a trial telnet-only account: pygmy@eskimo.com. I think I'm going to pay them for 5 years. I rather like it so far. It seems a little slow, but should be fine for reading the odd newsgroup that my local ISP might not carry. One of its users said "eskimo is a dog ... albeit a fairly reliable, friendly one." I'll use my sergeant@axiom.net account, which is a local phone call, to access the Eskimo account. Apparently, I'll also be able to use my Eskimo account for a Web page and an ftp site. Eventually, look for a downloadable version of Pygmy Forth there. Send email to info@eskimo.com for full details. By logging in as 'new' you can get a free 2-week trial account.

Pobox

Pobox offers an even cheaper solution to the permanent email address problem. For $15/year or $75 for 6 years, they provide an email forwarding service. You get 3 aliases for that price. If your real email address changes, just update your listing with Pobox. I was having a lot of trouble deciding between Eskimo and Pobox, so I think I will get them both. If one fizzles, the other should still be good. My permanent address is now pygmy@pobox.com. Logged into Eskimo, I sent a test message to pygmy@pobox.com. It arrived at axiom within 10 seconds. This seems fast enough. Send email to info@pobox.com for full details. They offer a 3 month free trial.

Axiom

A year ago I didn't expect any local ISPs in San Marcos. Now there are three! (Not counting the university.) Axiom was started by some people who have run a BBS in town for years. I expect sergeant@axiom.net to be my primary account. They charge a little more than I would like, but less than long distance or Century. The other ISP is Century Telephone Company (our local phone company). I have a beta testing account with Century, but don't like the very slow response from their technical support. I expect to drop them when they start charging me money. A third ISP has just started up, matching their rates to Century's.

GEnie and SWT

I just dropped my GEnie account to help pay for my Axiom and Eskimo accounts. My SWT (Southwest Texas University) account (fs07675@swt.edu) is inactive since I'm "laying out" this (fall) semester. Even if I enroll in further courses, I don't think I'll use my SWT much, due to its temporary and unpredictable nature. As Pobox says, "the only thing worse than losing your job is losing your email."

PC Software Miscellanea

J and APL and Windows and cetera

I've had a mild interest in the language J for sometime. It is the successor to the APL language, which is well known for intensely compact and unreadable mathematical expressions (I mean this in the kindest way). I more or less gave up on J after attempting to work through some of its documentation written by its inventor Kenneth Iverson, but hope to get back to it someday. The latest news is that Byte magazine recently had an article on J. Strand Software Inc (19235 Covington Court, Shorewood MN 55331, USA, (612) 470-7345) is the current distributor. I received their very nice *Jottings* newsletter. Phone, or email Anne Faust at amfaust@aol.com, if you'd like a copy of the newsletter. A freeware version of J for Windows (and perhaps other computers?) is available by ftp from watserv1.waterloo.edu or wuvieai.wu-wien.ac.at in directory languages/j.

One reason J interests me is that it apparently can be used to develop Windows applications. I may have mentioned that in spite of trying to think of myself as a Forth purist, I find myself accepting money for maintaining and enhancing a medical accounting package written in Clipper (a DOS-based XBase language, more or less). Call me a whore, I don't care. There seems to be pressure in the air toward making the system run under Windows. (It will run now in a DOS box.) Why slow it down so Windows can hand-draw ugly fonts? Anyway, I try to keep an eye open for how to move the application to Windows if the pressure becomes irresistible. J is just kinky enough I might look into it further. Realistically, there are several other alternatives to consider. I especially want to look at Delphi, which I still hope to install any day now. It gets rave reviews in the Clipper and database newsgroups. VO (Computer Associates' complex, bloated version of "Clipper for Windows" named Visual Objects) gets mixed reviews. Some say they hate it; some say they like it. All say it takes a lot of RAM.

Speaking of VO, I found a review of it in PC Techniques magazine (Aug/Sep '95, pp. 88-89) by James Bean to be very interesting from the standpoint of the English language and logic. He says "Overall, CA-Visual Objects is extremely powerful and provides many rich features for the client/server developer..." Wow! Sounds great. "Rich features" ooh ... not just "powerful" but "extremely powerful" ... doesn't that suggest that the product *works?* Later in the same paragraph he says "Also, when the build function executes correctly and no errors are noted from the compiler, the resulting .EXE should be considerably more robust and error free than it actually is." To what have our collective standards sunk?

C

Speaking of language implementations, I see Mix Software (800-333-0330, 214-783-6001) offers a C compiler for $19.95. From the ad, it sounds as good as Borland's, maybe better, for straight DOS work. Of course, it's another $19.95 for the debugger and $8 for shipping. Now you are pretty close to what Borland's Turbo C/C++ would cost. Various other add-ons cost $10 to $30. If anyone tries it out, please let me know what you think of it.

Bigger Hard Disks and Alternate Operating Systems

Gigabyte Harddrives

I finally succumbed to the lure of under $300 1 GB hard drives. I added a Maxtor 1.2 GB EIDE hard drive. There are various FAQs (Frequently Asked Questions) on the internet discussing IDE vs EIDE. The bottom line is either kind of harddrive will work with either kind of controller. You'll probably get faster data transfer with EIDE. There are serious questions involved in accessing a disk with more than 504 MB because of the way the PC's BIOS ROM handles disks. Before I was finished, I almost wished I had bought two 500 MB drives instead of the 1 GB drive. 1024 is the magic maximum number for cylinders. DOS can't read past the 1024th cylinder. I was reluctant to use the Disk Manager software that came with the Maxtor drive. I had heard there might be problems between it and OS/2 and/or Linux, etc. What finally worked for me was to tell the motherboard BIOS setup (the "CMOS") that I had a drive with 816 cylinders and 48 heads and 63 sectors (rather than the 2448 cylinders, 16 heads, and 63 sectors printed on the drive). The product in bytes, either way, is the same. This keeps the number of cylinders under the 1024 limit. I also told the BIOS setup to use "LBA" (logical block addressing). I _think_ this was necessary. If I understand the problem, DOS can't handle more than 1024 cylinders while IDE controllers can't handle more than 16 heads. The LBA mode of the motherboard BIOS apparently converts between DOS's view of the drive geometry to a linear block addressing mode that IDE controllers can handle. I was tempted to turn off the LBA mode in the BIOS setup and see if the drive was still accessible, but "let sleeping dogs lie" came to mind.

If I had not had an LBA mode in my motherboard BIOS, I could have set a jumper on the Promise EIDE controller to use LBA. So, it turned out I probably could have used my same old IDE card without buying the Promise card. Oh well. Maybe it's faster now. Also, I had a terrible problem with the old controller. I had a 1.2 MB floppy as drive A and a 1.4 MB floppy as drive B. Diskette changes in drive B (the only floppy drive I used, practically) were never recognized. Whenever I wanted to switch floppies, I had to reboot! This was an annoyance, but was only a major problem when installing software from multiple floppies.

Since the world has moved to 1.4 MB 3-1/2 inch diskettes, I put the 1.4 MB drive as A and the 1.2 as B when I installed the new controller card. Now changing diskettes in drive A works fine. I suspect the motherboard/controller combination, as I used the same model IDE controller and same model floppy drive, with the same switch settings, in another computer with no trouble.

I bought the EIDE card and the 1.2 GB Maxtor hard drive from the Internet Shopping Network (http://shop.internet.net). They shipped fairly promptly, but I paid for next-day deliver and got 2-day in one case and and 3-day in the other. Why two cases? Because they shipped the items separately and charged me the separate shipping on each. This automated ordering has its disadvantages. Next time, if I deal with them, I'll probably ask for UPS ground.

I put in a "MEI Premium Backup 420" tape drive (about $99 from MEI/MICRO). This uses the QIC-80 tape format and variants. It is really an IOMEGA drive relabeled for MEI. The ad said NOTHING, the salesman I spoke to on the phone said NOTHING about the potential problem of using certain kinds of tape. After I bought it, after it arrived, I read in the documentation that using the 2120 (ordinary QIC-80) tape can ruin the tape head and that the more expensive QIC-80 WIDE tape is recommended. The ad listed the 2120 as one of the several kinds of tapes the drive supported. It's not worth the trouble to send it back. I'll use it until the tape heads are ruined and then buy another drive *from another vendor!* A "420" tape drive really only holds about 210 or 220 MB. The "420" comes from the possibility of compressing the data. Maybe your 420 MB of data can be compressed down to 220 MB, maybe not. That "420" is on the more expensive tape cartridges. The cheap ones I got for about $9 each hold much less.

The actual tape backing up and restoring went smoothly. I'm really glad to have a tape backup system. One day, I'll probably get a better one. For now, I'll back up to tape and also back up the most important files to floppy. For rearranging my harddrives, the tape was great.

OS/2 Warp

While I was putting in a new drive and having to reformat and repartition it anyway, I thought this might be a good time to try out OS/2. It was mostly a waste of time, but I learned a lot.

The OS/2 I read about in one of the FAQs sounded great! I wish

that had been the version I tried out (joke). It seems that OS/2 will run essentially all DOS programs and all Windows programs, plus, of course, any programs actually written for OS/2. Not only that, it will let you multitask them. The Internet reported a magazine article claiming Win95 and OS/2 ran applications approximately the same speed if only a single app was running, but that once you started to multitask, OS/2 was perhaps 4 or 5 times faster. I wanted all that plus the HPFS file system, which allocates disk space in units of 512 bytes, instead of the 4K to 16K chunks used by DOS.

I never could get OS/2 to talk to my printer, which was on a serial port. DOS doesn't have any trouble with this. Nor, could I get OS/2 to give my DOS comm program the data from the modem quickly. There was about a two second delay between pressing the "show me the next page" key and seeing the next page displayed! DOS has no such trouble. I feel sure there are work arounds for these problems, but life is too short. It is this sort of trouble that softens me up for Windows 95. Even if Win95 is no damn good, it will still own the market. Time spent learning to install, use, support Windows is probably a better investment than spending it on OS/2.

On the other hand, I mildly complained of my troubles on an OS2 newsgroup and got very nice email from an IBM employee offering several suggestions on setting up OS/2 and where to look for further information. Later, I also could not get Linux to work with my serial printer, so I don't see how I can feel too bad about OS/2. I gave up, moved my printer so a parallel cable would reach, and put the printer on LPT1: (the first parallel port), where most PCs have their printers. After that, I re-installed OS/2 and it worked fine with the printer. I still haven't figured out how to get fast modem response, but I'm sure it can be done.

Disk Partitioning

Here is how I finally split up the 1.2 GB drive:

1 MB primary partition for Boot Manager	
11 MB primary partition for booting MS-DOS 5	Drive C:
45 MB primary partition for booting MS-DOS 6.22	or Drive C:
255 MB FAT data partition for DOS	Drive D:
255 MB FAT data partition for DOS	Drive E:
255 MB FAT data partition for DOS	Drive F:
255 MB FAT data partition for DOS	Drive G:
53 MB reserved for Linux	
66 MB HPFS OS/2 boot partition	Drive H:

On a FAT file system, the size of the allocation unit depends on the total size of the logical drive. I would prefer to combine logical drives D through G into a single logical drive, but can't afford the wasted space. That's why I have four 255 MB partitions instead of a single large partition for my DOS data. With the small partitions the allocation unit is 4 KB. So, a 200 byte file actually occupies 4 KB on the disk. If I had combined the four 255 MB partitions into a single 1 GB partition, the allocation unit would have been 16 KB. Ooh, that hurts.

Partitions beginning with Drive D are all logical partitions in the single remaining extended partition. Only 4 real partitions are allowed. If you want more, you must make at least one of the partitions an "extended" partition, which you then keep carving up into additional logical partitions. DOS and Boot Manager will only boot from a primary partition. OS/2 will boot from a logical partition and so will Linux (so I've heard). Drive H: is only visible to OS/2, since it is not formatted as a FAT (File Allocation Table) type of file system. Instead, it uses OS/2's native HPFS (High Performance File System).

Yes, I left OS/2 on the machine. Its FDISK command works better than DOS's. When OS/2's boot manager comes up, it offers me a choice of which operating system to boot, but defaults to DOS 6.22 if I don't pick another within 30 seconds.

Networking

I also bought two ethernet cards from MEI and installed one in the 486DX40 and the other in the 386SX25. Their test programs indicated all was well. They could "ping" and "pong" data back and forth. Then I tried to install the Little Big Lan. It sort of worked. It would start a transfer, then lock up. I gave up on it. When I have more time, I'll try Novell or Lantastic or Linux to see if they work ok. If so, I'll blame the Little Big Lan. If not, I'll blame the 386SX25.

Linux

I installed Linux. I bought the "Linux on CD-ROM Slackware 2.3 Slackware ELF Beta" ADRAS July Edition for $10 plus $3 shipping and handling (plus sales tax for Texas destinations) from

Daniel Jimenez adras@crl.com
ADRAS Computing
PO Box 29391
San Antonio, TX 78229.

Linux and the entire Slackware package are impressive. You sure get a lot for the $13. A massive amount of documentation comes on the CD. You can install Linux entirely to your hard disk in its own partition or to your hard disk in a DOS partition in a regular DOS subdirectory or you can install a minimal amount to your hard disk and run the rest of Linux from the CD, as the CD contains a "live file system." The version of the kernel is 1.2.8. I believe the CD contains other versions of the kernel as alternatives, plus source code for even the newest experimental kernels (the 1.3.x series). (The stable kernels have an even second number while the kernels under active development have an odd second number.)

I ran into a little problem. My CDROM drive quit working. I opened the computer case, removed the drive, opened it up, reconnected the cables, and watched it try to run. The CD wasn't spinning. I started it spinning with my fingers and

Continued on page 17

The European Beat

by Helmut Jungkunz

Regular Feature

All Users

East German Z80

Today: Jubilee - 10 Years of Schneider/Amstrad CPC User Group Munich!

Fascinating, fascinating ... the unthinkable has happened. Our User group has survived heavy attacks of Bill Gates and Co. for many years. We have seen many an operating system come and go. We ran them all. We loved them and we hated them, but our secret love still is CP/M. It was such a neat little system, so much in order, so easy to dive into, so easy to understand, once the exploring mind had the basics.

In February 1985, I got my AMSTRAD CPC 464 (with cassette drive only), a right angular gray box, with a gray Monitor, called "Green Monitor" because of the color of its monochrome output. It cost me a fortune (500 $+) and after a while I found out, I couldn't do without a disk drive. So, I spent another 350 $ on that little (gray) box. I bought the "CPC Amstrad International", a national magazine for Schneider/Amstrad CPCs. For those of you, who haven't been following from the start of "The European Beat", Schneider was a German company, working as an agent for AMSTRAD in Germany, who renamed the CPCs into "Schneider CPC", so that everybody thought, that Schneider was the maker of the CPCs. I learned how to type in BASIC programs without mistakes, yes, even how to correct errors in the programs. Then, I discovered the CP/M side of the machine. After a short period of A> ? and A>BDOS error on A: Select!, I managed to work with it - and was immediately limited by the low TPA of 39K.

So I went and bought the brand new Vortex RAM expansion kit. That's when problems started. I couldn't use my disk drive the way I had, since I kept getting these "hangers" when switching the computer on: The initial "ROMWALK", that reset all the ROMs and/or EPROMs, failed to return from the disk drive controller, that contained the disk ROM. Then I read about the "VORTEX User Group" and joined them. I was one of the first seven, which put me on the lucky side, but more about that later on. Anyway, I met a clever guy, called Andreas Kisslinger, who had the correct explanation for what was happening. Since you couldn't do a disk catalog command "CAT", it was named "The CAT Syndrome". After replacing the ROM in the controller with a new EPROM, the problem was gone. Also, the BIOS within the expansion unit was very poorly written, plans arose to rewrite some of the code.

Another guy in the club soon took some of us aside and pointed out, that the host of the VORTEX User Group, a clever dealer, always tried to sell us some stuff and more or less ripped us off. He said, that he was going to form his own CPC User Group and invited us to join. After a Powwow, we decided this was a good plan and sat down for the founding procedures. In November 1985, the formal foundation of the Schneider/ Amstrad CPC User Group (SCUG) was a fact.

Then followed a period, when the host of the VORTEX User Group tried to force "membership fees" on us. A couple of us had heard from the first ones, what was happening, and we immediately sent our announcements notes to that guy. He then wrote back "Oh, but you guys didn't have to pay anyway, since you were among the first seven ...". We quit anyway. The others got together an alarmed the district attorney. Thus we found out, that the dealer played the same trick on people in other parts of Germany as well, but had his company site legally in Switzerland(!). He was fined in a law suit at long last.

The SCUG soon had lots of new members, and the usual discussions and arguments took place, like in every club, perhaps. There were a couple of people, that came to us from the VORTEX User group, who were very much afraid of any incorrect phrasings in our "membership agreement". After months, we had it all sorted out, and the group started to produce useful output. Andreas Kisslinger wrote B360K, the famous first CPC BIOS RSX, that allowed not only to hook up an 80 track drive to the CPC as drive B:, but to also patch the

40 track BIOS to accept 80 Tracks single sided, which gave 360K minus reserved system tracks == 348K net capacity formatted. There was a hardware diagram, allowing you to build your own Z80 PIO. The first Public Domain disc was collected, and - the first SIG/M programs found their way into the group.

There was one guy, who sold the disks to members at a reasonable price (8 Deutschmarks a piece), and I bought a couple. When he announced his moving to another city, I decided to buy the whole packet from him (up to #271 then). I started to browse through the disks soon and found lots of useful tools, that I put together on another P.D. disk, free to club members. Those were tools like NSWEEP (1.99), NULU (1.5), DELBR,USQ, SAP10 and ALIENS. That was the start of my hacking into the system. Reading the patch documentation from ALIENS, I learned how to adapt cursor and other terminal functions to the program. I looked at other programs, too and found it pretty annoying, to have to install every program to our weird CPC CP/M 2.2 terminal. The main problem was, that most programs were written for TELEVIDEO terminals, using CTRL-Z to clear the screen and home the cursor. The CPC in CP/M 2.2 uses CTRL-Z to define a true screen Window! Imagine, the Syntax would be CTRL-Z,number of window (0 being the main screen!), starting row, starting column, end row, and column. Now imagine the CP/M seeing CTRL-Z by itself. This would be the equivalent of "Do a Window". Now CP/M would look for other characters to complete the window. In most cases, the CTRL-Z (Clear Screen) would be terminated by Zeroes. The result was a catastrophe to the CPC screen: WINDOW 0, 0 row, 0 column, 0 end, 0 end! - ==> cursor at home position, stuck in a tiny spot at the upper left corner, unable to move!

I got so disappointed, after failing to install Polish Pong, that I turned away from the SIG/M stuff for three weeks. After that, I gave up, trying to install P.D. games. Instead, I started to look at more sophisticated stuff, to help increase the performance of our CPCs. After a short while, I reached SIG/M 101 and got pulled into the deeps of Richard Conn's ZCPR introduction. I presented this to our club (end of 1986!) and tried to catch our programmers interest to implement ZCPR into our CP/M 2.2

I got weak replies, phrases of "missing BIOS source", "no time", "other projects", and "Is it really worth it?", until the whole thing just dropped dead. It was then when I bought my first IBM XT, in order to keep up with the small side jobs at computer fairs, that I got from "Markt&Technik" (M&T), when I was connected with their "Happy Computer" magazine. I bought a 1200 BAUD Worldport pocket modem and started to "walk the phone lines". I soon became CP/M Co-Sysop at a BBS in Munich and made contact to other Sysops from other BBSs. I met Peter Spaeth, who was a member of the German Computer Club (forgot the actual name), but also operated as a librarian for the Osborne User Group. From him ,I got lots of new files and insights. The most important thing was, that he had all the FOG magazines, where many articles dealt with CP/M programs, hitherto unknown to me an my fellow users. I read about REMBRANDT (graphics program for the KAYPRO), and other nice things. There were a lot of mentioning of ZCPR and Z-tools.

I also had access to the Morrow Owner's Review (MOR), for one of our newer members, Timothy Slater, was working on a Morrow Micro Decision. He is a translator for technical German into English, since he is American.

I read all these wonderful developments in ZCPR in articles like "Tools for Tyros" and "Forever Z!". The latter series was written by Rick Charnes, a guy, whose phone number happened to be included. So one night, I sat up and called Rick Charnes "on the dodgy subject: are there or are there not Flying Saucers" (I can see all them Jimi Hendrix fans grin!). Well, I asked his opinion on ZCPR. He was so overwhelmed with Z, he set my phone counter spinning, telling blooming stories of how Z changed his life and that it was not an operating system replacement, but a religion. Believe me, he made it sound that way! I was stunned (at five o'clock in the morning, who wouldn't be!) and decided to watch out for more. I sent some programs on a disk to Rick Charnes and that was all for then. Replies died out soon. Then I read about that new development NZ-COM, available from Echelon! Automatic ZCPR installation for every system! I showed this to our members and got a more interest than before. But of course, they would still sit and wait, if I hadn't tried to contact Jay. The rest of the story you already know.

SCUG has lots of members with Z-System, of course, and I provided them with the AMSTRAD terminal definitions. I initiated the production of the DOBBERTIN hard disk for the CPC (they asked, if 50 pieces were realistic) and they sold hundreds of them. The DOBBERTIN RAM expansion is a lawless unit, fully compatible to most everything, the ROM-Box allows you to add-on further EPROM software and/or BIOS extensions.

Today, we are facing the fact of "Future OS", the "final operating system" for the AMSTRAD CPC, written by one of

our members, a no-compromise solution, that doesn't claim to be compatible to anything. It plainly isn't. But, it offers nice software hooks for other programmers, plus, it comes with source.

The official SCUG celebration will be held in our traditional Club home, that we only shortly regained, the "ASTALLER HOF" in Munich, on Saturday, November 25th. The fact, that the very next day, the famous Electronic Flea market takes place only 1 mile from that location, might well help to attract lots of (German) people.

What does the future hold? Well, some of us still have a CPC and run it, some keep it in a nice, romantic corner of their homes, some have a PCW and most of us have some kind of x86 machine with emulators like CPE or CPCEMU on them. We use all kinds of operating systems, like CP/M-80, CP/M86, CP/M-68K, MP/M-80, CP/M Plus, MS-DOS, IBM-DOS, Novell-DOS, LINUX, Coherent and SYSTEM 7 (MAC). We plan to keep the CPC in our banner as a signal to others, showing our tolerance against weaker systems, our acceptance of help-seekers and a good will to keep up a piece of tradition in this oh so fast world.

Regards and cu
Helmut Jungkunz

(P.S: I'd be glad to receive your comments on my articles, best through e-mail, since this creates a computer file that can be passed to others.)

XT Corner Continued

everything worked but it wouldn't start up reliably by itself. I fooled around with the upper bracket that clamps the CD in place, bending it higher and lower, until I finally set it so it seems to start up reliably. We'll see.

After that, the Linux installation went smoothly. I had planned to put Linux in its own partition, having left a 54 MB partition just for Linux on my new 1 GB disk . Well, 54 MB is nowhere near large enough to install everything. I planned to install the minimum and run the rest from the CD. I think this would be too slow for comfort, but would be ok for a quick look at Linux. Since I had the case open any way to fix the CDROM drive, I took that opportunity to reconnect my 330 MB hard drive and devote 280 MB of it to Linux. I installed nearly everything. I occasionally said no to things such as the fonts and macros needed to typeset Turkisk documents, but I installed damn near everything else! Far more than I'll ever use. I said it was impressive. My impression is Linux, like the rest of the modern software world, is too damn big and complex. On the other hand, a 330 MB harddisk is pretty cheap these days and a 120 MB is large enough if you don't install all the extra packages and source code you'll never use. You don't have to have a CDROM drive (you can install from floppies), but they can be had for about $79 now. I don't expect to do much with Linux, but I wanted to take a look. So far, I've barely looked at it. I've used its comm program (minicom) to log into my internet accounts. The medical accounting package written in Clipper partially runs under Linux's DOS emulator. I think I need to change some settings related to the type of video display.

I've been subscribing to the

Linux Journal
P.O. Box 85867
Seattle, WA 98145-1867
(206) 782-7733
$22/year for 12 issues

and enjoy it. I think working through various back issues will serve as my Linux tutorial.

Frank Sergeant
pygmy@pobox.com (permanent)
sergeant@axiom.net (current)

Special Feature

XT Support

Different CPU on BUS

Alternatives to the XT

By Bill Kibler

For some time I have been trying to complete an article on using the PT68K motherboard with OS9 and SKDOS. Orginally I had intended to cover it in this article. Now that I am passing on the control duties of *TCJ* to Dave, I decided I will have time later to more fully explain and document the ideas I had in mind. It became apparent as I started on the topic that an introduction was needed about systems in general that give you alternatives to just buying a faster XT. I will give some background of the PT68K card, but this time the topic is mostly what alternatives you have avaialable.

The Why

Why would you want to do something other that use an off the shelf XT system for your work? Defining work immedately gets us going and is somewhat responsible for the change of this article. At my work, we had a recent discussion of a problem requiring better than one second turnaround from the time a person touches a screen, requesting a door to be opened, and having the door actually open.

The use of Touch Screens is very common, but all seem to have one problem, they are slow. Generally speaking, a standard 386/486 PC has a special screen surface attached and putting your finger to this surface, generates the equalivalent of a mouse click at the location your finger touched. The problem is the ever present tons of PC overhead, extreme slowness of conversion as your commands works their way through the many layers of various programs and TSR drivers to eventually appear out the back into yet more interfaces.

It has not been uncommon for this action to take 3 or 4 second to be completed. That is from the time the finger hits the screen to have a door open. The question asked was what alternatives are possible. We know the times possible for PC/XT type systems running the latest CPUs, so the first direction we look at is fasters CPUs that can use the same hardware. This is where I said "how about a 68000 based XT system running OS9?"

At this point in my work discussion I realized that few people are aware that systems using the ISA BUS format (the bus structure in a PC/XT) but running CPU types other than 386/486 do exist. That same realization said this article needs to be more general and explain the alternatives.

Same but Different

The option to the problem presented would be simply cutting down the layers of software, since 386/486 are basically speedy processors and DOS is not. OS9000, the PC/XT varation would be one answer, EXPRESS by Forth Inc another. Numerous embedded DOS and real time variations might also speed up the system. Moving to PC104 by itself would not help, but ROMing code that only read the screen and sent a command might. The PC104 bus is simply a form factor change of the ISA bus. Instead of edge connectors for the bus interface, you have a pin and socket system of stacking cards. The main difference we see here is separate CPU cards allowing a pure building block approach.

The building block approach is what most organizations are taking to solve their problems. This approach is not new, just the hardware options have become more interesting. In my last article on the STD BUS we saw that they started as a building block solution, giving the users many choices of small pieces in which to plug and play. The PC104 vendors are providng the same options.

What is new on this front, has been the wealth of different CPU types now available. Z-World has moved their Z180 based product line to the PC104 and ISA bus as well. This means you can take an application written for the Z80 and make it talk to regular PC bus devices, say ethernet cards? I am not sure this would be faster, but it does show that the wealth of PC compatible hardware can be used on other CPU types.

I found one company who has taken this approach to extremes. The parvus company from Salt Lake Utah produces several CPU types on 2 by 3 inch boards. All the boards have a memeory mapped interface that can be plugged onto a PC104 or ISA bus interface card. They can sit as co-processors with one type of card, or take over the bus if plugged into another type of card. Their main CPU of choice is 68HC11, for which they have written drivers to talk to most of the standard PC devices (CGA/Mono, floppy, hard drives, etc.). 8051 and Z180, and 80198 are the other CPU options.

Might any of these varaiation slove the problem, I am not sure, but if your old application needs to talk to newer PC devices then here is an alternative. The ISA bus is basically an 8 bit bus with 8MHZ clocking. This 8MHZ clock speed makes it possible for the smaller CPU's to use it, but also limits I/O speeds and is often the reason for poor performance, even on a Pentium CPU. To change this we have IBM's approach that failed, the

MCA bus. Better performance but required major changes in hardware and had numerous compatiblity and licnseing problems.

The latest option we find is the PCI bus. This is basicaly a 30 or 50MHZ variation on the ISA bus. I don't have all the actual specs yet, but from what I can see, the industry leaders are jumping on this bus quickly and from many directions. Since it is getting major support and doesn't have any licensing agreement needed for use, I think it will become the high speed bus standard for many years. An other advantage of this bus is it's acceptance by several industrial manufactureres. Prolog which helped start the standard bus, has said they are making it their bus of choice. All the new Mac Power PC's use it as well.

The industrial version of the bus, much like PC104, uses sockets and form factors closer to VME or European. Prolog has gone even farther by using PCI and STD bus devices in one box, giving the user access to both worlds. The idea here is high speed CPU/Memory/Network components, while keeping the slower real world interfaces of STD BUS I/O.

68K Option

Most of the faster options so far use the standard 486 or Pentium CPU. I have found these CPU's overkill and overbudget for many simple projects. For those simple items only needing a few PC based I/O cards or migrating from a 8085 STD bus project to the PC world, using a 486 is hard to take. You could use one of the Z180 ISA bus controllers or try a 68000 system. Why 68K CPU?

I can think of many reasons for using the 68K over the 486. Generally you can gain some speed using 68000 CPU's just because of the better internal CPU design. The instruction set alone is better optomized and will produce smaller program sizes for similar operations. The 68000 has always been a 32 bit CPU with a flat address space. The dreaded segmentation that has made programs on x86 CPU's a night mare, doesn't exist on 68K.

Generally I find the cost of CPU power cheaper in the 68K's than Intel's, although recent CPU Clones have brought all prices down. The alternatives I like are using the $200 68K based clone boards (cost including CPU but no memory.)

The PT68K

So here we are now looking at an alternative. Where did this alternative come from? Well certainly there were several vendors who were producing 68K bussed systems and saw the need for the use of cheaper PC interface cards. One person however was looking more for a means to teach how computers go together. That person is Peter Stark of Star-K Software System Corporation, the maker of SK*DOS.

Peter is a college teacher and came out with a series of articles in Radio-Electronics Magazine in October of 1987. The article had the reader building their own 68K based PC compatible system. The kits were available from Peripherial Technology and you could build your own as the articles explained the various parts. The idea was to talk about and teach a different part of the system in each article and thus over several issues completely explain how all the different sections and components work together to make the full computer.

We have attempted to do similar projects here in *TCJ*, but I must say that Peter's series is excellant and worth hunting down in your local libarary. Peter also sells the material as a home course, called "68000 Hardware Course" at $25 (the hardware kit cost extra).

So what do you get and is this unusual? Since getting mine, I have discovered that at least two or three other vendors sell similar systems. They appear to all have similarities in design. The main idea is some version of the 68000 CPU (68K to 68040), on board I/O (serial and floppy), a number of standard ISA or ISA/AT bus slots. ROM monitors are provided that allow for running OS9 or SK*DOS and can do some debugging. Some ROMs may even contain BIOS like code for accessing PC cards such as hard disk controllers.

Since the idea is mostly to run these products as embedded controllers, the serial I/O is normally included. This means they can be brought alive with a simple terminal or in the case of the PT68K and it's floppy controller, you can boot from floppy disk atuomatically. I have done some playing with the PT68K and found the ROM tools good, and the machine fairly snappy in it's performance. A more detailed testing is scheduled for a later issue.

Let me take a second of your time to explain the hardware in more deatils. First off the board size is that of full AT motherboard. It comes with the standard PC power supply connectors strips. You can install monochrome or CGA video cards in one of the seven XT slots. A connector is provided for one PC compatible keyboard. I beleive that later versions also support VGA cards, if not in ROM, at least in loadable drivers.

In the case of the PT68K-4, you have two floppy disk controllers to chose from. It comes with both a WD1772 and the WD37C65. The 1772 is for XT level floppy drives, 360 and 720K. The 37C65 will handle all drive types, including the 1.2 and 1.4 meg. The 1772 does not use the twisted drive cables found in PC's, while the 37C65 does. This gives you backward and forward compatiblity.

Standard reset and speaker pins are provided. I have mine mounted in a small tower case and it now works fine. I did have problems earlier when I tried mounting it in a older style case with metal standoffs. Apparantly there is a problem existing around one or two of the mounting holes that you need to watch out for. If you think this is only their problem (PT68K) let me say that many of the PC motherboards have the same problem. This explains why almost all PC cases come with plastic mounting studs these days.

What else? Printer cable header is provided just like the floppy headers. All the afore mentioned interfaces are on the motherboard as header strips. This means

you have to make cables that run from the header strips to the appropriate connectors on the back of your unit. Also you can use the printer port of your monitor card, if it has one.

Hard drives are supported by using the WDXT-GEN2 interface cards. These were very popular hard disk controller cards used in XT and early AT systems. There are many clones of these around and one that I tried didn't work. Since all my cards are very old, I haven't had time to find out why it didn't (might be bad - I certainly don't know - yet) work. Supposedly you can use IDE drives as well, but that all remains for the next article.

To expalin the next part of the design, I need to review how these systems came about. The vendors were all 6809 providers, mostly SS50 BUS running OS9 (this case the "9" stands for 6809 CPU). OS9 has always been multitasking, or able to run more than one user. I have several stories from people who were running whole departments on single 6809 systems. Naturally when the 68000 came out the users wanted to migrate to it and gain all the extra power it offered. Remember that DOS machines are still not multitasking unless you go to Unix or NT boxes. Those early day systems running Unix in fact where all being done on 68000 CPU's.

Using Unix, also meant that more than one user would be on board, and most users would be on terminals. To help out the boards come with two 68681's DUARTs, set as 4 terminal ports. Here also is one of my complaints about the design, mainly that the four header stripes used for serial terminals are not cable ready. By that I mean you must hand wire the header strips to the RS232 sockets. You can't just crimp a header on one end and a DB25 on the other. Now we are only talking about four wires, but then why a header strip at all?

That minor complaint aside, the board does come up looking for either terminal on com port one or a PC keyboard. The ROM menu is sent to both output options so you can play with either. The orignial ROM from Peripherial Technology is ready for OS9 and REX. REX can also be had from them on disk with source code. This means you could hack this version for your own personal project.

If you want to try SK*DOS a different ROM is needed from Peter. Then you can run either SK*DOS or OS9 from that ROM. The ROM also has BASIC and HUMBUG. You can partition your hard disk to boot either 68K system, and thus try or use either. I think you can also throw REX in this as well, but not sure at this moment. I do plan on reviewing all three and try some hacking to find out which system is easiest to port to other hardware platforms.

Overall the system seems ideal for many projects needing specialized support, and where a 8 bitter will not work. The board comes with enought sockets for 4MEGs of RAM, which should be more than enough for single user projects. Now OS9 is used in many of the newer CDROM projects and has been in all the big name systems, like NASA. It is UNIX like and a real time system that can support more than one user. Yet I know it works well for single users and many CoCo users fell in love on with their 6809 based systems.

SK*DOS on the other hand is more of a single user product with it's roots in FLEX the other 6809 operating system of choice. Peter has added some nice features, like an 6809 emulator that will run your FLEX code. I plan on trying that part out and see how it compares with my GIMIX 6809 system. A good source of software is the 'C' programs and utilties avaialble from Unix. Many of these are provided with SK*DOS and so you get a chance to see how other code works on this platform.

Closing Words

At this point I leave many items and topics unexplored, but I will have more time to devote to this project very soon. From the hardware side, many companies are starting to use PC hardware for their 8 BIT projects. Alternatives to the Intel CPU's are also out there, the most common being 68K based mother boards. What I didn't mention are the numerous co-processors available and soon multiprocessor system. These systems are often DSP or Math co-processors that enhance the basic PC with more horsepower of a certain type. There are several vendors starting to produce systems with more than one CPU so that the task can be broken up and done in parallel. Brad Rodriguez's project using 6809's (on hold while he finishes his degree) has itself moved to the PC BUS platform to take advantage of the cheap hardware.

What all the options are giving the user and designer is the abiltiy to custom fit any project or task with hardware that provides the exact range of solutions. In the past you often simply tossed the same box at any project and adjusted the software to make up for missing hardware support. Hardware now is so cheap that it has become more cost effective to custom fit the hardware than squeeze out corrections in software.

I hope you found items in this article that got you thinking. Should you have more questions, drop me a note or contact the vendors, be sure of course to say you heard about it in *The Computer Journal.*

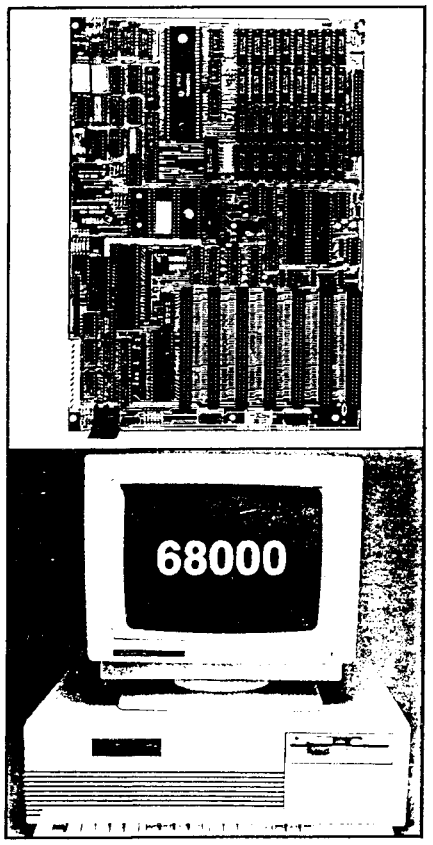
Star-K Software Systems Corporation, PO Box 209, Mt. Kisco, New York, 10549, (914)241-0287, BBS (914)241-3307

The parvus Corporation, 1214 Wilmington Ave., Salt Lake City, Utah 84106, (801)483-1533, BBS (801)483-1563
parvus@parvus.com

Peripheral Technology, 1250 E. Piedmont Rd., Marietta, GA 30062, (404)973-2156

Z-World Engineering, 1724 Picasso Ave., Davis. CA 95616
(916)757-3737

PT 68K-4 SINGLE BOARD COMPUTER



FEATURES

- MC68000 Processor, 16MHZ Clock.
- 512K to 4096K of DRAM (0 wait state).
- Optional memory board supports an additional 6MB of memory.
- 4K or 64K of SRAM (2K×8 or 32K×8).
- 32K, 64K or 128K of EPROM.
- Four RS-232 serial ports; uses the MC68681 DUART CHIP.
- Winchester Interface — supports a Western Digital MFM XT hard disk controller.
- Floppy Disk Controller
  Supports 2 360K, 720K, 1.2M or 1.44 drives.
- Clock with on-board battery.
- Two programmable interrupt timers.
- 2 — 8 bit Parallel Ports. May be used with a parallel printer.
- Supports VGA (up to 1024×768 resolution) with OS9 Operating System
- 4 layer board — 12.0×8.5 inches.
- Board may be mounted in an XT or Baby AT cabinet. Power connector matches an IBM type power supply.
- Expansion ports — 7 IBM PC/XT compatible I/O ports. One memory expansion port. Use of the memory expansion port also requires use of one of the XT slots.

PERIPHERAL TECHNOLOGY
1480 Terrell Mill Rd., Suite 870
Marietta, Georgia 30067 USA
404/984-0742

REX OPERATING SYSTEM

REX is a single user operating system for the Motorola 68000 family of microprocessors. REX is simple and easy to use and will run on PT68K2 or PT68K4 computers with at least 512K of RAM. REX supports advanced features such as ramdisk and track buffering and includes over 50 utilities. Full source code is also available.

OPTIONAL LANGUAGES/UTILITIES

BASIC ASSEMBLER EDITOR

SUPPLIED UTILITIES

ACAT	APPEND	ASN	CAT	CHECK	CMPBNY
CMPMEM	CS	DATE	DELETE	DISKNAME	DRIVESET
DUMP	DUP	ECHO	EDD	EJ	EPBURN
EXAMINE	EXEC	FORMAT	FORMATDD	HARD20	HDF20
HECHO	KEYCHECK	KRACK	LINK	LIST	MAP
MEMEND	MAKDISK	NEWDISK	OLOAD	P10	PDEL
P12STAR	PDEL	PE2	PN2	PNLQ	PRDVR
PS	PS2	REDATE	RENAME	SAVE	SAVETEXT
SCAN	SETIME	SYSTEM	TCOPY	TIME	TURBO
VIEW	WORK	YEAR	ZAP		

MONK MONITOR PROGRAM

MONK is a BIOS and debugging monitor for the 68000 family of microprocessors. MONK supports RS232 terminals, PC compatible MONO, CGA, EGA or VGA display cards, and PC compatible keyboards. The MONK monitor includes 21 commands which are listed below.

B - Set Breakpoint
C - Change Register
D - Display Memory
F - Restart REXDOS
G - Continue after Breakpoint
J --Jump to an address
K - Kill Breakpoints
M - Memory examine and change
N - Calculate Checksum
O - OS9 Boot Menu
P - Display Breakpoint
Q - Set quick breakpoint
R - Display registers
S - Execute one instruction
T - Test memory
U - Load REXDOS from 1772
V - Load REXDOS from 37C65
W - Load REXDOS from Hard Disk
X - Select Monitor type
Z - Fill memory with data
? - Help command

Regular Feature

Intermediate

Jade BUS Probe

Dr. S-100

By Herb R. Johnson

"Dr." S-100 column for Nov-Dec 1995

Herb Johnson, CN 5256 #105, Princeton NJ 08543. hjohnson@pluto.njcc.com

Even though my grass is now the height of the average cat, I've decided to lock up my mower (at least until the grass dries!) and sit down and write up some technology. Many people...well, Bill Kibler...wanted a write-up of the Jade Computer Products Bus Probe I mentioned several months ago. Bill discussed articles and products with the Jade company - still in business, but not the S-100 bus business of course. While they did not want to "release" the designs to the public domain nor sell licenses to their designs, they did allow "non-commercial" use of their materials, for which I thank them and Bill.

In addition, there is progress on the GIDE front to report! So no letters this column, but please tell me what you are running on **your** S-100 bus, or what you are looking for, as I am still getting boards and systems and reports. And, now that winter is approaching, I can spend more time in the basement..uh, the Archive Lab...sorting and examining all this technology.

GIDE IDE hard drive to Z80 interface

After a delay in locating parts, Tilmann Rei has recently shipped some GIDE development units to the United States! I have one prototype already, and should have more by the time you read this. And several developers will have them in their hands from mine. I hope to get software from them to support the IDE drive in real machines (Kaypros, Cromemcos, Zorbas, other Z-80 based machines) in the near future. Then the "rest of us" can buy and use these devices.

For those of you not familiar with GIDE, it is a **"Generic IDE" hard drive interface card that plugs into a Z80 socket**. Tilmann Rei in Germany is the designer of this card, based on his work two years ago on a Z180 single-board computer with an IDE interface using a PAL (programmed array logic) chip for the logic "glue" to communicate with the drive. Several people, including myself, suggested he come up with an independent interface for any computer; I suggested he make it pluggable into a Z80 socket, and that he add a **clock chip** for a time of day clock. A year later, he now has some prototypes available and have imported a few of these into the United States. Software, of course, is not available beyond routines to read and write the clock and the IDE drive: those who buy this prototypes must be able and willing to write Z80 drivers and share the code with others. But I'll include Tilmann's IDE articles, and some BIOS software from other folks for other kinds of devices like RAM disks, and docs on the clock chip.

I'm taking orders in the USA at **$73 each including the clock chip**, delivered unassembled. You will get a board and parts, with whatever docs and is available. If you want details, contact me (Herb Johnson) via mail, phone, or (preferably) e-mail. If you want **information** (two years of Tilmann's TCJ articles on IDE and earlier GIDE, and descriptions of the clock chip), **send $5**. Another $2 gets a disk of whatever software we have (mostly other people's software on doing various BIOS extensions).

Remember, this is **development** stuff, not "plug and play", so if you can't write Z80 assembly language and don't know your own Z80 system at the BIOS level, you will not find this useful **at this time. BUT, a letter or call showing support and interest when you CAN use it in your machine will be greatly appreciated.** After we get some useful software, and when I can manufacture a regular supply here in the USA, it may become a product.

People talk about how hard it is to get support for their old computers. Well, here's a chance to show some support!

The S-100 bus and the Jade Bus Probe

Through all my years of S-100 work, there are two systems everyone knows about and wants. One is the Altair 8800, of course; the other is the IMSAI 8080. Why? Not because they want to toggle all the binary switches on the front panel, but because they want to **see all the blinking lights** on the front panel! Few other systems offer this display of address and data lines and status signals: a notible exception is the **Ithica Intersystems** model with a really slick front panel design, including an automatic step-through of the program at a few steps per second. I think it's one of the prettiest S-100 systems around (except for the choice of dark brown for the case color!).

But a useful alternative to the IMSAI/ Altair front panel was available from Jade Computer Products in 1981 (fourteen years ago): the **Jade Bus Probe**. Jade moved the front panel from the, er, front panel, to an extended S-100 card that elevated the display above the card cage. In addition, they updated the dis-

play to support interrupt and extended addressing. Imagine if you will (or look at the figures) a display of four lines of 24 LED's- **almost 100 indicators** - displaying status and data in real time! And keep in mind that at processor speeds of only a few megaHertz, it's not just a blur of activity, but a useful display of program activity! In addition, they provided some simple controls to deselect some of these displays, and some "logic bits" for wiring up additional signals for test and display.

Some background on the S-100 and the IEEE-696 Bus

It's been a few years since I've listed the S-100 bus lines, so let's begin with the original Altair list of S-100 lines. There are at least two flavors of S-100 buses: the original Altair 8800, front-panel supporting bus and, the IMSAI/ Cromemco bus that eliminated some of the Altair lines; and the IEEE-696 bus that expanded the address bus from 16 lines to 24, and allowed 16-bit bidirectional data transfers instead of 8-bit unidirectional transfers. I'll explain some of this later, but for those somewhat familiar with the bus already, the list following represents the lines on these two very similar buses.

S-100 BUS Standard

PIN	Altair	IEEE	PIN	Altair	IEEE
1	+8 V	+8 V	51	+8 V	+8V
2	+18 V	+18 V	52	-16 V	-16V
3	XRDY	XRDY	53	/SSW DSBL	GND
4	VI 0	/VI0	54	/EXTCLR	/S CLR
5	VI 1	/VI1	55	RTC(—)	/DMA0
6	VI 2	/VI2	56	/STSTB(—)	/DMA1
7	VI 3	/VI3	57	DIG1(—)	/DMA2
8	VI 4	/VI4	58	FRDY(—)	/SXTRQ
9	VI 5	/VI5	59	—	A19
10	VI 6	/VI6	60	—	/SIXTN
11	VI 7	/VI7	61	—	A20
12	—	/NMI	62	—	A21
13	—	/PWRFAIL	63	—	A22
14	—	/DMA3	64	—	A23
15	—	A18	65	—	—
16	—	A16	66	—	—
17	—	A17	67	—(/PHANT)	/PHANTOM
18	/STADSB	/SDSB	68	MWRT	MWRT
19	/CCDSB	/CDSB	69	/PS(—)	rsvd
20	UNPROT(T5)	GND	70	ROT(GND)	GND
21	SS	—	71	RUN	rsvd
22	/ADDDSB	/ADSB	72	PRDY	RDY
23	/DODSB	/DODSB	73	/PINT	/INT
24	phi2	phi2,phi0	74	/PHOLD	/HOLD
25	phi1	/PSTVAL	75	/PRESET	/RESET
26	PHLDA	PHLDA	76	PSYNC	PSYNC
27	PWAIT	rsvd	77	/PWR	/PWR
28	PINTE	rsvd	78	PDBIN	PDBIN
29	A5	A5	79	A0	A0
30	A4	A4	80	A1	A1
31	A3	A3	81	A2	A2
32	A15	A15	82	A6	A6
33	A12	A12	83	A7	A7
34	A9	A9	84	A8	A8
35	DO1	DO1|D1	85	A13	A13
36	DO0	DO0|D0	86	A14	A14
37	A10	A10	87	A11	A11
38	DO4	DO4|D4	88	DO2	DO2|D2
39	DO5	DO5|D5	89	DO3	DO3|D3
40	DO6	DO6|D6	90	DO7	DO7|D7
41	DI2	DI2|D10	91	DI4	DI4|D12
42	DI3	DI3|D11	92	DI5	DI5|D13
43	DI7	DI7|D15	93	DI6	DI6|D14
44	SM1	SM1	94	DI1	DI1|D9
45	SOUT	SOUT	95	DI0	DI0|D8
46	SINP	SINP	96	SINTA	SINTA
47	SMEMR	SMEMR	97	/SWO	/SWO
48	SHLTA	SHLTA	98	SSTACK	/ERROR
49	/CLOCK	CLOCK	99	/POC	/POC
50	GND	GND	100	GND	GND

These lines are listed as they appear on the S-100 cards: with the card's IC's tops facing you, and the bus connector below, the leftmost pin is pin 1, and the pin on the back of the card behind it is pin 51. If you turn the card to its back where all the IC pins stick out, pin 100 is on the left and pin 50 is behind it. S-100 cards are always the same width, 10.5 inches; and almost always the same height, 6.5 inches. The bus connector is 3.5 inches from the left of the card and 4.5 inches from the right edge of the card, and the pins are separated by .1 inch center-to-center.

Bus lines: Altair(IMSAI) vs. IEEE-696

Active low signals are shown as "/". Unused lines are "—". Reserved lines are "rsvd". The first column of signals are Altair signals; if the IMSAI signals are different they follow immediately in "()'; and the next column are the IEEE-696 signals. Signals on lines 24 and 25 refer to the Greek letter "phi". IEEE-696 signals separated by "|" represent the 8-bit and 16-bit data path signals, respectively.

The most important thing to remember about the S-100 bus, and to a lesser extent about the '696 bus, is that the original processor for this bus was an Intel 8080. All the signals and data lines on the Altair in particular were to support this processor. In addition, lines were needed to support front-panel operations. These two considerations define most of the signals of this bus. Consequently, one of the best references for figuring out these signals is an Intel 8080 processor book!

Let's quickly run through the more obvious signals. The **power lines** are the voltages as noted: +8, -8, +16, -16 volts. S-100 cards have on-board regulators to reduce these to more reasonable +5 and + and - 12 volts. The **address lines** are

A0, A1, and so on through A15 for the Altair; and through A23 for the '696. **The data lines** are DI0 through DI7 for data to the processor (like IN instructions); and DO0 through DO7 for data from the processor (like OUT instructions). Note for the '696 bus, these unidirectional lines are optionally bi-directional for 16 bits of data. The **vector interrupt lines** are VI0 through VI7, which when active force an interrupt of the current program and a jump to an interrupt support program at a specific address.

The status lines are output from the processor card, and show the status of the processor. These signals usually begin with "s" and are clustered at the right end of the card; such as SM1, SOUT, SINP, SMEMR, SHLTA, SINTA, /SWO, SSTACk. These are available from the 8080 processor as representing processor states as follows: instruction fetch, OUT instruction, IN instruction, memory read, processor halt, interrupt acknowledge, write, and stack (respectively). These are sneaky signals and should be reviewed more carefully than I've noted here.

The command lines are also output from the processor card, and more immediately control the operation of the bus. These lines usually begin with "p" and are clustered in the center of the card: PWAIT, /PWR, PDBIN, PINTE, PHLDA, PSYNC. These represent wait states (stretched read or write cycles), write (I/O or memory), I/O input, interrupt enabled status, processor hold acknowledge, and a "sync" signal for the start of a processor cycle (respectively). Additional command lines control the processor: /PINT, /PHOLD will force an interrupt or processor hold condition, respectively.

The reset lines /RESET will reset the processor and /EXTCLR will reset other bus devices, and **ready lines** PRDY and XRDY will force the processor into a HOLD state. **Clock lines** phi1 (removed from the '696) and phi2 are clocks synchronized to the processor's cycles; \CLOCK is a free-running 2 MHz clock signal. **The power-up clear line** /POC will be active (low) during the powering up of the bus.

The **bus disable lines** will disable groups of the above lines: /DODSBL disables the data out lines, /CCDSBL disables the command lines, /ADDSBL the address lines, /STDSBL the status lines. Not available on the '696 bus are the **front panel lines**: SS for single step, /SSDSBL to allow the front panel switches to send their data to the processor; MWRITE to control writes to the processor. A line "T5" **(pin 20)** on the IMSAI that the Altair calls "PROT" uses to unprotect memory (allow writes) AT THE MEMORY CARD and which is complimented by the Altair line PROT **(pin 50)** to disable memory writes AT THE MEMORY CARD (not the processor). A common problem with IMSAI's is the use of card that have **pins 20 and 50 grounded**: they will disable the front panel! Use a piece of paper to cover these pins, or cut the grounding traces to these pins.

There are a handful of lines used only by the Altair for who knows what: RTC, STSTB FRDRY, DIG1. Note that the '696 has reassigned them to **DMA channel priority signals** DMA1, DMA2, DMA3; and one of the **16-bit transfer signals** SIXTN. When a 16-bit capable board receives a /SXTRQ signal from a 16-bit capable processor card, the SIXTN signal is asserted and the processor turns the two 8-bit data busses into one 16-bit bus. Neat, eh?

Whew! Even a brief description of the S-100 bus takes a few hundred words, and I'm sure I missed some lines! If this seems like a busy bus, keep in mind the IBM-PC ISA bus is 68 pins, so this is not so many more. And, by the way, many of these signals have their counterparts on the PC bus as well. After all, the PC bus was designed for an 8088 processor which is not much different from the 8080 (he he..).

The Jade Bus Probe

In 1981, Jade Computer Products (of Hawthorne CA at the time) produced a card to display the state of almost all of these bus lines. As I mentioned initially, the Probe extends five inches above the standard S-100 card to display four rows of 24 LEDs, as illustrated elsewhere in this magazine. Each LED is connected to a 2-input logic NAND gate, part of a 74LS38 chip. This provides that one input of the gate, the signal to display, must be logically active (a 3 to 5-volt signal level); and the other input is tied in parallel with several other NAND gates to related signals, to permit a bank of common signals to be enabled or disables as the user desires. Bank selection of signals will be discussed after the signals and displays are described.

The Bus Probe circuit board display area has written titles to describe the signals displayed, and the display is organized as follows. The top row shows the **24 address lines**, with the most significant bits (A23, A22, etc) to the left and the least significant address bits (A2, A1, A0) to the left. The row below shows the **16 data lines** on the right, with the eight data in lines (DI7 through DI0) in the center and the eight data out lines (D07 through DO0) to the far right. If you review the IEEE-696 bus lines corresponding to these, you will note that the 16-bit data lines use the the DO lines for the upper data byte (D15 through D8), and the data out lines for the lower data byte (D7 through D0) so these '696 signals are displayed in the proper order. On the far left are the four '696 **DMA lines**, titled "TMA" from the IEEE-696 specification. Following these are the **bus disable lines** of /DODSB, /SDSB, /ADSB, and /CDSB for data, status, address, and control lines disabling respectively.

The next row of LED's displays (from left to right) the **eight interrupt lines, the utility lines**, and the **status lines**. The interrupt lines are as noted on the bus line list. The utility lines are, from the left, /INT, /NMI, /PHAN, /HOLD, /PWRFAIL, /ERROR, /SIXTN, and /MWRT. The eight status lines are, from the left, SMEMR, SM1, /SWO, SINP, SOUT, /SXTRQ, SINTS, and SHLTA. The status line display is also controlled by a switch (S1) that supports additional

Continued on page 28

TCJ Center Fold

Special Feature

All Users

JADE BUS PROBE

In this issue we actually do some old machine support, but stay tuned and look closer at what the Jade Bus Probe does for you. Herb covered most of the ideas in his Dr. S-100 column. Let me add that this concept is not limited to S-100 products.

The idea of monitoring what happens on your computers BUS is not new. The orginal IMSAI and Altair MITS both came with toggle switches and lights. It was absolutely important in the early machines, as this was the only way you had to input your programs.

We have come a long way since then, but monitoring what happens on the BUS is just as important. The problem is more difficult as the BUS speeds increase. Logic analyziers are in essence doing the same thing as the Jade BUS probe. The main difference is they can store the several hundred operations in their RAM and play them back for you one step at a time. The BUS probe instead has you stepping through your instructions one at a time and watching lights.

Now the lights do make it more fun to explore what is happening. Not letting that become the main idea, experience will allow you to get the feeling about whether your program is stopped or locked up. What I mean is you get to feel just what a proper working set of lights looks like. Their dancing around has a real specific pattern and you will get to know it so well that any changes become quickly a sign that you have programmed a wrong operation.

Can this design be used on other systems. I see no reason that we couldn't make systems like this for almost all types of computers in use today or from the past. The main problem is being able to single step the computer so you can see and read what is happening. There are many very wonderful ideas and designs that have been done to enable non-single stepping CPU's to do single stepping.

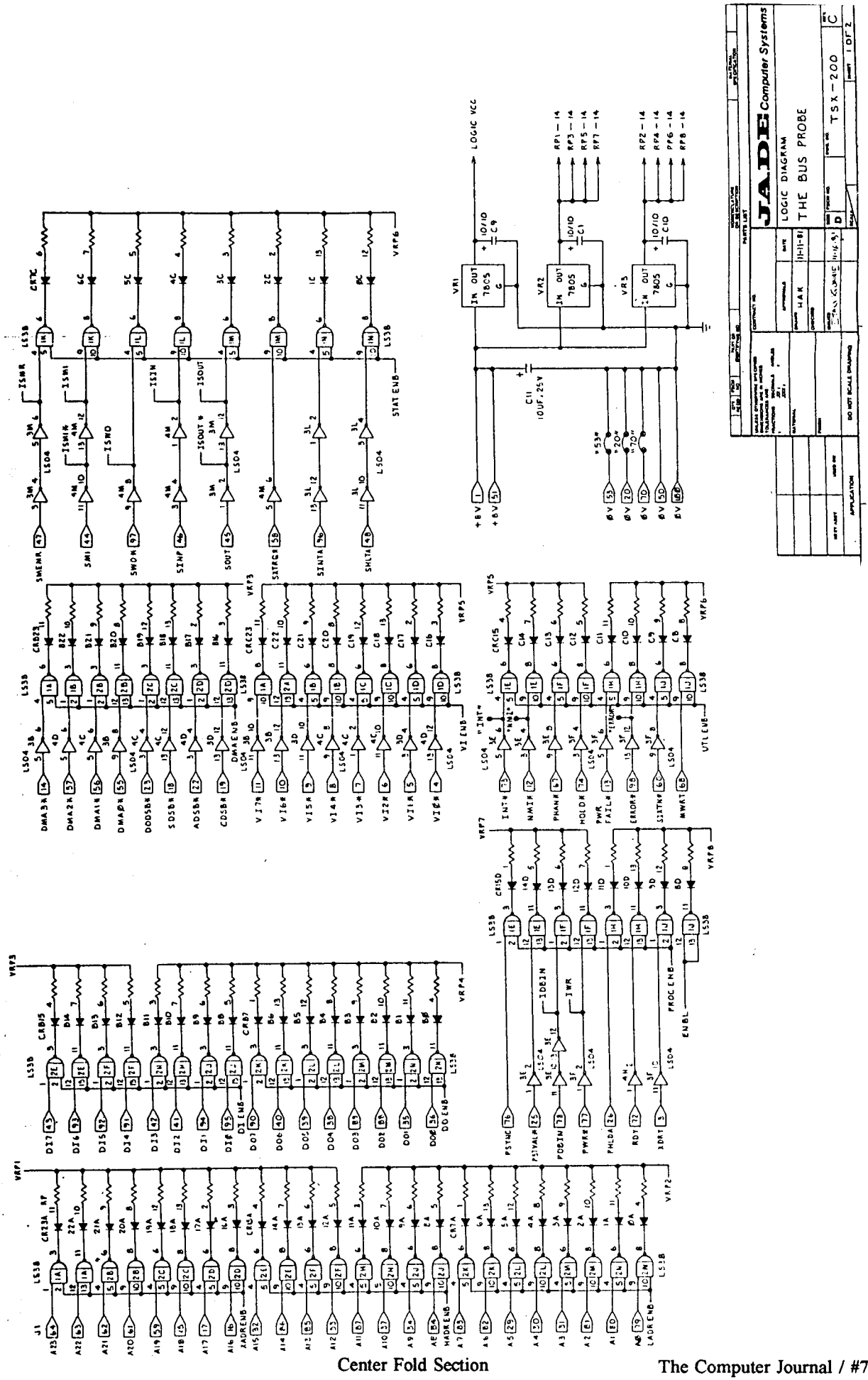
You must understand that some computers CPU's must run at full speed and any form of stepping through the program will end up in lost or erroneous data. I have even noticed that a couple of the newest chip do have single stepping and diagnostics built in, although that is all copyrighted and pretty much a trade secret. Or put an other way, how to do the operations and get the data is not possible for us normal people. If you have figured out some of these secrets and how to single step the more complex systems, how about an article, were all ears!

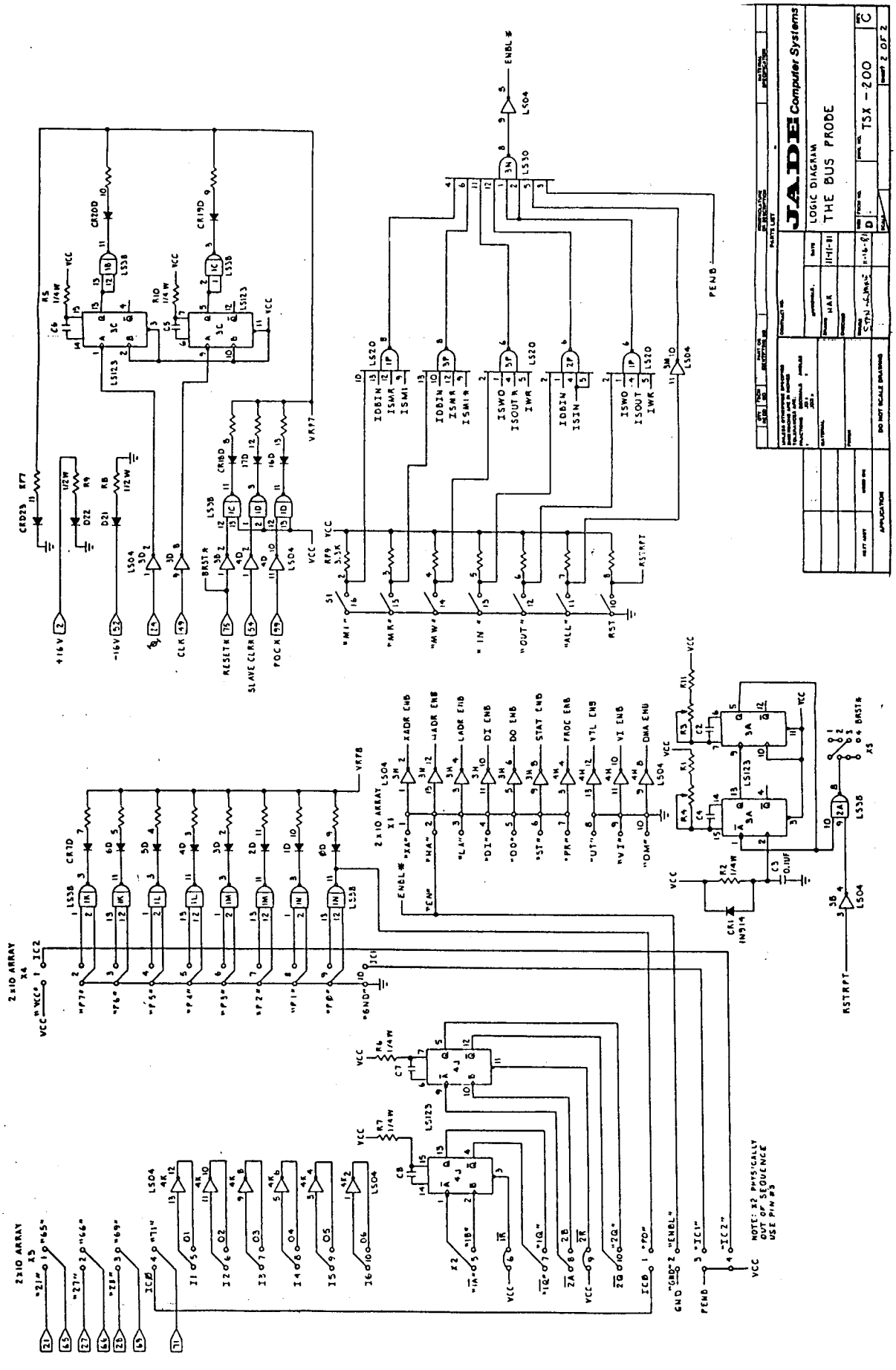
What IBM decided to do instead of a front panel or a diagnostic board like the Jade BUS Probe was to do POST processing. POST is a series of diagnostic programs run as the system is powering up. The results of these test are written out to I/O port 80 on the BUS. This means if you put an expansion card in the BUS and read the data from I/O address 80, it would be the results of the variuous tests. Should you have a failure some where, the test number would tell you what the problem is. The solution to the problem is left to you to still trouble shoot on your own.

I think a combination of the POST and BUS probe are the type of tools we at *TCJ* should consider building as a project. A long time ago I started laying out my own logic analyzer, pretty much like the one that came along for inserting into the PC BUS. It basically amounts to a small bit of RAM that just stores 1 or 0's as fast as it can. What my new idea is about is a combination POST storage reader, say store the last 20 post instructions. The last 20 to 100 BUS addresses and their data values would also be needed with the option to trigger either on or off at some preset address or data value.

So lets look over the Jade BUS Probe, see what it gave us, and how we might be able to do the same with fewer parts or a different design. What struck me the most, is just how many parts are used. You have to remember, that some device must be used to drive each LED, and there are 100 LEDs! This means that you need at least 100 buffers or data latches to isolate you from the BUS. There are 48 (14 pin) sockets used on the board. Now that is a lot of chips to stick on anything. And let me say that if ever there was a place for PALs or Programmable devices with many pins this is one of those cases. Even still, I think some of the largest parts are about 100 pins, so two parts would be needed, and lots of traces.

I really am serious about needing some ways to checkout systems. They also need to be simple and possible for others with limited skills to build. I can think of no other way of getting new people up to speed, than having them see what they are doing. The blinking lights can guide the new novice better than any other idea I can think of.


JADE Computer Systems
LOGIC DIAGRAM
THE BUS PROBE
TSX-200
HAK
11-11-81
DO NOT SCALE DRAWING
APPLICATION
LOGIC VCC
STAT ENB
VI ENB
UTL ENB
PROC ENB
ENBL
LADR ENB
HADR ENB
XADR ENB
DI ENB
DO ENB
DMA ENB
VRP1
VRP2
VRP3
VRP4
VRP5
VRP6
VRP7
VRP8
VR1
VR2
VR3
7805
C11
10UF,25V
RP1 - 14
RP3 - 14
RP5 - 14
RP7 - 14
RP2 - 14
RP4 - 14
RP6 - 14
RP8 - 14


JADE Computer Systems
LOGIC DIAGRAM
THE BUS PROBE
TSX - 200
SHEET 2 OF 2
NOTE: X2 PHYSICALLY OUT OF SEQUENCE USE PIN #5

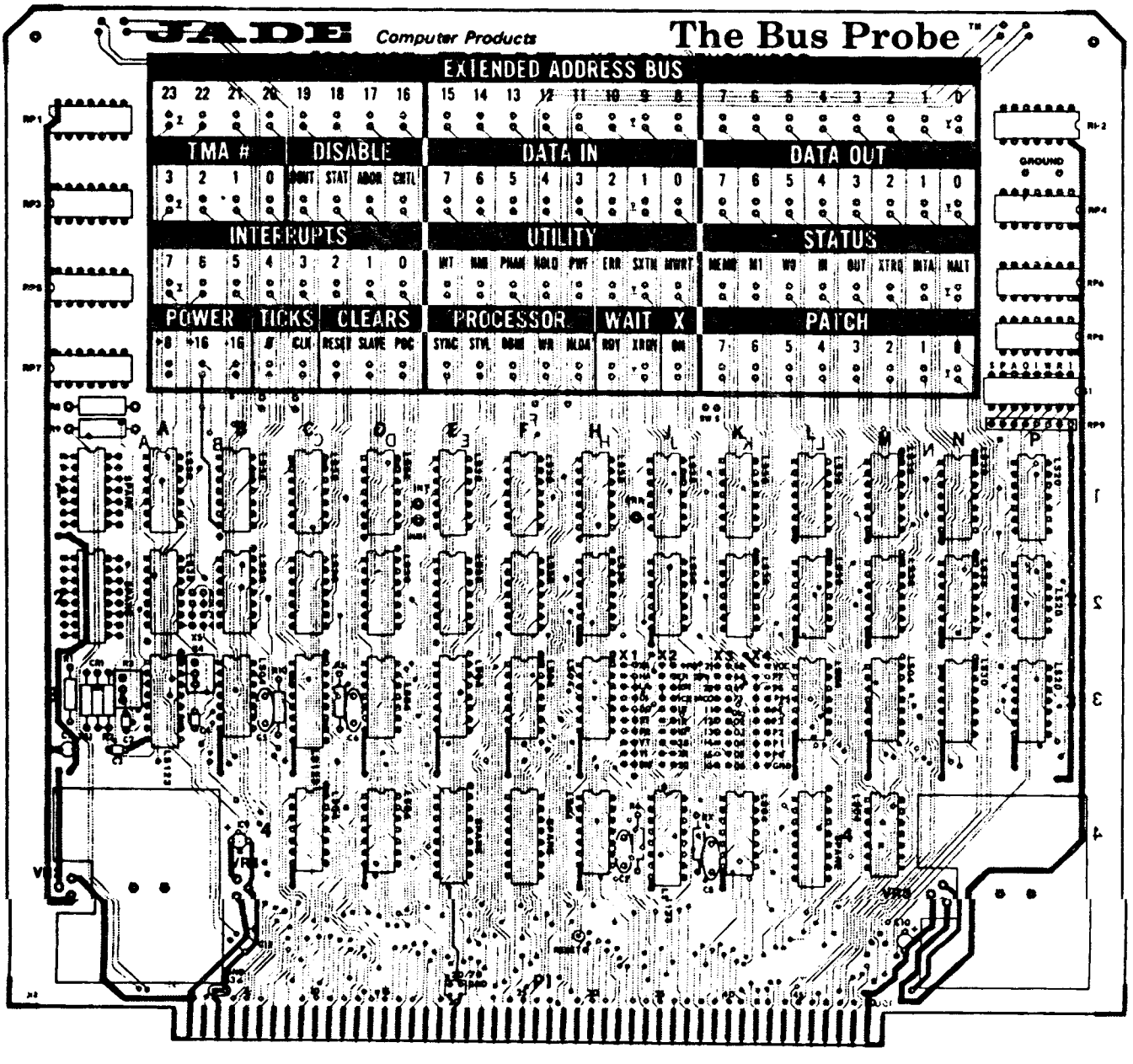
logic to select particular processor states to display by decoding the status lines for those states. Processor states are M1, memory read, memory write, I/O read, I/O write, "all", and reset.

The bottom row of LED are grouped as follows, from left to right: **power** (+8, +16, -16); **"ticks"** (a latched version of phi0 and CLOCK), **"clears"** (/RESET, /SLAVE CLR, /POC), **processor** (PSYNC, /PSTVAL, PDBIN, /PWR, PHLDA); **"wait"** (actually "ready": RDY, XRDY); **"X"** (a logically gated "power" signal explained later); and **"patch"**. The patch display are 8 LED's which have their NAND gates connected to a "patch" area on the board (block X4) for the user to connect to anything in their system.

Most of the above-mentioned groups of gated LED displays can be disabled. Their common gate enabling connections are routed to a selection block (X1) of jumpers that can be removed to disable that set of lines. You can leave them all enabled, to watch the entire state of the bus; or disable all but a selected set or two to monitor specific conditions.

There is also a block of jumpers (X2) to access a 74LS123, a dual monostable latch or "one-shots", which provide a pulse when their logical inputs are triggered by a logical event: inputs and outputs are to be wired by the user. The **non-defined bus lines** are connected to another set of jumpers (X3).

It is clear that Jade was very thoughtful in not only providing a controlled display of the S-100 bus, but also providing a few simple but useful bits of logic to display other signals from within your S-100 system.



PC TIME CLOCK

By Terry Hazen

Special Feature

XT Support

Improving Accuracy

PC Real Time Clock chips are notoriously inaccurate and can suffer from severe drift. Many of them seem to lose or gain time at about 1 to 15 seconds a day, with 5 or 6 seconds per day being typical (although the worst one I've heard of runs 1.5 hours a day fast!) The accuracy of the often-used Dallas Semiconductor DS1216E SmartWatch, for example, is specified as +/- 1 minute per month (about 2 seconds a day) at 25C (77F).

While these clock chips don't keep very accurate time, they usually drift at about the same rate from day to day as long as the ambient temperature doesn't vary wildly. If we make the assumption that the clock inaccuracy is relatively linear over time, we can improve things quite a bit by using some simple linear time correction when we read the clock.

The PC world already has a utility that does the job. ClockWrighT is an inexpensive commercial PC utility that performs linear time correction on IBM-compatible PC's, but a simple linear time correction algorithm can easily be implemented for almost any clock chip on almost any computer, including common 8-bit CP/M machines.

For example, I used linear time correction in SCLOCK, a ZCPR33+ Type 3 utility (loading and running at 8000h) that runs on 8-bit Z80 Ampro or Z180 Yasbec systems using SmartWatch chips for timekeeping. It reads and sets the SmartWatch, displays the corrected SmartWatch date and time, automatically adjusting for Daylight Savings Time, and can update the ZSDOS or BIOS clock with the corrected SmartWatch date and time.

Biff Bueffel's CLOCK utility also uses the same linear time correction algorithm to set the MM58167a chip in the Kenmore Clock, Anapro Clock for the Heath H89/90 or CDR Super 89 clock. CLOCK reads and writes directly to the MM58167 and automatically adjusts for leap years and for Daylight Savings Time.

If you are using an Ampro or Yasbec system or a Z80 system that uses the MM58167 clock chip, you can probably use one of these utilities directly. If you are using a different system, processor or clock chip and you want to write your own clock correction utility, read on...

Clock Accuracy Calibration

Before a correction algorithm can improve a clock's accuracy, that you must measure its accuracy over time and save the accuracy history data. You can do this by setting the clock to an accurate and consistent time standard, such as WWV. This must be done twice, several days to several weeks apart. Within reason, longer intervals result in better long-term correction accuracy. The first time you set the clock, you save the set time (as WWV1, for example.) The second time you set the clock, you save the clock time that is read just before the clock is reset (TIME2) and the new time to which it is reset (WWV2.)

Now that the clock is calibrated and you have the accuracy history, you can calculate a simple linear correction factor. The time baseline is:

BASE = WWV2 - WWV1

and the clock deviation over the baseline period is:

DEV = |TIME2 - WWV2|

which is saved as an absolute value. If the sign of DEV is positive, the clock is running fast. If it is negative, the clock is running slow. The sign is also saved so you can determine later on whether to add or subtract the time correction.

Given the time baseline and the clock deviation over the baseline period, the linear correction factor is:

LCF = DEV / BASE

which is the number of seconds of clock inaccuracy during the clock resetting baseline divided by the baseline. You can calculate LCF the second time the clock is set and save its value for later use each time the clock is read.

Each subsequent time the clock is read, CLKTIM, the required time correction is:

CORR = (CLKTIM - WWV2) * LCF

The corrected clock time is then:

CORR_TIME = CLKTIM +/- CORR

In other words, take the difference between the current time read by the clock and the last time it was set, and multiply that difference by the linear correction factor. If the clock is running fast, subtract the correction from the current clock time. If the clock is running slow, add the correction. Pretty straightforward stuff, but surprisingly effective.

Time Standards

You should consider your time standard before attempting tocalibrate your clock. Telephone company times may not always be reliable or even consistant. For best results, the clock should be set using a reliable time standard such as WWV. WWV (Colorado) and WWVH (Hawaii) broadcast time information on 2.5, 5 and 10 Mhz. WWV also broadcasts on 20 Mhz. 2.5, 5 and 10 Mhz are allocated by international agreement as time and frequency standard frequencies, and at least 15 other stations broadcast time data on those frequencies internationally.

WWV reports time as Coordinated Universal Time (UCT), formerly called Greenwich Mean Time, which is the time in Greenwich, England. The US Eastern Standard Time zone is 5 hours behind UCT. For example, 2 hours UCT is 21 hours (9pm) EST (2-5[+24]=21.) Pacific Standard Time is 8 hours behind UCT, so 2 hours UCT is 18 hours (6pm) PST.

Time and Date Format

You can simplify the time conversion code and speed up your calculations somewhat by using a reduced range of available correction and assuming the clock deviation will never exceed some fixed number of seconds between the times you need to reset the clock so that you can use simple 8 or 16-bit arithmetic. I chose to get wild and created a 32-bit binary "Julian Seconds" time and date format that can be easily manipulated using some 32-bit arithmetic routines from ZSLIB, one of the wonderful ZCPR3 series of Z80 routine libraries.

My systems all run ZCPR3 and ZSDOS, whose system times are usually kept in a convenient 6-byte "System Standard Time" (SST) format. The first two bytes of the SST are the BCD year, month and day expressed as a Julian date, which is defined as the number of days elapsed since January 1, 1978. Converting the current year, month and day to the Julian date is easily done with the BCD-to-Julian routine from the NZTIM library. Converting to Julian Seconds just involves converting days and hours to seconds with several simple multiplications. The resulting 32-bit Julian Seconds times can then be easily manipulated with the 32-bit addition, subtraction, multiplication and division routines from the ZSLIB library.

Clock Accuracy History

The accuracy history for each physical clock needs to be stored for later use by the correction algorithm. Because each clock chip will have a different accuracy history, only that specific clock chip can be corrected using its history data.

Some chips have several bytes of internal nonvolatile SRAM available to the user which can be used to store the history data. If like the SmartWatch, yours doesn't, the accuracy history can be stored in a one-record history file for each clock to be corrected, or in one record of the clock utility file itself. SCLOCK, for example, stores the history data in its first record, which is updated and rewritten each time the clock is set from WWV.

Bells and Whistles

While you're at it, you can also provide some other useful date and time services. Some clock chips already have DST correction available, but if yours doesn't, you can test the date and correct to Daylight Savings Time when it's in effect between the first Sunday in April and the last Sunday in October. While the SmartWatch already takes care of it, you can also correct your clock for leap year.

Results

I use several SmartWatch chips that vary considerably in accuracy. I've used the SmartWatches in my 8mhz Ampro and 18mhz Yasbec as my SCLOCK test beds. I calibrated both the Ampro and Yasbec SmartWatches by seting them just after Thanksgiving and again on New Years, 35 days apart. In the 42 days since then, the Ampro SmartWatch has gained 108 seconds and the Yasbec SmartWatch has lost 97 seconds, but the corrected times for both SmartWatches remain within 1 to 2 seconds, which is within the normal SmartWatch read-jitter. Your results will probably vary by chip type and ambient temperature variation.

References

1) DS1216E SmartWatch data sheet, Dallas Semiconductor (1-800-336-6933.)

2) ClockWrighT, Barberry Hill Software and Engineering, 26 Barberry Hill, Woodstock VT 05091-1269.

3) Jeff Bachiochi, "A Comparison of Real-time Clocks", The Computer Applications Journal, Issue #52, November 1994.

4) Michael A. Lombardi, "Keeping Time on Your PC", BYTE, October 1993.

5) SCLOCK20.LBR, CLOCK25.LBR (both libraries include the source code) and the ZCPR3/ZSDOS REL library series, including, but not limited to, LIBS45A.LBR (containing SYSLIB, ZLIB, VLIB and DSLIB), ZSLIB36.LBR, NZTIM12.LBR and PDAT13.LBR (source code for these libraries is included or is available in separate libraries) are available on Znode #2, Los Angeles, CA (310)670-9465 or your favorite local Znode. If you're a Z80 or Z180 assembly language programmer and haven't taken advantage of these rich sources of well-tested Z80 REL routines, do yourself a favor and check them out!

PC Security System

By Michael Krabach

Special Feature

XT Support

XT and Basic

While I have not engaged in 8 bit machines lately, I still keep them for history and they all tell a story. The best is the SB180 which is built into a WWII range finder box. The B&W monitor is built with plexiglas and aluminum perf board. It would fit right into a 1930's Buck Rogers movie.

Any way... my latest project was building an alarm system using a I/O board given to me by a friend. The Micromint 68HC11 was to be the core of the alarm system, but I was going to have to build TTL interfacing and I didn't have the time before I was to leave for vacation. So the initial alarm system was built around the PC and the Metrabyte I/O board. When I got back, the system was cleaned up, debugged and set up as a two tier system to prevent false alarm. After a year of no false alarms and many enhancements the system is pretty much finished.

While this is not an 8 bit project, it is a hardware/software project and it sure has taught me alot of subtle tricks in designing a polling program. I have used PowerBasic 3.0 which sure is nice compared to QuickBasic. It has bit operations which were necessary for the kind of control I needed for the I/O board. Feel free to run with the program, and see if you can play Break-In and not get caught.

The Hardware

The hardware consists of a Keithly Metrabyte 8 channel isolated relay input/output interface board (Model PDISO-8). This board plugs into the ISA bus of a typical IBM type PC. There are many similar I/O boards that are modeled after the Metrabyte configuration. Any of the similar interface I/O boards can be used as long as it conforms to the method of accessing the I/O board. No special drivers are required. The board occupies 4 consecutive addresses in the PC I/O address space, of which only 2 are actually used. The base address is set up with a dip switch on the I/O board and again defined in the control program. It can be any port address from 100 to 3F8. While port 300 hex is the default address shipped with the board, I used 280 hex because when the PC boots up one of the bits goes high at address 300 hex and the alarm bell sounds.

The I/O board consists of 8 individually optically isolated inputs which can accept AC or DC from 5 to 24V rms. These inputs are not TTL/CMOS compatable, so they are easily operated with on/off switches such as magnetic reed switches or pressure contact micro switches.

The 8 individual output relays are capable of handling 3A at 120 vac or 28V DC resistive load. Five of the relays are bipolar (type C) and can be used as normally open or normally closed, while the other 3 are normally open (type A) only.

Both the isolated input and relay output ports are controlled with an 8 bit byte with each bit corresponding to each input or output. The bits are controlled by writing or reading directly to the port. The base address (280h) controls the output which can be either read or write. The base address plus 1 (281h) is the input which is only a read address.

The I/O board itself has a DB-37 pin connector that is exposed at the back of the PC when plugged into the ISA bus. The pins are grouped logically so that a 37 wire ribbon cable can be used to connect to a IDS type dual row standard 1/10" pinned connector which then connects easily to a standard 1/10" perforated circuit board. For this system, standard off-the-shelf components were used to design a small adapter board to accept the input wires from the sensors and direct the output wires to the light control and alarms.

INPUT SENSORS

Each of the 8 input lines consists of a 5 volt circuit with a LED to indicate when the circuit is closed. Closed is considered the normal non-alarm mode since it gives a positive indication that an alarm sensor is operable. When all input LEDs are on, the system is active and in a non-alarm condition. The LSD (least significant digit) bit is controlled by a micro switch attached to the kitchen deadbolt lock. When the door is locked the system is activated and all sensors circuits are continually scanned for an open circuit which would indicate a house break in.

A local control panel which contains indicator LEDs, has 8 local three way toggle switches which are in parallel with the 8 individual sensor circuits. These are used to deactivate any single sensor or to open the sensor circuit to test the alarm system.

The actual sensors consist of two types. The windows and doors are wired in series using trip wires on the windows and magnetic sensors on the doors. The trip wires are such that they will be broken by either opening or breaking the windows. Since the downstairs win-

dows are never opened, this system works ok. Upstairs the windows are on magnetic sensors. The rooms of the house are scanned by passive infrared dectectors (PIRs).

The PIRs are constructed from common Zenith PIR wall switches. The PIR, a 2 receptacle wall socket and a 120 VAC (Radio Shack) relay are packed in a common plastic wall junction box such that when the PIR is actuated the relay is opened and the signal is broken with the alarm controller. The Zenith units require a load to operate so a small 40 watt reflector bulb is installed in one of the receptacle sockets. The PIR can be adjusted for daylight intensity so that they only operate at night or all the time. A side benefit is that they function as automatic night lights. The alarm is not tripped because the inner PIRs only come into play when the outer window or door perimeter is broken. This double layer of secuity prevents false alarm.

OUTPUT CONTROL

The five bipolar output relays control 5 vdc which is used to power solid state 120 vac relays that in turn control 4 house lights. The fifth relay controls another solid state relay that activates the house stereo system to play a message to anyone in the house at the time of the alarm. This can be used for psychological warfare against the burglar in the mildest sense. It could also be used to control an automatic phone dialing machine if desired. The other three output relays are used to supply 12 vdc to control a bell and two separate 2-tone power horns. Since the 1st five bits control bipolar relays, the normally closed contacts supply 5 vdc to individial circuits that have LEDs. In this case the LEDs are on when the respective house lights (and stereo system) are off. There is no LED indication for the three alarms operated by the type A single pole relays.

X-10 MODULES

The output control which is normally handling the house lights and alarms via a hardwired circuit, also operates up to 8 X-10 light or appliance modules. The operation is done with a Circuit Cellar PLIX module (Power Line Interface) and an X-10 TW523 or PL513 module which sends the signals to the X-10 light modules. The unit is attached to any standard unidirectional parallel printer port on the PC. This unit is available from Micromint, Inc. as a Demo Kit. This could be used as input if desired, although I prefer using hardwired input signals.

The system will function without the PLIX controller.

TEMPERATURE MONITORING

Temperature is monitored through the PC's Game Port. Since a game port paddle is a pseudo analog input that measures dual axis 100K ohm potentiometers, it is simple to install 100K ohm thermistors in place of the pots. Calibration is performed with a decade resistance box and a Steinhart & Hart calibration equation. The result is a cheap temperature monitor. The accuracy is about 2 degrees, but so far (for some reason not understood,) it tends to drift several degrees over several months. Sometimes it even drifts back! Since the game port has two inputs, 4 temperatures can be measured, one for each floor of the house and one outside.

SOFTWARE

The control program basically runs in a loop that continually scans all the input sensors. If any of the sensors indicate a non-normal condition, some or all of the alarms are sounded. While scanning the sensors it also compares the time since midnight against the light control setpoints. If the time is within a setpoint window, a light is either turned on or off.

OPERATION

The easiest way to see the operation of the system is to run the actual software. The program ALARM12.EXE and ALARM12N.EXE (for systems with no coprocessors) is designed so that all the input functions can be emulated by the keyboard. All the functions can operate in real time or compressed time. Compressed time condenses 9 hours into about 2 minutes. The system has two help screens (Alt-H and Alt-J) to explain the operation.

The system does not use a key pad to arm and disarm the alarm system. Instead a key switch on the kitchen door deadbolt lock is used to arm the system when the door is locked. Unlocking the door deactivates the alarm system. A time delay is built into the keyswitch to prevent false alarms. The perimeter circuit is also associated with the keyswitch so that the keyswitch wires can not be cut without causing an alarm condition.

If you have a game port on your PC with a joy stick plugged in, you will see the temperature monitoring function, although with unrealistic results. For the temperature function to operate you must have the file TEMP.INI, which supplies thermistor calibration data, in your default directory. All temperature data will be saved in an ASCII file TEMP.LOG.

The program will also record all input and output, (optionally light control) in an ASCII file HOUSE.LOG. Each time a sensor event occurs a menu screen dump will be made to a binary file MENU.LOG which can be viewed later with SEEMENU.EXE.

The purpose of the log files is to not only preserve what happens during a break-in, but to allow someone to dial into the house and view the status of the control and alarm system. In my system the alarm computer is connected to a second computer via peer-to-peer LAN which has a modem. An alternative would be to use a multitasking system such as OS/2 for both the alarm system and the dial in capabilities. Under any system that takes time slices on the CPU, the game port will not operate as consistently.

These programs written by:
Michael Krabach
747 Nate Whipple Hwy
Cumberland, RI 02864
ph 401-333-5350

Available on *TCJ* BBS as Alarm12.zip.

First few pages of Alarm12.BAS

```
'Home control and alarm system using a Metrabyte PDISO-8,
' an 8 channel isolated relay input/output interface PC board.
' With the following revisions;
' an autodialer or speaker warning using control bit 4,
' with enhanced software testing mode,
' with IR sensors dependent on activating window/door sensors,
' with three seasons for timing lighting control,
' with autoscaling test mode,
' with remote access to copy of display menu,
' with auto select season,
' with 4 sensor temperature monitoring thru game port,
' with optional event logging to printer.
' with individual thermistor calibration
' added PLIX controller for X-10 modules, rev 12, 4-6-95

'Copyright, (c) Michael Krabach, 1994
'Written in PowerBASIC 3.1

$EVENT OFF
rev$="rev 12a"
revdate$="4-16-95"

'The I/O output is attached to the interface board as:
' bit 0 living room
' bit 1 kitchen
' bit 2 back room
' bit 3 bedroom
' bit 4 aux alarm, eg. autodialer or speaker warning
' bit 5 bell
' bit 6 horn 1
' bit 7 horn 2

'The I/O input (sensors) is attached to the interface board as:
' bit 0 kitchen door keylock microswitch
' bit 1 downstairs windows and doors using magnetic reed switches
' bit 2 upstairs windows using magnetic reed switches
' bit 3 upstairs spare room, and bedroom passive IR sensors
' bit 4 living room passive IR sensor
' bit 5 dining room passive IR sensor
' bit 6 kitchen passive IR sensor
' bit 7 back room passive IR sensor

'Note any variable dealing with time has been given a prefix t.

DEFINT a,c,d,e,f,h,i,l,n,p,q,s,u,w,x,y  'integer variables
DEFSNG o,t,r
'single precision
DIM b(4), m(4)
'for ohms from game port
DIM rcoefa(4), rcoefb(4), rcoefc(4)    'for steinhart calibration
DIM degF1(4), degF2(4), degF3(4), degF4(4)  'for summing temps

'Set up control setpoints for the seasons.
GOSUB checkseason                      'find out what season it is

setseason:
season$=temp$          'transfer string
SELECT CASE season$

'User defined setpoints and delay intervals for normal light control.
' Some of these setpoints may be advanced by the NEWTIME routine.
' The following setpoints are in 24 hour time (hr.min format).
' Kitchen, backroom, and living room lights are operated twice daily.
' The computer clock needs to be adjusted for Daylight Savings Time.

'Winter setpoint schematic: (EST)

' 12 hr clock 3  4  5  6  7  8  9 10 11 12  1am
' 24 hr clock 15 16 17 18 19 20 21 22 23 24 01 hrs
' normal mode |...|...|...|...|...|...|...|...|...|...|...|...|
'kitchen       ?-----------?   ?--?
'back rm         ?-------?   ?----------d
'living rm         ?-----------? ?--d
'bed rm                        d--?
'clear lights                    x
' test mode   |...|...|...|...|...|...|...|...|...|...|...|...|
'  seconds   0 10 20 30 40 50 60 70 80 90 100

CASE "-Winter-"                    'Winter setpoints:
tkitchon=16.00                     'kitchen on 4:00 pm
tkitchoff=19.30                    'kitchen off 7:30 pm
tkitchon2=21.00                    'kitchen2 on 9:00 pm
tkitchoff2=21.45                   'kitchen2 off 9:45 pm
tbackrmon=16.45                    'backroom on 4:45 pm
tbackrmoff=18.30                   'backroom off 6:30 pm
tbackrmon2=20.15                   'backroom2 on 8:15 pm
tbackrmoff2=23.00                  'backroom2 off 11:00 pm
tlivrmon=18.00                     'living rm on 6:00 pm
tlivrmoff=21.15                    'living rm off 9:15 pm
tlivrmon2=22.00                    'living rm2 on 10:00 pm
timeclear=1.15                     'turn off any extraneous lights at
1:15 am
tvariation=0.30                    'max random variation to setpoint in
NEWTIME
GOTO cont

'Spring setpoint schematic: (EST)

' 12 hr clock 3  4  5  6  7  8  9 10 11 12  1am
' 24 hr clock 15 16 17 18 19 20 21 22 23 24 01 hrs
' normal mode |...|...|...|...|...|...|...|...|...|...|...|...|
'kitchen       ?-----------? ?----?
'back rm          ?-------? ?-d
'living rm        ?-------?  ?----d
'bed rm                        d--?
'clear lights                    x
' test mode   |...|...|...|...|...|...|...|...|...|...|...|...|
'  seconds   0 10 20 30 40 50 60 70 80 90 100

CASE "-Spring-"                    'Spring setpoints
tkitchon=17.30
tkitchoff=20.15
tkitchon2=21.00
tkitchoff2=22.15
tbackrmon=19.45
tbackrmoff=21.45
tbackrmon2=22.30
tbackrmoff2=23.00
tlivrmon=18.45
tlivrmoff=20.30
tlivrmon2=21.30
timeclear=1.15
tvariation=0.30
GOTO cont
```

```
'Summer setpoint schematic: (EDT)
'-------------------------------------------------------------------
' 12 hr clock 3  4  5  6  7  8  9 10 11 12  1am
' 24 hr clock 15 16 17 18 19 20 21 22 23 24 01 hrs
' normal mode |...|...|...|...|...|...|...|...|...|...|...|...|
'kitchen              ?—? ?——?
'back rm               ?——? ?-d
'living rm             ?—? ?-d
'bed rm                   d—?
'clear lights                  x
' test mode  |...|...|...|...|...|...|...|...|...|...|...|...|
'  seconds   0 10 20 30 40 50 60 70 80 90 100
'-------------------------------------------------------------------

CASE "-Summer-"          'Summer setpoints
tkitchon=20.00
tkitchoff=20.45
tkitchon2=21.30
tkitchoff2=22.45
tbackrmon=20.15
tbackrmoff=21.45
tbackrmon2=22.30
tbackrmoff2=23.00
tlivrmon=21.15
tlivrmoff=22.00
tlivrmon2=20.30
timeclear=1.15
tvariation=0.30
GOTO cont

'Fall setpoint schematic: (EDT)
'-------------------------------------------------------------------
' 12 hr clock 3  4  5  6  7  8  9 10 11 12  1am
' 24 hr clock 15 16 17 18 19 20 21 22 23 24 01 hrs
' normal mode |...|...|...|...|...|...|...|...|...|...|...|...|
'kitchen              ?—? ?——?
'back rm               ?——? ?-d
'living rm             ?——? ?—d
'bed rm                   d—?
'clear lights                  x
' test mode  |...|...|...|...|...|...|...|...|...|...|...|...|
'  seconds   0 10 20 30 40 50 60 70 80 90 100
'-------------------------------------------------------------------

CASE "-Fall-"     'Fall setpoints
tkitchon=19.15
tkitchoff=20.15
tkitchon2=21.00
tkitchoff2=22.15
tbackrmon=20.15
tbackrmoff=21.45
tbackrmon2=22.30
tbackrmoff2=23.00
tlivrmon=20.00
tlivrmoff=21.15
tlivrmon2=22.00
timeclear=1.15
tvariation=0.30

cont:                                  'For all seasons
'User defined delays in seconds referenced to a previous setpoint.
tdelaylivrmoff2=10  '2nd living rm off 10 seconds after 2nd back room off
tdelaybedrmon=8         'bedroom on 8 seconds after 2nd living room off
tdelaybedrmoff=60       'bedroom off (minimum of) 1 minute after light on

END SELECT

'User defined alarm delays in seconds for normal operation.
twaitbell=5          'delay between event and bell sounding
twaithorn1=10        'delay between event and bell and horn sounding
twaithorn2=15        'delay between event and all three alarms sounding
twaitaux=60        'delay between event and aux alarm (allow pir's to clear)
twaitwinbad=60     'delay between event and ignoring window sensors
talmoffdelay=300   'delay before shutting off alarms after cleared (5 min)
tdowncount=60      'initial starting down-counter for bell repeat sequence
                   ' this number only valid with delay 1 in scanloop
                ' note since integer, downcount can only can be <=32000
tdelaytemp=900     'read house temperatures every 15 minutes,
                   ' to be recorded at 4x interval

'-------------------------------------------------------------------
'Test mode setpoints generated from above seasonal setpoints.
' The setpoints are adjusted to seconds from the start of the test.
' This sequence simulates the real control cycles in compressed time.

testkitchon=INT(makesmall(thrtosec(tkitchon)))       'kitchen on (F2)
testkitchoff=INT(makesmall(thrtosec(tkitchoff)))     'kitchen off
testkitchon2=INT(makesmall(thrtosec(tkitchon2)))     'kitchen on (F2)
testkitchoff2=INT(makesmall(thrtosec(tkitchoff2)))   'kitchen off
testbackrmon=INT(makesmall(thrtosec(tbackrmon)))  'back room on (F3)
testbackrmoff=INT(makesmall(thrtosec(tbackrmoff)))   'back room off
testbackrmon2=INT(makesmall(thrtosec(tbackrmon2))
                                                  'back room on (F3)
testbackrmoff2=INT(makesmall(thrtosec(tbackrmoff2)))  'back room off
testlivrmon=INT(makesmall(thrtosec(tlivrmon)))       'living rm on (F1)
testlivrmoff=INT(makesmall(thrtosec(tlivrmoff)))     'living rm off
testlivrmon2=INT(makesmall(thrtosec(tlivrmon2)))     'living rm on
testtimeclear=INT(makesmall(thrtosec(timeclear)+86400))
                                                  'turn off any lights still on
testvariation=INT((tvariation/0.06)+1)            'max random variation to
setpoint in NEWTIME

'User defined test delay seconds referenced to a previous setpoint.
testdelaylivrmoff2=3  '2nd living rm off after 2nd back room off (F1 re F3)
testdelaybedrmon=3        'bedroom on after 2nd living rm off (F4 re F1)
testdelaybedrmoff=8       'bedroom off after light on (F4 re F4)

'User defined alarm delays in seconds for the test mode.
testwaitbell=5       'delay between event and bell sounding
testwaithorn1=10  'delay between event and bell and horn sounding
testwaithorn2=15  'delay between event and all three alarms sounding
testwaitaux=25       'delay between event and aux alarm
testwaitwinbad=25'delay between event and ignoring window sensors
testalmoffdelay=5  'delay before shutting off alarms after cleared
testdowncount=5   'initial starting down-counter for bell repeat sequence
                  ' this number only valid with delay 1 in scanloop
testdelaytemp=7   'read house temperatures every 7 seconds,
                  ' to be recorded at 4x interval

'-------------------------------------------------------------------
'Additional definitions.

bbase=&h380      'base port address for PDISO-8 I/O card
         ' note that base &h300 comes up on reboot with bell on!

%AX=1            'Equates for temperature measurments
%BX=2
%CX=3
%DX=4

FUNCTION thrtosec(thrs)  'convert hrs to seconds after midnight
        thrtosec=INT(thrs)*3600+FRAC(thrs)/0.6*3600
END FUNCTION

FUNCTION makesmall(tsecond)
                         'convert setpoints to interval for test mode
        makesmall=(tsecond-54000)/360
END FUNCTION

'-------------------------------------------------------------------
```

```
'Set up sounds for alarms.
PLAY "mb"                    'play in background thru buffer
bellsound$= "t240 l64 O4cc.cc.cc.cc."      'a bell sound
horn1sound$="t200 l8 O3gc"                 'a horn sound
horn2sound$="t200 l8 O3a-d-"               'another horn sound
dialsound$="t200 l32 O3dgdgdgdg"           'a phone ringing

'-------------------------------------------------------------------
'Set up function keys for key trapping.
KEY 15, CHR$(&h08, &h02, &h70)  'alt plus !1 KEY
KEY 16, CHR$(&h08, &h03, &h70)  'alt plus @2 KEY
KEY 17, CHR$(&h08, &h04, &h70)  'alt plus #3 KEY
KEY 18, CHR$(&h08, &h05, &h70)   'alt plus $4 KEY
KEY 19, CHR$(&h08, &h06, &h70)   'alt plus %5 KEY
KEY 20, CHR$(&h08, &h07, &h70)   'alt plus ^6 KEY
KEY 21, CHR$(&h08, &h08, &h70)   'alt plus &7 KEY
KEY 22, CHR$(&h08, &h09, &h70)   'alt plus *8 KEY
KEY 23, CHR$(&h08, &h23, &h70)   'alt plus H key
KEY 24, CHR$(&H08, &h0b, &h70)   'alt plus )0 KEY
KEY 25, CHR$(&H08, &H24, &H70)   'alt plus J key
KEY 26, CHR$(&H08, &H19, &H70)   'alt plus P key

ON KEY (1) GOSUB testliving              'turn on living room light
ON KEY (2) GOSUB testkitchen             'turn on kitchen light
ON KEY (3) GOSUB testbackrm              'turn on back room light
ON KEY (4) GOSUB testbdrm                'turn on bedroom light
ON KEY (5) GOSUB testaux                 'turn on aux alarm
ON KEY (6) GOSUB testbell                'turn on bell
ON KEY (7) GOSUB testhorn1               'turn on horn 1
ON KEY (8) GOSUB testhorn2               'turn on horn 2
ON KEY (9) GOSUB zero          'turn off all output, and reset flags
ON KEY (10) GOSUB shutdown               'exit program
ON KEY (11) GOSUB showlog                'show the log file
ON KEY (12) GOSUB settest                'setup local setpoint test
ON KEY (13) GOSUB setfakesensor          'fake sensor test setup
ON KEY (14) GOSUB loglights
ON KEY (15) GOSUB bit0on       'toggle the bit 0
ON KEY (16) GOSUB bit1on       'etc.
ON KEY (17) GOSUB bit2on
ON KEY (18) GOSUB bit3on
ON KEY (19) GOSUB bit4on
ON KEY (20) GOSUB bit5on
ON KEY (21) GOSUB bit6on
ON KEY (22) GOSUB bit7on
ON KEY (23) GOSUB help
ON KEY (24) GOSUB deletelog   'delete the house.log file from menu
ON KEY (25) GOSUB help2        'spare help file
ON KEY (26) GOSUB printevents

KEY (1) ON
KEY (2) ON
KEY (3) ON
KEY (4) ON
KEY (5) ON
KEY (6) ON
KEY (7) ON
KEY (8) ON
KEY (9) ON
KEY (10) ON
KEY (11) ON
KEY (12) ON
KEY (13) ON
KEY (14) ON
KEY (15) ON
KEY (16) ON
KEY (17) ON
KEY (18) ON
KEY (19) ON
KEY (20) ON
KEY (21) ON
KEY (22) ON
KEY (23) ON
KEY (24) ON
KEY (25) ON
KEY (26) ON
'-------------------------------------------------------------------
'Reset all flags and variables.

GOSUB getini     'get calibration data for temperature measurements

CLS                          'start with clean screen
GOSUB findplix

'Also is restart point for testing light control.
restart:              'reset individual sensor event flags

flag1=0               'for sensor on bit1 (bit0 is not a sensor)
flag2=0               'for sensor on bit2
flag3=0               'for sensor on bit3
flag4=0               'etc.
flag5=0
flag6=0
flag7=0

flagtest=0            'used to initiate light test only once while in scanloop
talm1a=0              'reset alarm level 1 event start
talm1b=0              'reset alarm level 1 event current time
talm2a=0              'ditto for alarm level 2
talm2b=0
talm3a=0              'ditto for alarm level 3
talm3b=0

twait=0               'zero time delay to shutoff the alarms
tevent=0              'zero common time for all alarm levels for twait test
sensor2=255           'set to assume all circuits are closed

'Set up base light control times for real and test mode.
IF localtest=0 THEN GOSUB tbase ELSE GOSUB testtbase

GOSUB zero       'reset output ports and generate light control times
count5=5       'Specific reset to prevent 5 sec delay from causing alarm
                '  when door is open, while booting computer.

GOSUB readscreen          'save a copy of the display screen.

'===================================================================
'Start of scanning loop

$EVENT ON        'start key event checking, during compiling

scanloop:

DELAY 1              'timing delay for accurate test timing and cyclebell
          'this also helps prevent bad cursor placement on screen

'Save copy of display menu to screen.log after a sensor or control event.
IF oldcount<>count OR oldcount2<>count2 THEN
          GOSUB readscreen
oldcount=count
oldcount2=count2
END IF
'Save copy of house temperatures to temp.log every delaytemp interval.
IF loopcount >= delaytemp THEN
          GOSUB readtemp
          loopcount=0
END IF
INCR loopcount
IF logflag=1 THEN GOSUB readlog  'show log only at this point in code
IF flagtest=1 THEN GOTO restart    'restart if doing local light test
'Screen menu for testing the light and alarm control.
COLOR 9
LOCATE 1,5
PRINT "====== HOME CONTROL AND ALARM SYSTEM ========"
```

End of listing for TCJ.

Regular Feature

68xx/68xxx Support

C & Assembly

Small System Support

By Ronald W. Anderson

Good grief, I am so far ahead with these lessons that they will be old when you get to read them. I will keep them and do some touch up before they are mailed to TCJ. First let's look at the next lesson in C programming.

C Tutorial #5

This time we have a few little items to clean up. First let's look at functions again briefly. C functions can "return" a value to the calling function. In some cases it is obvious that this is the case. For example, how would a function like getchar() be of any use if you didn't get some value back from it.

```
char ch;

ch = getchar();
```

When you define a function you generally have to declare the type of the value that is returned:

```
double sin(angle);
double angle;
        {
        .....
        }
char getchar();
```

The first is an example of a function that has a value passed to it and also returns a value. The trigonometric functions are defined in math.h, part of the standard library. All arithmetic in C is done in double format. If you use a function that returns a double but assign the result to a float type, half of the digits are simply thrown away!

The second example is a function that returns a character. If a function returns an integer you don't need to declare that fact explicitly since it is the default, but you may and it is a good idea since it is a double check that you have declared the type of the return value of all functions in your program. Since no return type means integer, ANSI C wants you to tell it that your function doesn't return any value by using the keyword "void".

Many of the standard library functions return a value (which, incidentally you can use or you can ignore). Some of them return a value that can be used to indicate the success or the failure of the function to do what was requested. For example there is a function called malloc() which allocates memory to a program. Such memory can be used and deallocated within the program so it is a form of memory management. If you want to allocate a block of memory, malloc returns the address of the first byte of the block. If for some reason malloc can't allocate the requested memory it returns 0 or NULL. You can test for null to see if the function was successful.

A more commonly used function is one to open a data file. It also returns an address in memory where the file handling takes place. If the file couldn't be opened (e.g. you tried to open a file for read, but it didn't exist), the function returns NULL. The file opening function is "fopen()" It requires two parameters, a filename and a mode. infile = fopen("data.dat","r"); This opens a file named data.dat for reading (use "w" for writing). If it is successful it returns an address that is assigned to "infile". Infile is a pointer to a file, a topic we will cover later. We can detect the error and exit the program. If I am in a hurry doing a one of a kind quick filter program for my own use, I sometimes assume there won't be any errors, and don't test for them. If the program is going to be used for a while, I will add the error testing and handling code as well.

A "filter" program by the way, is one that does some transformation on an input file and writes the result to an output file. A filter may be as simple as a program that converts all upper case alfa characters to lower case or vice versa. It might convert all tab characters to three spaces. Perhaps it would find linefeed carriage return combinations and delete the linefeeds, etc. A filter is not necessarily limited to working on a text file. It could just as well read a file of floating point numbers and convert them from one format to another.

```
infile = fopen(data.dat,"r");
if (infile == NULL)
{
    printf("Can't open the input file\n");
    exit(1);
}
```

Exit is another function in the standard library. In some cases when multiple programs are being run returning the value 1 for an error or 0 for a successful completion of a program is useful to the operating system. ("Some cases" basically means UNIX systems).

The "return" mechanism only allows for one value to be returned. If a function must return more than one result a different method must be used. We will get to one of those methods shortly when we discuss pointers. A simple possibility that might be appealing on first thought is to use global variables, ones accessible by both the calling program and the function, and simply have the function assign it's multiple results to multiple global variables (pricisely how BASIC does it). A bit later we are going to talk about the "scope of variables". At that time we will discuss why this is a messy solution that can lead to serious "side effects" that can make a program hard to debug.

Assignment Statements Revisited

We discussed assignment statements earlier. We didn't mention a couple of things about them. First, you can stack assignment statements, for example to initialize a bunch of variables to zero.

```
a=b=d=y=q=0;
```

The assignment works from right to left so it puts 0 in q and then the value of q in y etc.

The second thing about an assignment statement is that the statement itself takes on the value of the assignment. For example the statement (a=3) has the value 3. That is why so many new C programmers get tripped up by the "==" comparison:

if (a=3) is guaranteed to be TRUE ALWAYS, since it is assigning the value of 3 to a. Since 3 is not equal to zero, the statement a=3 is always TRUE.

if(a==3) on the other hand is a valid comparison test.

Since the first construct if(a=3) is syntactically correct, it is not an error, but generally it is not what you wanted to do. Turbo C issues a warning that says "Possibly incorrect assignment statement" or something close to that.

Is there a place where you might want to use this value of a statement?

```
while((ch = getc(infile)) !=EOF) putc(ch,outfile);
```

The statement gets a character from the input file and tests the value of the character (i.e. the value of the assignment statement) to see if it is EOF. C's file handling requires that ch is an integer and it returns -1 when it reaches the end of a file. That is, a true integer 16 bit -1 so it is different from a signed character -1, hexadecimal FFFF as compared to hexadecimal FF.

The while loop has one statement. If the test passes it writes the character that it got out to the output file. All you need is to open two files and then this single line copies the contents of infile to outfile. When you exit the loop because EOF was detected you simply close the two files and quit the program.

Pointers

This is perhaps the one most difficult part of C to grasp initially. A pointer variable contains a memory address of another variable. The symbol "*" is used to indicate a pointer.

```
char *ptr;
```

This declares a variable named ptr to be a pointer to a data item of type character.

```
char message[] = {"Hi there"}

ptr = &message[0];
```

ptr now contains the address of the character "H" in the message. We talked about how an array name is a synonym for the address of the first element in the array so we could equally well say:

```
ptr = message;
```

Now we access the "H" by using:

```
putchar(*ptr);
```

The combination *ptr means "the character pointed at by ptr". The "*" is said to "dereference" the pointer. Now with this in mind let's do a function to print out our familiar character string "message" two different ways:

```
ptr_print_string(string);
     char *string;
{
     while(*string)
     putchar(*string++);
}

array_print_string(string);
     char string[];
{
     while(string[n])
     putchar(string[n++]);
}
```

The two methods are so much equivalent that we can mix and match:

```
mixed_print_string(string);
     char *string;
{
     int n=0; while(string[n])
     putchar(string[n++]);
}
```

In the calling function, remember "message" is the address of the first location in the array, exactly what we want in mixed_print_string. char *string is exactly equivalent to char string[]. In the first example of the three, the expression *string++ increments the pointer value, not the value pointed at. Further a pointer to an integer or a long will be appropriately incremented by 2 or 4 because integers are two bytes long and longs are 4 bytes. C takes care of this automatically. You can increment the thing pointed at by a pointer by using parentheses as:

```
(*string)++;
```

One more thing will show the equivalence a bit better:

```
char message[] = {"Hi there")};
char* ptr;

ptr = message;
// or ptr = &message[0];
```

Now, we can use message[3] to point at the space in message or we can use *(ptr+3). The two are exactly equivalent*. C gives the name "message" a constant value that is the address of message[0]. In other words it points at the first element in the array.

* Well, not EXACTLY equivalent. The name "message" is a constant pointer. That is it will always contain the address of the message array's first byte. We can assign this value to a pointer, and the pointer can then be incremented (but "message" cannot). That is, the pointer's value is not constant, but the value of "message" IS a constant.

We can initialize a pointer with an address or with a literal constant as well, using an initializer when we declare it just as we can say int x=3;. Of course if we use an initializer like the following, we must have defined (but not necessarily initialized) a character array named message[].

```
char *ptr = message;

                or

char *ptr = {"Hi There"};
```

Now we can print the string with: puts(ptr);

Of course you don't have to call a pointer "ptr" but it might help someone reading the program if you include the letters ptr or at least append a "p" (or maybe a "ptr") to the end. For example a pointer to a message could be called messagep or messageptr or mesptr.

There is one other way in which pointers and arrays are not equivalent. Several issues ago I was chastised for my remarks that a textbook that I have indicates that array references in a program are converted by the compiler to generate the same code as when pointers are used. One reader disassembled some output code of a compiler and found that using pointers generates less code than using arrays. I did the same for my Turbo C output code for some simple cases and I did find that code that used pointers generated smaller object code than array references. The difference was small, however. With an older compiler and an "antique" computer you may well want to do whatever is necessary to save memory and speed up the program execution. With Turbo C on a 486 system with 8 megs of RAM and a 500 Mbyte hard drive, who cares? (I'm sure someone is going to tell me that HE cares!, but in general, I don't).

This ought to leave you with a few things to think about until next time! Maybe you've gotten your curiosity arroused enough to get hooked and go to buy a textbook on C in a bookstore. If so, this series is accomplishing its purpose. I've been trying to adapt my original lessons that were based on ANSI C, and I am afraid the translation is not as clean as a start-from-scratch series would have been. Part of the problem is that my mind set has been "converted" and I've (purposely) forgotten the "old way". I remember fussing and fuming about some of my older C programs having to have a bunch of things changed before they would successfully compile under an ANSI C compiler version, but now that I am converted, I really like the new way much better.

Some Thoughts

These are current notes added to the end of my old class note. I realize that we haven't really covered files in C just yet, but this is not terribly profound, and it is of current and immediate relevence to me. Something I've been having trouble with (for a very long time) is a simple matter of detecting an end of file. A loop that starts:

```
while ((ch = getc(infile)) !=EOF)
```

worksjust fine since it tests every character that it gets for the end of file condition. I tend to make this my outer loop and then process words or lines or paragraphs or whatever inside the loop. I very frequently forget that I have to test for end of file every time I get a character (except for some unusual file structures that signal the last item in a list). The result is that my program goes into an infinite loop and I spend a while figuring out why and correcting the problem. A case in point is one that I did last night to use for this series. I wrote the C class notes using WordStar for Windows on a PC. I wanted to translate them to straight ASCII files. WS lets me "export" them in several modes, one being "stripped ASCII". Stripped means that all non printable characters and such things as font descriptions are removed.

Most text processors treat a paragraph as one long line. When you type a paragraph in, you don't have to type a CR at the end of the line. The word processor wraps the text around to the next line, but it doesn't insert a CR. The result is that a paragraph is really just one long line with a CR at the end. My

ASCII editor needs lines with CR's and it doesn't automatically insert them. I therefore wrote a program in C to copy any line shorter than a number of positions I set up as a #define constant LINLEN in the program without change. If a line (paragraph) exceeds this length, the program backs up to the preceeding space and outputs that much as a line, adding a cr or a \n in the terms of C. It then starts at the character after that space and counts characters etc. When it gets to the end of the paragraph (as defined by text ending with \n) it reads another paragraph and starts over again.

I found a shareware program called RAP (as in wrap) the other night, that makes each line X characters long. You specify X in the command line. The problem is that if there are short lines (as in a program listing) it combines them to make a longer line. I wanted to split long paragraph-lines but not touch program listings etc. so I thought I would do it for myself.

I finished my program and it went into orbit. I found my problem with end of file and it began to work. After a few more bugs were removed it worked fine and I used it to modify the text file for the above lesson. Since this is a short program (about two pages of source) that is complex enough to be a little more challenging than "printf("Hi There\n");" I am going to present it later as an example, with a discussion of how it works.

Lest you all think that I have "arrived" simply because I can write some simple tutorials on basic "C" programming, let me tell you that I found a program on my hard drive the other day that I vaguely remembered doing for a project that never materialized. One part of it was a program that reads a text file and a data file containing search - replace pairs, and generates an output file in which each search word is deleted and the corresponding replace word is inserted. Apparently I had dumped this program into my computer and never started debugging it. Now that I think about it a while, I suspect it was written in "Whimsical" for a 68000 machine and translated to C but never tested at all.

Well, the thing didn't work. First I had trouble with allocating a couple of large buffers in which to shuttle the text file back and forth once for each search replace pair. Then I found that I needed to use a larger memory model in the compiler. All in all, I think I spent about 16 hours on the project before I finally got it all working. Part of the problem was the End of File problem discussed above. The rest of it finally boiled down to my not having initialized an array index at the start of each pass through a nested loop. Along the way, the program got so bad it would hang the computer and I would have to reset it and try again.

As I look back on the "good old days", when I spent that much time working on a program, I ended up with something that took 15 minutes to compile and at least a few minutes to run. This program for a fairly large text file and a list of 15 or so substitutions, seems to run in less time than it takes to type the command line. In fact, it is done about the time I get my finger off of the ENTER key! What must I do to write a program that takes more than a few seconds to execute?

Assembler

I'm going to "chicken out" a little this month and discuss an assembler program that is just a slight variation of COPY that we have done previously. The difference is significant, however. I want to introduce an example of a "filter" program. COPY, as you probably remember if you have been following this series, reads a file named on the command line and writes a copy to another file named on the command line.

As I mentioned above, a filter program does a little more than copy a file. It makes some significant modification to the data in the file as it reads one file and writes another. As a fairly simple example I present a program called UPLOW that reads an input ASCII text file and converts all uppercase letters to lower case, otherwise passing the text straight through. To do that we simply add a few lines between the reading of characters and the writing of characters. To keep things simple we are assuming an ASCII text file.

```
* A FILTER PROGRAM TO CONVERT UPPER TO LOWER CASE

          NAM       UPLOW
PUTCHR    EQU       $CD18
WARMS     EQU       $CD03
RPTERR    EQU       $CD3F
FMS       EQU       $D406
FMSCLS    EQU       $D403
NXTCH     EQU       $CD27
SETEXT    EQU       $CD33
WRKDRV    EQU       $CC0C
PCRLF     EQU       $CD24
GETFIL    EQU       $CD2D

CR        EQU       $0D
FOPENR    EQU       $01
FOPENW    EQU       $02
FCLOSE    EQU       $04

START     LDX       #IFCB
          JSR       GETFIL
          LDA       #1
          JSR       SETEXT    SETS EXTN TO .TXT IF NONE SPECIFIED
          LDA       #FOPENR   OPEN FOR READ CODE
          STA       0,X
          JSR       FMS
          BNE       ERROR FMS SETS NON ZERO ON ERROR

* NOW DO IT AGAIN FOR THE OUTPUT FILE
          LDX       #OFCB
          JSR       GETFIL
          LDA       #1
          JSR       SETEXT
          LDA       #FOPENW   OPEN FOR WRITE CODE
          STA       0,X
          JSR       FMS
          BNE       ERROR

* LEAVE FILES AS TEXT RATHER THAN BINARY
LOOP      LDX       #IFCB
          JSR       FMS READ  A CHARACTER TO ACCA
          BNE       ERROR

* NOW LET'S BUILD A "FILTER"

          CMPA      #'A
          BLT       CONTIN    IF LESS THAN CHAR A LEAVE ALONE
          CMPA      #'Z
          BGT       CONTIN    IF GREATER THAN CHAR Z LEAVE ALONE
          ADDA      #$20      ELSE MAKE IT LOWER CASE

* END OF OUR SIMPLE FILTER TO CONVERT UPPER CASE ASCII TO LOWER

CONTIN    LDX       #OFCB
          JSR       FMS       WRITE A CHARACTER
          BNE       ERROR
          BRA       LOOP      GO AROUND AGAIN
```

```
ERROR   LDB     1,X
        CMPB    #$08        TEST FOR END OF FILE
        BEQ     DONE
        JSR     RPTERR      TELL USER WHICH ERROR - X POINTING AT FCB
        JSR     FMSCLS      CLEAN UP BY CLOSING ALL OPEN FILES ON ERROR
        JMP     WARMS

DONE    LDX     #IFCB
        LDA     #$04
        STA     0,X         CLOSE THE FILE
        JSR     FMS

        LDX     #OFCB
        LDA     #$04
        STA     0,X         CLOSE THE FILE
        JSR     FMS

        JMP     WARMS

        ORG     $200
IFCB    FCB     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        RMB     305
OFCB    FCB     0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
        RMB     305

        END     START
```

I don't remember if we have discussed the TSC Assembler's convention for an ASCII character code. It uses an apostrophe before a character to stand for the ASCII code for that character. Thus 'A, $41 and 65 are all equivalent. Since we are manipulating characters here, it makes the program more readable to use the ASCII character representation.

The ASCII code for the letter "A" is $41. That for the letter "a" is $61. Since all upper and lower case letters are in alphabetical order, the lower case letter's code is $20 larger than that for the same letter in upper case. ($20 is decimal 32).

The program therefore checks to see whether each letter is in the range of the codes for the capital letters. If the code is less than that for A or greater than that for Z, it is simply passed through. Otherwise it has $20 added to it before being written.

Here is a test file for input to this program:

```
01234567890@{}[]
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

This is the result of a run of the program:

```
01234567890@{}[]
abcdefghijklmnopqrstuvwxyz
abcdefghijklmnopqrstuvwxyz
```

A word is in order concerning testing. When you do something like this it is easy to get the test "off by one". It is a good idea to check the codes immediately before "A" and immediately after "Z". These are "@" ($40), and "[", ($5B) respectively. You will note that they both went through the filter unchanged, and that all the upper case letters were nicely replaced by their lower case versions. The test shows that codes $41 through $5A are modified but codes $40 and $5B are not. The program therefore works as we expect.

Though the example code is simple, the idea can get as complex as you like. For example a TAB character ($09) has a different meaning in a FLEX text file than it has in an MS-DOS text file. As we have mentioned previously, in a FLEX text file a $09 means that the next byte is going to contain a space count. $09 $08 would mean to insert 8 spaces. FLEX does this transparently. It is built right into the operating system. A string of spaces is converted into an $09 and a count when a file is written, and back to a number of spaces when it is read. If it is opened as a binary file this is not the case. You could write a filter to read a FLEX file and convert all tab sequences to a number of spaces, which would be dumb because FLEX does it automatically when you read the file as a text file. Better you could translate an MS-DOS file into a FLEX file by reading it as a binary file and inserting, say, three spaces whenever you encounter a $09. Then you could write it in TEXT mode so the spaces would be compressed to a $09 and count sequence.

Above I referred to the problem of breaking a long line (a paragraph from a text formatter) into shorter lines by adding CR and LF at appropriate word breaks. I described doing such a program in C, but you could easily do the same in Assembler. Our COPY program is a nice template for doing all sorts of filter programs. All you have to do is to add the filter action between the read and the write. Obviously you would have to read and store characters until you had enough to make a line before writing a line to the output file.

You could write a program to recognize words and substitute other words in a text file. You would have to buffer whole words and compare them with a search word, then replace that with a replace word. It would be just like the search and replace function of many editors. You could write a program that would accept input filename, output filename, search word, and replace word as parameters on the command line. Such a utility might be useful, but if you decide to try one watch out for some traps. If you want to change "the" to "a" you have to include leading and trailing spaces or you will end up changing "theodore" to "aodore". Also you will have to handle capitalization in order to make a useful utility for a text file. Change "The" to "A" and change "the" to "a" in other words.

If anyone out there is interested in moving files between a PC and a 6809 running FLEX, I have written two pair of utilities, one for each machine that allow copying a disk of text files in either direction. My FLEX to MS-DOS transfer program is a bit specialized to handle Extended BASIC integer and floating point numbers as well since it is set up for reading virtual arrays in Extended BASIC and converting them to ASCII number arrays on the DOS end. I can remove some code and clean them up to handle TEXT files only if anyone would like copies. There are some cautions, but I would be glad to share these with anyone wanting them. They work over a simple serial cable with no hardware handshakes, so it is not very hard to make the proper cable for them. The PC programs are written in Turbo C and the 6809 end is written for John Spray's 6809 Whimsical compiler.

Floppy Disk Problems

By Claude Palm

Special Feature

Beginning Forth

Trouble Shooting Design

I received this article a rather long time ago, and had asked Claude to send me the disk. Well somewhere in time, the messages got lost and I finally broke down and scanned it in. It is important for several reasons that you should read it. Mainly it talks about the floppy chip used in PC clone machines, but more importantly it explains about finding problems. Maintaining your own system is mostly about learning how to hunt down those elusive problems that others would simply use as an excuse to "throw the #$%^& machine away." Fortunately Claude didn't give up, even better he documented his actions for you to see. Thanks. Claude! Bill Kibler.*

CASE OF THE CONFUSED FDC.

This is a detective story. The clues lead in one direction, then another, and the solution found in yet another. Is it software, hardware, buggy chips, or a bit of everything? I got into electronics in the 60's, computers in the 70's but this is the most confusing bug I ever came across. It may interest your readers, and some may even learn something from it.

It all happened while designing the CPUZ180 SBC and concerned interfacing a National DP8473 FDC (Floppy Disk Controller) to a Z180 CPU. Before I come to the problem I have to give a reasonably detailed description of what was involved, as it is rather technical and caused by some of the finer points of the behavior of both chips.

THE CHIPS

The DP8473 was designed to interface with the IBM PC bus, thus all its interface logic and I/O ports are IBM compatible. The FDC is accessed via its STATUS and DATA ports (plus a few others). The STATUS port holds 8 bits and indicates things like BUSY, DATA REQUEST, and whether the request is to receive a byte or if it wants to send a byte. The byte itself is transferred through the DATA port. To select the chip there is a CS (Chip Select) pin, then 3 address lines (AO-2) to specify the required port. One of the RD or WR pins must then be asserted before a byte is transferred.

To execute a command you send a specified number of bytes to the DATA port, wait for the command to execute which may involve sending/receiving sector data. On completion the FDC issues an interrupt, after which you read the result, again from the DATA port. For example, to read a sector you write 9 bytes of command and sector information, and afterwards read 7 bytes of result. The STATUS port informs you of the progress, the BUSY bit will be set after the first byte of the command, and the sector read will commence after the 9th byte is written. These operations are referred to as the 'command phase', 'execution phase', and 'result phase'.

During the execution phase, sector data can be transferred in several ways. The polled method lets you read it byte for byte via the DATA port as it comes to hand, then you put it somewhere. The STATUS port will advise when, but if you miss a byte then it it's lost. When the entire sector is transferred the FDC will issue an interrupt and the result phase commences. After the last result byte is read, the FDC clears its BUSY bit so you don't have to count the bytes.

Polling works ok, but holds up the CPU for long periods, as it may take some time for it to find the particular sector. Interrupts must be disabled so an interrupt driven keyboard could easily loose a character. If left enabled you probably loose some data instead. There is also an interrupt driven method, but I won't go into that.

The DMA (Direct Memory Access) method is by far the best way to transfer sector data, but it requires extra hardware. You must have a DMA controller (the Z180 has two of them built in) to do the actual transfer. While it is shuffling data the CPU can go about its normal business, handling interrupts and so on. The FDC sends out a DRQ (Data ReQuest) signal to the DMA controller. The DMA replies with a DACK (Drq ACKnowledge) then transfers a byte between the FDC and memory. This takes only a fraction of a microsecond per byte and the CPU is disabled for that time only. Before starting DMA the controller must of course be instructed with what to do, memory locations and so on, but after that the operation is completely transparent to the CPU.

As the FDC interrupts the CPU when transfer is complete, the interrupt routine can read out the result and store it somewhere, then return to the main program. The FDC also interrupts on completion of other commands, such as a RESTORE or after it has been reset. These interrupts are distinguished by the BUSY bit being clear, when you must issue an SI (Sense Interrupt) command to the FDC, and read the result to see what the interrupt was about. In case there were several

conditions causing interrupt you should repeat the SI command until the result byte is 80h, which is the same result as an illegal command i.e. the first (and only) result byte is 80h if the FDC didn't understand the command. The rest of the story hinges on that.

THE HARDWARE

Now for some hardware description. The Z180 DMA reacts to a DRQ signal by transferring a single byte between the FDC and memory, but it won't issue a DACK. Instead it performs a normal Z180 I/O cycle by asserting its IORQ signal (to advise the outside world that this is a data transfer to/from an I/O port) together with the specified I/O address and a RD or WR strobe to denote the direction.

With the FDC in DMA mode, data can only be transferred after receipt of the DACK and RD/WR signals, regardless of the CS (Chip Select) and address lines. The FDC only enters the DMA mode during actual sector data transfers, it is always in the polled mode during the command or result phases. Then it uses the CS, RD/WR, and address inputs to work out what action to take. The RD/WR signals come from the CPU but are ANDed with the IORQ line so they won't appear during memory accesses.

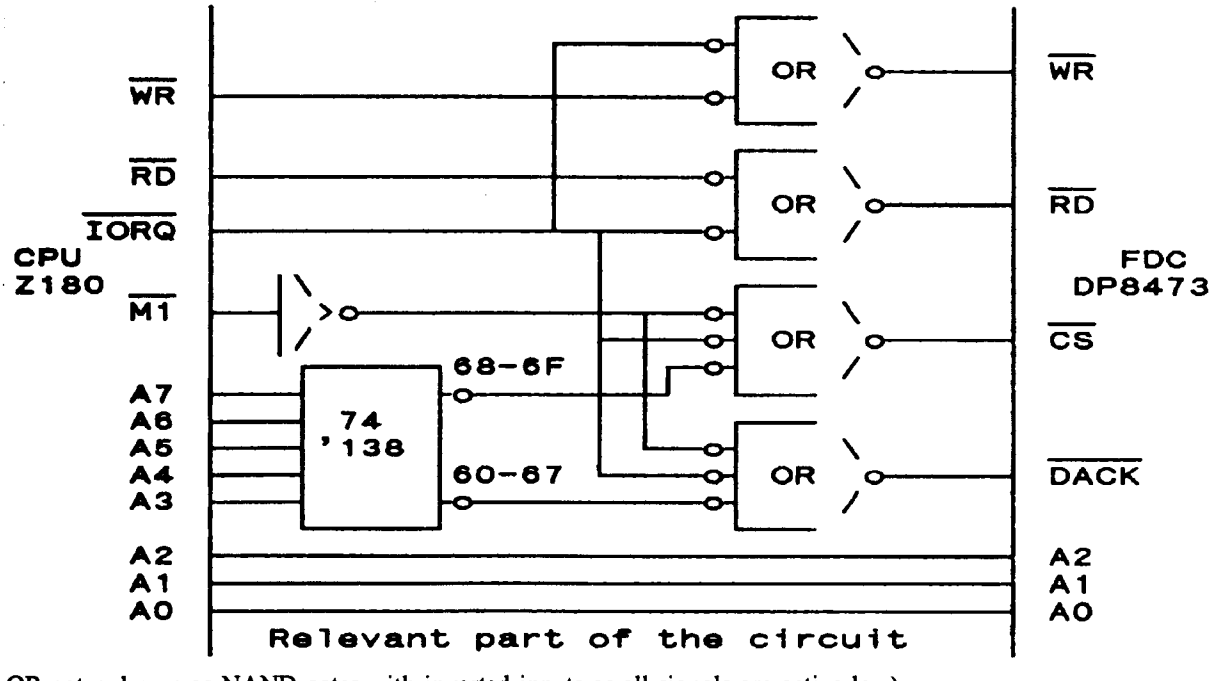
The CPUZ180 uses a simple address decoder (74ACT138) for all its I/O port selects. As it only decodes-5 address lines, it will also send out select signals when a matching address is during normal program execution. The data sheet for the FDC states that a CS may not occur during DMA, and further that any DACK received will effectively clear a DRQ, regardless of any RD or WR signal. Another point is that a Z180 INTACK (Interrupt ACKnowledge) cycle is similar to its I/O cycle, it asserts IORQ but also an M1 signal which is not used, during I/O.

Bearing all that in mind I conditioned the FDC select signal from the decoder by ANDing it with IORQ and disabling it during M1. That way there wouldn't be any stray FDC CS pulses. I then devised a DACK signal by picking an unused port address from the decoder and conditioning it the same way. As the DMA would be the only user of that port, there wouldn't be any stray DACKs either, or so I thought.

NO PROBLEMS

Lets put it all together now. I had the same FDC working with a HD64180 using polled I/O so I used that software to test the FDC in the CPUZ180. After a lot of changes I ended up with a working floppy disk system. Next step was to change from polled to DMA data transfer because as expected, the keyboard did loose characters when you typed during floppy access. Secondly, polled I/O is not a very neat way of doing things and I already had the hardware setup.

The whole exercise ended up being relatively straight forward, as far as these things go anyway. An interrupt routine was written so that it would read out the FDC's result to a buffer, handling other FDC interrupts and so on. The sector I/O routine would put the command in another buffer, then write it out to the FDC which would then start any required seeks, find the sector etc. While it was doing that, the DMA controller was loaded with addresses, byte count, and also set to work waiting for the first DRQ from the FDC. Then all that was left was to sort out some housekeeping and settle down to wait until



(note: OR gates drawn as NAND gates with inverted inputs as all signals are active low)

the FDC interrupted things and the result was read into its buffer. The interrupt routine would set a flag to signal completion. Finally the result buffer would be checked to see if any error had occurred.

After sorting out the usual bugs that crop up in this sort of work, everything was working satisfactorily. The routines had been working out of RAM during the trials, so I moved them into

FLASH (semi-permanent like EEPROM) memory with the rest of the BIOS code to keep the RAM as free as possible. Final testing of the new FLASH based BIOS showed no problems concerning the floppy access, so I went on to other things, and that would have been the end of an uneventful story.

PROBLEMS EMERGE

However, the 'other things' involved several additions and modification to the FLASH resident BIOS code, not directly involving the FDC routines. I had been using a serial communication line to transfer programs between the CPUZ180 prototype and my main computer, but one day I needed a program that I had on a floppy disk nearby. The easiest way was to insert the disk and copy the file to the hard drive, which I did. Only I got an 'Undefined Error' with a query if I wanted to reboot, retry or skip the sector. Retry or reboot didn't help, so I shut it down, cold started, and tried again. Same result, I tried another disk, still no go. I tried reading the disks in the main computer and they were ok. I tried another disk drive but that didn't help either. I could not see why because it had worked flawlessly before.

Well, it wasn't the obvious, so what was it, software, hardware, or a bad chip. I don't believe in bad chips, they almost never go bad if looked after properly. Maybe I inadvertently changed something in the FLASH memory during my other modifications? I checked against my listings, but apart from residing in a different location, there were no discrepancies. I reassembled the code and did a byte for byte compare in case of a bad memory cell. Same code. All right, try the chips involved, but that made no difference either.

Maybe the processor speed. You can insert up to 3 extra wait states in the Z180, each wait holding the processor (and DMA controller) up for another clock cycle. It had been running at 9 MHz with no waits, so I set it up for the slowest possible performance with an extra 300 nS per machine cycle. That ought to fix a slow memory cell or some marginal FDC interface. It didn't.

Could it be in the board itself? It was a double sided board produced in house. It used copper eyelets for feed-throughs (350 of them) in lieu of plating, and everything hand soldered. Definitely a suspect. But a thorough checkout with an oscilloscope only showed that all signals were present and correct and still no go. Some old fashioned prodding also failed.

UP THE GARDEN PATH

Into the debug monitor and there I got a surprise. There was definitely floppy disk data in the main sector buffer. It must be reading something. The command buffer showed the correct command and sector address, but the result buffer had 80h in the first location. If you remember the previous discussion, 80h stands for either an illegal command or some other interrupt caused the interrupt routine to issue SI commands until the result was 80h. As everything was legal in the command buffer, some other interrupt must have caused it. As 80h is not a valid result after a read command, BIOS reported it as undefined.

But where did the extra interrupt come from? I went through the code several time, but could find nothing. The debug monitor is unable to set breakpoints in FLASH memory as it can't be written to readily (only in 16KB blocks), but is capable to single step through it by moving each instruction into RAM. Very tedious as I had to single step through a large number of unimportant instructions before arriving at the business end of the code. Then I had to start the DMA manually because it is only milliseconds before the FDC starts sending data and I was stepping through each instruction via the keyboard.

The end was a complete anticlimax. The sector was transferred with no error at all, so I had to try another tack. The BIOS will retry an undefined error once, by resetting the FDC, reloading all parameters and then executing the command again. This could easily be altered to up to four resets, which I did. That actually worked, with the occasional error reported and, by
replying 'retry' to BIOS's query, I could load an entire file. Next step was to log the errors. That didn't shed any more light on the problem. A particular sector might read at first go in one load, but needed half dozen retries the next time. Completely random, but only about 1 sector in 5 got through on the first try.

FURTHER UP THE GARDEN PATH

I started thinking about a timing problem. Maybe something was extremely marginal. One way to upset timings is to either heat the board or freeze it. I tried the fridge and then a heat shrink blower, but neither made any difference. The errors occurred at about the same rate, cold or hot. That seemed to discount the timing theory.

Then I thought of the original test program, the one that resided in RAM. At least I could set breakpoints, and move around in it a lot easier, as I seemed to be in for the long haul. I dug it out, loaded and ran it, and would you believe, perfect. I even zeroed out all retries, and still no errors, yet it was the same code. Must be in the FLASH memory, but it had previously compared ok when tested, tried at a slow speed without

improvement. Still, I set up a routine to constantly compare the FLASH with its supposed contents that I had loaded into RAM. I let it run for awhile and it could not find any mismatches. Yet the disk I/O routine would execute properly out of RAM but not from the FLASH memory.

I concentrated on the RAM based disk program. I had it working without errors, so I started to modify it to use some of the routines in the FLASH memory. It kept on working perfectly until I added some code so that it would use the FLASH based interrupt routine. Then the errors started up again. Only occasionally mind you, about 1 error in 4 reads, instead of the other way around as before, but with exactly the same result. That was the queer part: running partly out of RAM only changed the frequency of the same error. Running entirely out of RAM produced no error. I rechecked FLASH error rate but it was unchanged. It just didn't make sense.

Now, with much of the code in RAM, I set a few breakpoints through the debugger. One just before the command was issued another one at the interrupt entry point. Then ran it. The first few times there were no errors, but then, after I let the read command start executing, the interrupt routine caught on another breakpoint and I single stepped through the result readout as I had done 3 or 4 time before. This time however the very first byte read out from the DATA register, supposedly the first byte in the result, was the by now dreaded 80h. Yet after checking, the DMA controller had finished and the sector had been transferred to the sector buffer with no apparent error. Before commencing each read I had cleared the buffer to all 0's, and now it contained 512 bytes of directory data. At least I had watched the bug in action. I should have continued to single step through the entire result phase as the STATUS register would have indicated that there was more to come, but I didn't (hindsight). I was instead trying to work out how the FDC could report an illegal command after it had just transferred a sector correctly. No way that it could do that.

I changed the code to accept the result of 80h as a no-error result and ran it. After several file transfers and compares, I was satisfied that the software and the disk system was working correctly. The FDC was just giving out the wrong result. Problem was: how to trap a real error if one occurred?

When doing single sector reads, I had used track 00 sector 0, simply because the command buffer was automatically cleared to these values. Now I was going to try writing to the disk, and as it had a valid directory there, I started using some empty tracks between 70 and 79. The undefined error condition occurred in their usual frequency during writes but the writes themselves were successful. I was examining the result buffer after another 8Oh found its way to it. The buffer should actually hold 3 bytes of status information (bit records), final track, head, sector numbers, and the sector size. I noted that, apart from the familiar 80h in the first location, it also had a track number in it. It could have got there from a previous non-error read, but it was in the wrong place, the 3rd instead of 4th byte. Before, when I was using 0,0,0 as sector address I wouldn't have noticed it.

GETTING WARM

I went back to the FLASH based system where I was almost certain to end up with an error, while relinquishing my ability to set breakpoints. I then cleared the result buffer and did a read on one of the outer tracks. To top it off I had put in a non-zero sector number, on head #1. Sure enough, BIOS reported an undefined error, and straight in with the debugger rather than replying. First check the result buffer. It started with 80h all right, but it also contained the track, sector, head, and size. Only they were in the 3rd to 6th byte NOT in the 4th to 7th location where they ought to be. Although they didn't ought to be there at all. The FDC is only supposed to send a single byte result on an illegal command, and in any case the routine that checks the FDC's SI results only reads up to two result bytes.

Then the penny dropped: 80h is a valid 2nd status byte, in fact it is the expected 2nd status byte, indicating that the last sector in the command has been processed (bit 7 set) The other bits represent certain errors when set. It just hadn't entered my mind before, I was so sure that the 80h was the illegal command result. The whole buffer was shifted one byte down. The first result byte which ought to be 40h was nowhere to be found.

Now I knew what was happening, but not why. Something was reading a byte out of the FDC before the result routine got to it. After finishing the housekeeping, the program was simply sitting in a tight loop waiting for the 'interrupt complete' flag to be set, and also checking for time-out in case the interrupt never occurred. As soon as the last data byte is transferred the FDC should generate the interrupt, indicating the end of the execution phase, then enter into the polled result phase. That would invoke the routine responsible to read the result. Not much in between, yet something got that first byte. While single stepping the result routine I had seen that the 80h really was the first byte read (if I had continued that time, I would also have seen the track, sector, and other information come forth, but that's life)

Then I had a glimmer of an idea. The address decoder only uses the lower bits of an address (actually bits 3-7). It assumes anything between 60 and 67h is from the DMA controller and sends it on to become the DACK signal. Similarly 68 to 6Fh goes on to become the FDC's CS signal, as long as they occur in conjunction with the IORQ signal while Ml is off. The sector data I/O uses 60h, sent by the DMA controller, the main STATUS port is at 6Ch while 6Dh is the DATA port. The loop waiting for the interrupt to occur in FLASH memory spent considerable time around addresses ending in those ranges. Coincidence? I reloaded the last RAM based program I had

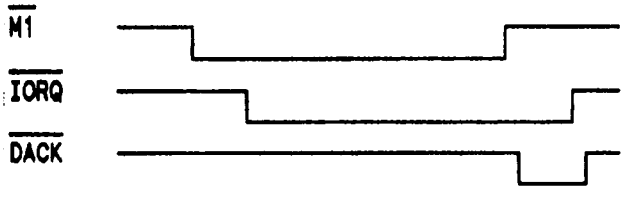
used, the one that produced the odd error, and that loop had used two addresses with the low 8 bits in the same range!

CAUGHT IT

I put in a few NOPS to shift the loop out of that range, reassembled, and ran it. No errors. I did it again, just to prove the point, this time shifting the loop so it executed right in the range. The 8 low bits in every address were sixty something hex. Plenty of errors this time. I had found it, but I still couldn't figure out why.

I wrote another tight loop around those addresses, then an interrupt routine that did nothing. I shut the machine down. As the RAM is battery backed it would hold the program. Next I pulled the FDC chip from its socket and hooked a pulse generator onto the interrupt pin so that the CPU would receive the pulses on its INTO input without fighting another output. Then I powered up in the debug monitor and set the whole thing running with the oscilloscope nearby.

The 'scope showed the expected pulses from the decoder as the loop executed, but nothing on the conditioned DACK or FDC select lines. So far so good. Then I applied the interrupt pulses from the generator, and there they were. Rather short pulses on both the DACK and FDC CS lines. That did explain the problem, but also posed new questions such as: why were they there, and why did the FDC react to them to the extent of loosing a byte. The first question was answered by the Z180 manual on interrupts: the Ml signal appears and ends half a clock before the IORQ signal during an INTACK cycle. Also, the address present on the bus at that time points to the instruction following the interrupted instruction. The conclusion was that for the last half clock cycle during INTACK we have an address - in this case the low bits are 6x, which the decoder recognizes. We also have IORQ without Ml which my system control PLD recognizes as a valid DACK or FDC CS and thus sends out either signal, albeit for a brief period.



The INTACK cycle as shown on the 'scope.

I shut down, removed all gadgets, put the FDC chip back in place and fired up again. Then I went back to the RAM based disk I/O routines and shifted the loop as before, this time so ,that it ended with the 8 low bits at 66h i.e the extra pulses would be recognized as DACK signals only, as the FDC select starts at 68h. Running that produced the errors as before. Then I shifted it again, this time starting the loop with the low address bits at 68h. No DACKs but should get some extra FDC CS pulses. Ran it again, this time with no errors.

The exercise proved that an extra DACK pulse, even a very narrow one, will remove a status byte, but there is nothing about that in the data sheet. It also proved that extra CS pulses makes no difference, which is something the data sheet warns you against. The DP8473 data sheet actually specifies that 'During DMA operations the Chip Select input must be held high' end quote. I think they got that back to front. It ought to read: 'During polled I/O the DACK input must be held high' It just shows that you cannot,believe everything you read.

THE FIX

I had to do something a bit more permanent. Just by putting the loop in another area would not be a good-idea. A timer or keyboard interrupt could occur, then the FDC interrupting while these being attended to. That could be at any address, quite possibly generating a spurious DACK signal to cause an FDC error.

The solution was not hard to come by, all I had to do was to qualify the DACK signal with the CPU's RD or WR strobe - either had to occur during DMA but not during INTACK - as well as the IORQ signal. The Ml signal could be dispensed with. It required some cut traces, a couple of wire jumpers and a redesigned PLD. All up it took about an hour to fix what had taken a week to find. It proved to be a permanent solution too, as the floppy disk part of the system has behaved flawlessly ever since.

AS AN ASIDE

A month or so later I had installed an 18Mhz CPU and running at full speed (with a fair few changes to FLASH memory contents). I wanted a printout of something, and the board was running without a printer as it does for most of the time, so I plugged one in. Would it print anything? No way. It didn't even hang or anything, just a short pause then continued as if the printing was completed. A sense of deja vu if I ever had one. First thing was to try manipulating the printer ports from the monitor, and it did print that way. A quick check through the printer driver listing revealed a likely culprit. With the CPU operating at 18 MHz an instruction does not take very long to execute and the entire operation of loading the data then pulsing the printer strobe would have taken well under the specified 1 uS. Probably not long enough for the printer to recognize the strobe. Insert an extra instruction to waste some time and that was that. Bug found, tried and executed in a quarter hour. I wish all bugs were like that.

Regular Feature

Contact Listing

SUPPORT GROUPS FOR THE CLASSICS

TCJ Staff Contacts

TCJ Editor: Dave Baldwin, Voice (916)722-4970, FAX (916)722-7480 or TCJ BBS (916) 722-5799 (use "computer", "journal", pswd "subscriber" as log on), Internet dibald@netcom.com, CompuServe 70403,2444, tcj@psyber.com.

Bill D. Kibler, PO Box 535, Lincoln, CA 95648, (916)645-1670, GEnie: B.Kibler, CompuServe: 71563,2243, E-mail: kibler@psyber.com.

Z-System Support: Jay Sage,1435 Centre St. Newton Centre, MA 02159-2469, (617)965-3552, BBS: (617)965-7259; E-mail: Sage@ll.mit.edu. Also sells Z-System software.

32Bit Support: Rick Rodman, 1150 Kettle Pond Lane, Great Falls, VA 22066. BBS/FAX (703)769-1169.

Kaypro Support: Charles Stafford, 4000 Norris Ave., Sacramento, CA 95821, (916)483-0312 (eves). Also sells Kaypro upgrades, see ad inside back cover. CompuServe 73664,2470 (73664.2470@cis).

S-100 Support: Herb Johnson, CN 5256 #105, Princeton, NJ 08543, (609)771-1503. Also sells used S-100 boards and systems, see inside back cover. E-mail: hjohnson@pluto.njcc.com.

6800/6809 Support: Ronald Anderson, 3540 Sturbridge Ct., Ann Arbor, MI 48105.

Regular Contributors:

Brad Rodriguez,Box 77, McMaster Univ., 1280 Main St. West, Hamilton, ONT, L8S 1C0, Canada, E-mail: bj@headwaters.com..

Frank Sergeant, 809 W. San Antonio St., San Marcos, TX 78666, E-mail: pygmy@pobox.com.

Tilmann Reh, Germany, E-mail: tilmann.reh@hrz.uni-siegen.d400.de. Has many programs for CP/M+ and is active with Z180/280 ECB bus/Modular/Embedded computers. Microcontrollers (8051).

Helmut Jungkunz, Munich, Germany, ZNODE #51, 8N1, 300-14.4, +49.89.961 45 75, or CompuServe 100024,1545.

USER GROUPS

Connecticut CP/M Users Group, contact Stephen Griswold, PO Box 74, Canton CT 06019-0074, BBS: (203)665-1100. Sponsors Z-fests.

Sacramento Microcomputer Users Group, PO Box 161513, Sacramento, CA 95816-1513, BBS: (916)372-3646. Publishes newsletter, $15.00 membership, meetings at SMUD 6201 S st., Sacramento CA.

CAPDUG: The Capital Area Public Domain Users Group, Newsletter $20, Al Siegel Associates, Inc., PO Box 34667, Betherda MD 20827. BBS (301) 292-7955.

NOVAOUG: The Northern Virginia Osborne Users Group, Newsletter $12, Robert L. Crities, 7512 Fairwood Lane, Falls Church, VA 22046. Info (703) 534-1186, BBS use CAPDUG's.

The Windsor Bulletin Board Users' Group: England, Contact Rodney Hannis, 34 Falmouth Road, Reading, RG2 8QR, or Mark Minting, 94 Undley Common, Lakenheath, Brandon, Suffolk, IP27 9BZ, Phone 0842-860469 (also sells NZCOM/Z3PLUS).

L.I.S.T.: Long Island Sinclair and Timex support group, contact Harvey Rait, 5 Peri Lane, Valley Stream, NY 11581.

ADAM-Link User's Group, Salt Lake City, Utah, BBS: (801)484-5114. Supporting Coleco ADAM machines, with Newsletter / BBS.

Adam International Media, Adam's House, Route 2, Box 2756, 1829-1 County Rd. 130, Pearland TX 77581-9503, (713)482-5040. Contact Terry R. Fowler for information.

AUGER, Emerald Coast ADAM Users Group, PO Box 4934, Fort Walton Beach FL 32549-4934, (904)244-1516. Contact Norman J. Deere, treasurer and editor for pricing and newsletter information.

MOAUG, Metro Orlando Adam Users Group, Contact James Poulin, 1146 Manatee Dr. Rockledge FL 32955, (407)631-0958.

Metro Toronto Adam Group, Box 165, 260 Adelaide St. E., Toronto, ONT M5A 1N0, Canada, (416)424-1352.

Omaha ADAM Users Club, Contact Norman R. Castro, 809 W. 33rd Ave. Bellevue NE 68005, (402)291-4405. Suppose to be oldest ADAM group.

Vancouver Island Senior ADAMphiles, ADVISA newsletter by David Cobley, 17885 Berwick Rd. Qualicum Beach, B.C., Canada V9K 1N7, (604)752-1984.

Northern Illiana ADAMS User's Group, 9389 Bay Colony Dr. #3E, Des Plaines IL 60016, (708)296-0675.

San Diego OS-9 Users Group, Contact Warren Hrach (619)221-8246, BBS: (619)224-4878.

ACCESS, PO Box 1354, Sacramento, CA 95812, Contact Bob Drews (916)423-1573. Meets first Thurdays at SMUD 59Th St. (ed. bldg.).

Forth Interest Group, PO Box 2154, Oakland CA 94621 510-89-FORTH. International support of the Forth language, local chapters.

The Pacific Northwest Heath Users Group, contact Jim Moore, 1554 - 16th Avenue East, Seattle, WA 98112-2807. be483@scn.org.

The SNO-KING Kaypro User Group, contact Donald Anderson, 13227 2nd Ave South, Burien, WA 98168-2637.

SeaFOG (Seattle FOG User's Group, Formerly Osborne Users Group) PO Box 12214, Seattle, WA 98102-0214.

OTHER PUBLICATIONS

The Z-Letter, supporting Z-System and CP/M users. David A.J. McGlone, Lambda Software Publishing, 149 West Hillard Lane, Eugene, OR 97404-3057, (541)688-3563. Bi-Monthly user oriented newsletter (20 pages+). Also sells CP/M Boot disks, software.

The Analytical Engine, by the Computer History Association of California, 1001 Elm Ct. El Cerrito, CA 94530-2602. A ASCII text file distributed by Internet, issue #1 was July 1993. E-mail: kcrosby@crayola.win.net.

Z-100 LifeLine, Steven W. Vagts, 2409 Riddick Rd. Elizabeth City, NC 27909, (919)338-8302. Publication for Z-100 (a S-100 machine).

The Staunch 8/89'er, Kirk L. Thompson editor, PO Box 548, West Branch IA 52358, (319)643-7136. $15/yr(US) publication for H-8/89s.

The SEBHC Journal, Leonard Geisler, 895 Starwick Dr., Ann Arbor MI 48105, (313)662-0750. Magazine of the Society of Eight-Bit Heath computerists, H-8 and H-89 support.

Sanyo PC Hackers Newsletter, Victor R. Frank editor, 12450 Skyline Blvd. Woodside, CA 94062-4541, (415)851-7031. Support for orphaned Sanyo computers and software.

the world of 68' micros, by FARNA Systems, PO Box 321, Warner Robins, GA 31099-0321. E-mail: dsrtfox@delphi.com. New magazine for support of old CoCo's and other 68xx(x) systems.

Amstrad PCW SIG, newsletter by Al Warsh, 6889 Crest Avenue, Riverside, CA 92503-1162. $9 for 6 bi-monthly newsletters on Amstrad CP/M machines.

Historically Brewed, A publication of the Historical Computer Society. Bimonthly at $18 a year. HCS, 2962 Park Street #1, Jacksonville, FL 32205. Editor David Greelish. Computer History and more.

IQLR (International QL Report), contact Bob Dyl, 15 Kilburn Ct. Newport, RI 02840. Subscription is $20 per year.

QL Hacker's Journal (QHJ), Timothy Swenson, 5615 Botkins Rd., Huber Heights, OH 45424, (513) 233-2178, sent mail & E-mail, swensotc@ss2.sews.wpafb.af.mil. Free to programmers of QL's.

Update Magazine, PO Box 1095, Peru, IN 46970, Subs $18 per year, supports Sinclair, Timex, and Cambridge computers.

Other Support Businesses

Hal Bower writes, sells, and supports B/PBios for Ampro, SB180, and YASBEC. $69.95. Hal Bower, 7914 Redglobe Ct., Severn MD 21144-1048, (410)551-5922.

Sydex, PO Box 5700, Eugene OR 97405, (541)683-6033. Sells several CP/M programs for use with PC Clones ('22Disk' format/copies CP/M disks using PC files system).

Elliam Associates, PO Box 2664, Atascadero CA 93423, (805)466-8440. Sells CP/M user group disks and Amstrad PCW products. See ad inside back cover.

Discus Distribution Services, Inc. sells CP/M for $150, CBASIC $600, Fortran-77 $350, Pascal/MT+ $600. 8020 San Miguel Canyon Rd., Salinas CA 93907, (408)663-6966.

Microcomputer Mail-Order Library of books, manuals, and periodicals in general and H/Zenith in particular. Borrow items for small fees. Contact Lee Hart, 4209 France Ave. North, Robbinsdale MN 55422, (612)533-3226.

Star-K Software Systems Corp. PO Box 209, Mt. Kisco, NY 10549, (914)241-0287, BBS: (914)241-3307. SK*DOS 6809/68000 operating system and software. Some educational products, call for catalog.

Peripheral Technology, 1250 E. Piedmont Rd., Marietta, GA 30067, (404)973-2156. 6809/68000 single board system. 68K ISA bus compatible system. See inside front cover.

Hazelwood Computers, RR#1, Box 36, Hwy 94@Bluffton, Rhineland, MO 65069, (314)236-4372. Some SS-50 6809 boards and new 68000 systems.

AAA Chicago Computers, Jerry Koppel, (708)681-3782. SS-50 6809 boards and systems. Very limited quanity, call for information.

MicroSolutions Computer Products, 132 W. Lincoln Hwy, DeKalb, IL 60115, (815)756-3411. Make disk copying program for CP/M systems, that runs on CP/M sytems, UNIFROM Format-translation. Also PC/Z80 CompatiCard and UniDos products.

GIMIX/OS-9, GMX, 3223 Arnold Lane, Northbrook, IL 60062, (800)559-0909, (708)559-0909, FAX (708)559-0942. Repair and support of new and old 6800/6809/68K/SS-50 systems.

n/SYSTEMS, Terry Hazen, 21460 Bear Creek Rd, Los Gatos CA 95030-9429, (408)354-7188, sells and supports the MDISK add-on RAM disk for the Ampro LB. PCB $29, assembled PCB $129, includes driver software, manual.

Corvatek, 561 N.W. Van Buren St. Corvallis OR 97330, (503)752-4833. PC style to serial keyboard adapter for Xerox, Kaypros, Franklin, Apples, $129. Other models supported.

Morgan, Thielmann & Associates services NON-PC compatible computers including CP/M as well as clones. Call Jerry Davis for more information (408) 972-1965.

Jim S. Thale Jr., 1150 Somerset Ave., Deerfield IL 60015-2944, (708)948-5731. Sells I/O board for YASBEC. Adds HD drives, 2 serial, 2 parallel ports. Partial kit $150, complete kit $210.

Trio Company of Cheektowaga, Ltd., PO Box 594, Cheektowaga NY 14225, (716)892-9630. Sells CP/M (& PC) packages: InfoStar 1.5 ($160); SuperSort 1.6 ($130), and WordStar 4.0 ($130).

Parts is Parts, Mike Zinkow, 137 Barkley Ave., Clifton NJ 07011-3244, (201)340-7333. Supports Zenith Z-100 with parts and service.

DYNACOMP, 178 Phillips Rd. Webster, NY 14580, (800)828-6772. Supplying versions of CP/M, TRS80, Apple, CoCo, Atari, PC/XT, software for older 8/16 bit systems. Call for older catalog.

The Computer Journal

Back Issues

Sales limited to supplies in stock.

Volume Number 1:
- Issues 1 to 9
- Serial interfacing and Modem transfers
- Floppy disk formats, Print spooler.
- Adding 8087 Math Chip, Fiber optics
- S-100 HI-RES graphics.
- Controlling DC motors, Multi-user column.
- VIC-20 EPROM Programmer, CP/M 3.0.
- CP/M user functions and integration.

Volume Number 2:
- Issues 10 to 19
- Forth tutorial and Write Your Own.
- 68008 CPU for S-100.
- RPM vs CP/M, BIOS Enhancements.
- Poor Man's Distributed Processing.
- Controlling Apple Stepper Motors.
- Facsimile Pictures on a Micro.
- Memory Mapped I/O on a ZX81.

Volume Number 3:
- Issues 20 to 26
- Designing an 8035 SBC
- Using Apple Graphics from CP/M
- Soldering & Other Strange Tales
- Build an S-100 Floppy Disk Controller: WD2797 Controller for CP/M 68K
- Extending Turbo Pascal: series
- Unsoldering: The Arcane Art
- Analog Data Acquisition & Control: Connecting Your Computer to the Real World
- Programming the 8035 SBC
- NEW-DOS: series
- Variability in the BDS C Standard Library
- The SCSI Interface: series
- Using Turbo Pascal tSAM Files
- The Ampro Little Board Column: series
- C Column: series
- The Z Column: series
- The SCSI Interface: Introduction to SCSI
- Editing the CP/M Operating System
- INDEXER: Turbo Pascal Program to Create an Index
- Selecting & Building a System
- Introduction to Assemble Code for CP/M
- Ampro 186 Column
- ZTime-1: A Real Time Clock for the Ampro Z-80 Little Board

Volume Number 4:
- Issues 26 to 31
- Bus Systems: Selecting a System Bus
- Using the SB180 Real Time Clock
- The SCSI Interface: Software for the SCSI Adapter
- Inside Ampro Computers
- NEW-DOS: The CCP Commands (continued)
- ZSIG Corner
- Affordable C Compilers
- Concurrent Multitasking: A Review of DoubleDOS
- 68000 TinyGiant: Hawthorne's Low Cost 16-bit SBC and Operating System
- The Art of Source Code Generation: Disassembling Z-80 Software
- Feedback Control System Analysis: Using Root Locus Analysis & Feedback Loop Compensation
- The C Column: A Graphics Primitive Package
- The Hitachi HD64180: New Life for 8-bit Systems
- ZSIG Corner: Command Line Generators and Aliases
- A Tutor Program in Forth: Writing a Forth Tutor in Forth
- Disk Parameters: Modifying the CP/M Disk Parameter Block for Foreign Disk Formats
- Starting Your Own BBS
- Build an A/D Converter for the Ampro Little Board
- HD64180: Setting the Wait States & RAM Refresh using PRT & DMA
- Using SCSI for Real Time Control
- Open Letter to STD Bus Manufacturers
- Patching Turbo Pascal
- Choosing a Language for Machine Control
- Better Software Filter Design
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 1
- Using the Hitachi hd64180: Embedded Processor Design
- 68000: Why use a new OS and the 68000?
- Detecting the 8087 Math Chip
- Floppy Disk Track Structure
- Double Density Floppy Controller
- ZCPR3 IOP for the Ampro Little Board
- 3200 Hackers' Language
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 2
- Non-Preemptive Multitasking
- Software Timers for the 68000
- Lilliput Z-Node
- Using SCSI for Generalized I/O
- Communicating with Floppy Disks: Disk Parameters & their variations
- XBIOS: A Replacement BIOS for the SB180
- K-OS ONE and the SAGE: Demystifying Operating Systems
- Remote: Designing a Remote System Program
- The ZCPR3 Corner: ARUNZ Documentation

Issue Number 32:
- 15 copies now available -

Issue Number 33:
- Data File Conversion: Writing a Filter to Convert Foreign File Formats
- Advanced CP/M: ZCPR3PLUS & How to Write Self Relocating Code
- DataBase: The First in a Series on Data Bases and Information Processing
- SCSI for the S-100 Bus: Another Example of SCSI's Versatility
- A Mouse on any Hardware: Implementing the Mouse on a Z80 System
- Systematic Elimination of MS-DOS Files: Part 2, Subdirectories & Extended DOS Services
- ZCPR3 Corner: ARUNZ Shells & Patching WordStar 4.0

Issue Number 34:
- Developing a File Encryption System.
- Database: A continuation of the data base primer series.
- A Simple Multitasking Executive: Designing an embedded controller multitasking executive.
- ZCPR3: Relocatable code, PRL files, ZCPR34, and Type 4 programs.
- New Microcontrollers Have Smarts: Chips with BASIC or Forth in ROM are easy to program.
- Advanced CP/M: OS extensions to BDOS and BIOS, RSXs for CP/M 2.2.
- Macintosh Data File Conversion in Turbo Pascal.

Issue Number 35:
- All This & Modula-2: A Pascal-like alternative with scope and parameter passing.
- A Short Course in Source Code Generation: Disassembling 8088 software to produce modifiable assem. source code.
- Real Computing: The NS32032.
- S-100: EPROM Burner project for S-100 hardware hackers.
- Advanced CP/M: An up-to-date DOS, plus details on file structure and formats.
- REL-Style Assembly Language for CP/M and Z-System. Part 1: Selecting your assembler, linker and debugger.

Issue Number 36:
- Information Engineering: Introduction.
- Modula-2: A list of reference books.
- Temperature Measurement & Control: Agricultural computer application.
- ZCPR3 Corner: Z-Nodes, Z-Plan, Amstrand computer, and ZFILE.
- Real Computing: NS32032 experimenter hardware, CPUs in series, software options.
- SPRINT: A review.
- REL-Style Assembly Language for CP/M & ZSystems, part 2.
- Advanced CP/M: Environmental programming.

Issue Number 37:
- C Pointers, Arrays & Structures Made Easier: Part 1, Pointers.
- ZCPR3 Corner: Z-Nodes, patching for NZCOM, ZFILER.
- Information Engineering: Basic Concepts: fields, field definition, client worksheets.
- Shells: Using ZCPR3 named shell variables to store data variables.
- Resident Programs: A detailed look at TSRs & how they can lead to chaos.
- Advanced CP/M: Raw and cooked console I/O.
- ZSDOS: Anatomy of an Operating System: Part 1.

Issue Number 38:
- C Math: Dollars and Cents With C.
- Advanced CP/M: Batch Processing and a New ZEX.
- C Pointers, Arrays & Structures Made Easier: Part 2, Arrays.
- Z-System Corner: Shells and ZEX, Z-Node Central, system security under Z-Systems.
- Information Engineering: The portable Information Age.
- Computer Aided Publishing: Introduction to publishing and Desk Top Publishing.
- Shells: ZEX and hard disk backups.
- Real Computing: The National Semiconductor NS320XX.
- ZSDOS: Anatomy of an Operating System, Part 2.

Issue Number 39:
- Programming for Performance: Assembly Language techniques.
- Computer Aided Publishing: The HP LaserJet.
- The Z-System Corner: System enhancements with NZCOM.
- Generating LaserJet Fonts: A review of Digi-Fonts.
- Advanced CP/M: Making old programs Z-System aware.
- C Pointers, Arrays & Structures Made Easier: Part 3: Structures.
- Shells: Using ARUNZ alias with ZCAL.
- Real Computing: The National Semiconductor NS320XX.

Issue Number 40:
- Programming the LaserJet: Using the escape codes.
- Beginning Forth Column: Introduction.
- Advanced Forth Column: Variant Records and Modules.
- LINKPRL: Generating the bit maps for PRL files from a REL file.
- WordTech's dBXL: Writing your own custom designed business program.
- Advanced CP/M: ZEX 5.0=The machine and the language.
- Programming for Performance: Assembly language techniques.
- Programming Input/Output With C: Keyboard and screen functions.
- The Z-System Corner: Remote access systems and BDS C.
- Real Computing: The NS320XX

Issue Number 41:
- Forth Column: ADTs, Object Oriented Concepts.
- Improving the Ampro LB: Overcoming the 88Mb hard drive limit.
- How to add Data Structures in Forth
- Advanced CP/M: CP/M is hacker's haven, and Z-System Command Scheduler.
- The Z-System Corner: Extended Multiple Command Line, and aliases.
- Disk and printer functions with C.
- LINKPRL: Making RSXes easy.
- SCOPY: Copying a series of unrelated files.

Issue Number 42:
- Dynamic Memory Allocation: Allocating memory at runtime with examples in Forth.
- Using BYE with NZCOM.
- C and the MS-DOS Character Attributes.
- Forth Column: Lists and object oriented Forth.
- The Z-System Corner: Genie, BDS Z and Z-System Fundamentals.
- 68705 Embedded Controller Application: A single-chip microcontroller application.
- Advanced CP/M: PluPerfect Writer and using BDS C with REL files.

Issue Number 43:
- Standardize Your Floppy Disk Drives.
- A New History Shell for ZSystem.
- Heath's HDOS, Then and Now.
- The ZSystem Corner: Software update service, and customizing NZCOM.
- Graphics Programming With C: Routines for the IBM PC, and the Turbo C library.
- Lazy Evaluation: End the evaluation as soon as the result is known.
- S-100: There's still life in the old bus.
- Advanced CP/M: Passing parameters, and complex error recovery.

Issue Number 44:
- Animation with Turbo C Part 1: The Basic Tools.
- Multitasking in Forth: New Micros F68FC11 and Max Forth.
- Mysteries of PC Floppy Disks Revealed: FM, MFM, and the twisted cable.
- DosDisk: MS-DOS disk emulator for CP/M.
- Advanced CP/M: ZMATE and using lookup and dispatch for passing parameters.
- Forth Column: Handling Strings.
- Z-System Corner: MEX and telecommunications.

Issue Number 45:
- Embedded Systems for the Tenderfoot: Getting started with the 8031.
- Z-System Corner: Using scripts with MEX.
- The Z-System and Turbo Pascal: Patching TURBO.COM to access the Z-System.
- Embedded Applications: Designing a Z80 RS-232 communications gateway, part 1.
- Advanced CP/M: String searches and tuning Jetfind.
- Animation with Turbo C: Part 2, screen interactions.
- Real Computing: The NS32000.

Issue Number 46:
- Build a Long Distance Printer Driver.
- Using the 8031's built-in UART .
- Foundational Modules in Modula 2.
- The Z-System Corner: Patching The Word Plus spell checker, and the ZMATE macro text editor.
- Animation with Turbo C: Text in the graphics mode.
- Z80 Communications Gateway: Prototyping and using the Z80 CTC.

Issue Number 47:
- Controlling Stepper Motors with the 68HC11F
- Z-System Corner: ZMATE Macro Language
- Using 8031 Interrupts
- T-1: What it is & Why You Need to Know
- ZCPR3 & Modula, Too
- Tips on Using LCDs: Interfacing to the 68HC705
- Real Computing: Debugging, NS32 Multitasking & Distributed Systems
- Long Distance Printer Driver: correction
- ROBO-SOG 90

Issue Number 48:
- Fast Math Using Logarithms
- Forth and Forth Assembler
- Modula-2 and the TCAP
- Adding a Bernoulli Drive to a CP/M Computer (Building a SCSI Interface)
- Review of BDS "Z"
- PMATE/ZMATE Macros, Pt. 1
- Z-System Corner: Patching MEX-Plus and TheWord, Using ZEX

Issue Number 49:
- Computer Network Power Protection
- Floppy Disk Alignment w/RTXEB, Pt. 1
- Motor Control with the F68HC11

- Controlling Home Heating & Lighting, Pt. 1
- Getting Started in Assembly Language
- PMATE/ZMATE Macros, Pt. 2
- Z-System Corner/ Z-Best Software

Issue Number 50:
- Offload a System CPU with the Z181
- Floppy Disk Alignment w/RTXEB, Pt. 2
- Motor Control with the F68HC11
- Modula-2 and the Command Line
- Controlling Home Heating & Lighting, Pt. 2
- Getting Started in Assembly Language Pt 2
- Local Area Networks
- Using the ZCPR3 IOP
- PMATE/ZMATE Macros, Pt. 3
- Z-System Corner, PCED/ Z-Best Software
- Real Computing, 32FX16, Caches

Issue Number 51:
- Introducing the YASBEC
- Floppy Disk Alignment w/RTXEB, Pt 3
- High Speed Modems on Eight Bit Systems
- A Z8 Talker and Host
- Local Area Networks—Ethernet
- UNIX Connectivity on the Cheap
- PC Hard Disk Partition Table
- A Short Introduction to Forth
- Stepped Inference in Embedded Control
- Real Computing, the 32CG160, Swordfish
- PMATE/ZMATE Macros
- Z-System Corner, The Trenton Festival
- Z-Best Software, the Z3HELP System

Issue Number 52:
- YASBEC, The Hardware
- An Arbitrary Waveform Generator, Pt. 1
- B.Y.O. Assembler...in Forth
- Getting Started in Assembly Language, Pt. 3
- The NZCOM IOP
- Servos and the F68HC11
- Z-System Corner, Programming for Compatibility
- Z-Best Software
- Real Computing, X10 Revisited
- PMATE/ZMATE Macros
- Controlling Home Heating & Lighting, Pt. 3
- The CPU280, A High Performance SBC

Issue Number 53:
- The CPU280
- Local Area Networks
- Am Arbitrary Waveform Generator
- Zed Fest '91
- Getting Started in Assembly Language
- The NZCOM IOP

Issue Number 54:
- B.Y.O. Assembler
- Local Area Networks
- Advanced CP/M
- ZCPR on a 16-Bit Intel Platform
- Real Computing
- Interrupts and the Z80
- 8 MHZ on a Ampro
- Hardware Heavenn
- What Zilog never told you about the Super8
- An Arbitary Waveform Generator
- The Development of TDOS

Issue Number 55:
- Fuzzilogy 101
- The Cyclic Redundancy Check in Forth
- The Internetwork Protocol (IP)
- Hardware Heaven
- Real Computing
- Remapping Disk Drives through Virtual BIOS
- The Bumbling Mathmatician
- YASMEM

Issue Number 56:
- TCJ - The Next Ten Years
- Input Expansion for 8031
- Connecting IDE Drives to 8-Bit Systems
- 8 Queens in Forth
- Kaypro-84 Direct File Transfers
- Analog Signal Generation

Issue Number 57:
- Home Automation with X10
- File Transfer Protocols
- MDISK at 8 MHZ.
- Shell Sort in Forth
- Introduction to Forth
- DR. S-100
- Z AT Last!

Issue Number 58:
- Multitasking Forth
- Computing Timer Values
- Affordable Development Tools
- Mr. Kaypro
- DR. S-100

Issue Number 59:
- Moving Forth
- Center Fold IMSAI MPU-A
- Developing Forth Applications
- Mr. Kaypro Review
- DR. S-100

Issue Number 60:
- Moving Forth Part II
- Center Fold IMSAI CPA
- Four for Forth
- Debugging Forth
- Support Groups for Classics
- Mr. Kaypro Review
- DR. S-100

Issue Number 61:
- Multiprocessing 6809 part I
- Center Fold XEROX 820
- Quality Control
- Real Computing
- Support Groups for Classics
- Operating Systems - CP/M
- Mr. Kaypro 5MHZ

Issue Number 62:
- SCSI EPROM Programmer
- Center Fold XEROX 820
- DR S-100
- Moving Forth part III
- Programming the 6526 CIA
- Reminiscing and Musings
- Modem Scripts

Issue Number 63:
- SCSI EPROM Programmer part II
- Center Fold XEROX 820
- DR S-100
- Multiprocessing Part II
- 6809 Operating Systems
- Reminiscing and Musings
- IDE Drives Part II

Issue Number 64:
- Small-C?
- Center Fold last XEROX 820
- DR S-100
- Moving Forth Part IV
- Small Systems
- Mr. Kaypro
- IDE Drives Part III

Issue Number 65:
- Small System Support
- Center Fold ZX80/81
- DR S-100
- Real Computing
- European Beat
- PC/XT Corner
- Little Circuits
- Levels of Forth
- Sinclair ZX81

Issue Number 66:
- Small System Support
- Center Fold: Advent Decoder
- DR S-100
- Connecting IDE Drives
- PC/XT Corner
- Little Circuits
- Multiprocessing Part III
- Z-System Corner

Issue Number 67:
- Small System Support
- Center Fold: SS-50/SS-30
- DR S-100
- Serial Kaypro Interrupts
- Little Circuits
- Moving Forth Part 5
- European Beat

Issue Number 68:
- Small System Support
- Center Fold: Pertec/Mits 4PIO
- Z-System Corner II
- PC/XT Corner
- Little Circuits
- Multiprocessing Forth Part 4
- Mr. Kaypro

Issue Number 69:
- Small System Support
- Center Fold: S-100 IDE
- Z-System Corner II
- Real Computing
- PC/XT Corner
- DR. S-100
- Moving Forth Part 6
- Mr. Kaypro

Issue Number 70:
- Small System Support
- Center Fold: Jupiter ACE
- Z-System Corner II
- PC/XT Corner: Stepper Motors
- DR. S-100
- Multiprocessing Part 5
- European Beat

Issue Number 71:
- Computing Hero of 1994
- Small System Support
- Center Fold: Hayes 80-103A
- Power Supply Basics
- PC/XT Corner: Stepper Motors
- DR. S-100
- Moving Forth Part 7
- Mr. Kaypro
- 8048 Emulator Part 1

Issue Number 72
- Beginning PLD
- Small System Support
- Center Fold: Rockwell R65F11
- Playing With Micros
- Real Computing
- Small Tools Part 1
- DR. S-100
- Moving Forth Part 7.5
- 8048 Emulator Part 2

Issue Number 73
- $10 XT
- Small System Support
- Center Fold: 640K XT
- IDE Part 6
- Real Computing
- Small Tools Part II
- DR. S-100
- Mr. Kaypro
- PC/XT Corenr
- 8048 Emulator Part 3

Issue Number 74
- Antique or Junk
- Small System Support
- Center Fold: S-100 Power Supply
- Moving Forth part 8
- Real Computing
- AMSTRAD PCW Now
- DR. S-100
- Mr. Kaypro
- Palmtech CPUZ180
- Disk I/O in Forth

Issue Number 75
- The European Beat
- Small System Support
- Center Fold: Standard Bus I/O
- Moving Forth part 8
- Real Computing
- Embedded Control Using the STD Bus
- DR. S-100
- EPROM Simulator
- High-Speed Serial I/O for the PCPI Applicard
- Disk I/O in Forth, Pt. 2
- T9600 Source Code

Subscriptions (CA not taxable)	U.S.	Canada/Mexico (Surface)	Canada/Mexico (Air)	Europe/Other (Surface)	Europe/Other (Air)
1year (6 issues)	$24.00	$32.00	$34.00	$34.00	$44.00
2 years (12 issues)	$44.00	$60.00	$64.00	$64.00	$84.00
Back Issues (CA tax)	add these shipping costs for each issue ordered				
Bound Volumes $20.00 ea	+$3.00	+$3.50	+$6.50	+$4.00	+$17.00
#20 thru #43 are $3.00 ea.	+$1.00	+$1.00	+$1.25	+$1.50	+$2.50
#44 and up are $4.00ea.	+$1.25	+$1.25	+$1.75	+$2.00	+$3.50

Items: _____

_____

_____ Back Issues Total _____

California Residents add 7.25% Sales TAX _____

Shipping Total _____

Subscription Total _____

Total Enclosed _____

Name: _____

Address: _____

_____

_____

_____

Credit Card # _____-_____-_____-_____ exp ___/___

Payment is accepted by check, money order, or Credit Card (M/C, VISA, CarteBlanche, Diners Club). Checks must be in US funds, drawn on a US bank. Credit Card orders can call 1(800) 424-8825.

TCJ The Computer Journal

P.O. Box 3900, Citrus Heights, CA 95611-3900
Phone (916) 722-4970 / Fax (916) 722-7480

Regular Feature

Editorial Comment

New Editor?

The Computer Corner

By Bill Kibler

Welcome to issue 76's Computer Corner and my last as editor. Next time I'll be doing this column as just a contributing editor. Will it be different, yes and no. Mostly I'll have more time to ponder and do research. I have considered focusing more on one topic which is what I did before becoming editor. But I think I might still hit a few "take this world" topics even still. Take for instance the last Embedded Systems Conference.

ESC

The Embedded System Conference this year, was not where I thought it was. Like many readers, you get so much mail, that when it comes in, you half look at it. I did that with the flyer from the conference. Fortunately, I took it with me, for when I arrived in front of last years meeting hall, no conference. I finally decided to read the flyer very closely only to discover it was in San Jose Convention center, about ten miles east somewhere.

After getting lost more than once by following incorrect signs, I did get to the big new center that was packed with people. The opening day is called October Feast as free beer and munches are provided after 7PM. I over heard a few of the ladies responsible for setting the show up, and they were commenting on how two to three times the number of people showed up that they had prepared for. Needless to say many went away hungry and upset from the food festival portion. I did get my usual beer mug however, so I have no complaint in this area.

As to vendors, they all seemed to be there, the big guys that is. I always find the show a bit strange, since very few small vendors can afford to go there. It pretty much is like most things in America. All the hoopla is over the big vendors which have a small part in the overall market, while the little guys who do all the real work can't even afford to pay the floor space fees. It is sort of like the fortune 100 get all the publicity, while something like 95% all business income comes from companies with less than 25 workers and who most people have never heard of. Embedded control is really like that too.

A FEW Surprises.

I was surprised by a few of the displays. Heading the list was IBM. I didn't expect to see them at all, but they had three booths. The main large one was pushing OS/2 and especially OS/2 Connect. I asked so many question about connect, that I got a free copy of DOS 7 so I would go away. The funny thing about DOS 7 was what I did before the show. I needed a copy of CP Backup and had tried to find one without luck. When I got home and opened my free DOS 7 package, I was pleasantly surprised to find CP Backup is now a part of DOS 7. For those who have not tried DOS 7 by IBM, do so, you will like it. I found it better and I think IBM may keep making it better, while Microsoft has definitely abandoned the DOS market.

Microsoft was there, but so low key, they might as well have not bothered. IBM on the other hand also had a booth for their RISC chip for embedded systems. I think it was a version of the RISC CPU used in their RS6000 system. That concept is sort like the Intel 386 version for embedded controllers. The idea here is to make moving programs from an development platform into the real world simpler and easier. My problem is, who needs another high powered CPU to do what a cheaper and simpler 8 bit CPU can do.

IBM is also starting to confront the operating system people, with their version of a real time kernel. Somewhat based on OS/2, they have a group starting to develop a real time operating system. It is just starting to take off and they said next year it will be a full package. I think they will have problems competing with all the other vendors, especially OS/9 which I think is probably number one. I'll be reviewing OS/9 in a later article.

Of Note

Motorola was there and I asked about their latest HC05 development board. It seems they changed the development package and you must now use an EPROMs to load from. In the past it was possible to do serial loading, but they dropped that. The person there didn't like my comments and reaction to finding out about the change. I guess they don't want to heard about the bad things they do.

Now don't get me wrong, most of the vendors were only selling $10,000 systems and support products, yet a few of the others were able to show up. You do feel like your running around in the land of the giants until you get back into the small booth area. Here you find all the small guys and even a few who's products you might be able to afford.

I was feeling still a little lost till I came across parvus Inc. These people have done their homework and provide an incredible line of products (801 483-1533). I am interested in PC based,

memory mapped 8031 interface cards. Not only did they have one, but I could have my choice of Z180, 68HC11, 8031, or 80166 on ISA or PC104 formats. They had all types of industrial software interfaces and even real I/O cards (relays and opto/isolators). It was like the concept of "one stop shopping." A single vendor who can fill your every need. Pricing was fair, neither high nor low, but the options just got me, like the choice between being a co-processor on the PC bus, or taking the bus over and talking to expansion cards. How about a 68HC11 running the PC bus? Sounds interesting to me!

Another item of note, was Siemens way of getting you a development systems. Instead of charging the usual low show price, they were giving them away. That is right, free if you got four of the six support vendors to sign their card off. The idea here was to make you sit through the support vendors lectures (typically about a half hour each and trying to sell you a $10,000 support package) and then you get the free system. I figured it would have taken me all day to do that, so I didn't get the free unit. Nice idea if you have the time and I expect to see more sales approaches like it next year.

Other fronts

I finally had the chance to use AP Circuits of Canada last month. We had a board needing proto typing and opted to use them. I couldn't be happier. I uploaded the file on Tuesday morning, and we had the board in house Friday morning. Cost was $99 for two boards about 3 by 6 inches each. The work was excellent, although all we had them do was their proto 1 service (two sided no silk or solder mask). Most proto houses want $200 to $300 setup charges before making even one board. Their BBS has free layout programs and information for "would be" circuit builders. Cost wise it beats doing it yourself, for all but the simplest of boards. (AP Circuits (403) 250-3406 voice, (403) 291-9342 BBS or mcmuldd@cadvision.com.)

I am considering trying to setup a deal with them for *TCJ* circuits. The idea would be based on giving them the Gerber files needed to make a *TCJ* board. Readers wanting one of the boards would just call up and give them a credit card number and two days later you get two cards. I am not sure why, but all their runs are based on producing two cards at a time. The cost would be a little higher than us producing a large run of cards, but then we don't get stuck with ordering, shipping, handling and storage cost. You need the board, simply call and zap it is in the mail faster than we would be able to do so.

I would like to hear from readers who have found a cheaper or faster option to get two sided platted through boards. (I checked out several units at WESCON, only the cost of the machines that would put a light plating in the were expensive, and the cost per hole was 5 cents each!) We checked many places and they by far are the cheapest and seem to be very good. Like I say the boards looked great and we had them NOW!

On the software front, I am starting to look at Linux as my operating system of choice. It has so much going for it, it gets really hard not to become hooked on it. You should also get one of the Linux bibles or FAQ reprints. These books (typically 1200+ pages) have tons of useful information about Linux and PC's in general. I have started using mine as a reference book about PC parts. Found out why not to use 3C501 ethernet cards on 486's, too slow and too small buffer space.

Now with Linux and unlike Microsoft products, you always get source code with the CDROM release and it seems to exceed any other product for power and options. If your looking for "how to" write some driver code, or how a CDROM works, consider looking at the source code inside Linux. They talk to most all the PC devices available.

What little I have done so far has impressed me to no end. I recently got a Sony 31B CDROM and had trouble finding a driver that worked on DOS. Linux found it and was able to run from it, right from the floppy.

I tried many of the Sony DOS device drivers, all with no luck. I was told that maybe the older drivers might work. I eventually found a box of cheap disks that were from a sound card. The label said drivers for Sony and Panasonic CDROMs and were 10 cents each. The driver label was SLCD.SYS and it works just fine. Since I have had many requests for CP/M systems being able to talk to CDROMs, I have been a bit more interested in talking to them now. I was about to try writing my own driver, based on using the Linux code until I found SLCD. My early reading however, brought up a minor problem with a CDROM for CPM.

If I remember right, CP/M has a maximum drive size of 32 megs. My studies so far indicate at least 512 megs must be minimum limit. So right off, we got one problem, how to deal with 660 megs of file system. The file format is not even DOS compatible, which is why you need MSCDEX on DOS boxes, to convert the CDROM directory to one DOS can understand. The only approach I can think of would be a separate program that treats the CDROM as a database, where entries are files, allowing you to retrieve records and save them to a local drive. This is some early musings about the problem and I certainly would like to hear your ideas on the subject.

Mail

Since I no longer am editor of *TCJ*, you need to use my personal or Kibler Electronic business e-mail address. The PO Box 535 Lincoln is still a good mailing address, but e-mail is kibler@psyber.com. I have started a personal web page at http://www.psyber.com/~kibler and will try and put things of interest there as well as in *TCJ*'s web page. So far the web page is (*TCJ*'s that is) getting lots of good reviews, even though it is not the fanciest one around. We keep ours a little simpler so people downloading the files don't have a lot of extra text to deal with. That means non-windows users can still get information from us on the net. So till next issue, Keep hacking.

TCJ CLASSIFIED

CLASSIFIED RATES!
$5.00 PER LISTING!

TCJ Classified ads are on a prepaid basis only. The cost is $5.00 per ad entry. Support wanted is a free service to subscribers who need to find old or missing documentation or software. Please limit your requests to one type of system.

Commercial Advertising Rates:

Size	Once	4+
Full	$120	$90
1/2 Page	$75	$60
1/3 Page	$60	$45
1/4 Page	$50	$40
Market Place	$25	$100/yr

Send your items to:
The Computer Journal
P.O. Box 3900
Citrus Heights, CA 95611-3900

Historically Brewed. The magazine of the Historical Computer Society. Read about the people and machines which changed our world. Buy, sell and trade "antique" computers. Subscriptions $18, or try an issue for $3. HCS, 2962 Park Street #1, Jacksonville, FL 32205

Start your own technical venture! Don Lancaster's newly updated **INCREDIBLE SECRET MONEY MACHINE II** tells how. We now have autographed copies of the Guru's underground classic for $21.50. Synergetics Press, Box 809-J, Thatcher AZ, 85552.

THE CASE AGAINST PATENTS Throughly tested and proven alternatives that work in the real world. $33.50. Synergetics Press, Box 809-J, Thatcher AZ, 85552.

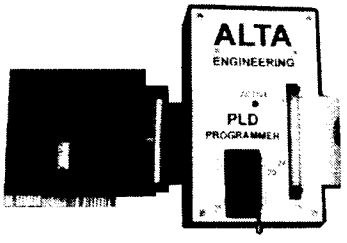
FOR SALE: TRS80 Model III, 48K, 2 floppies, 5meg hard drive, hi res graphics, carrying case, excellent confition, with many books and much software. spare Model III mother board for parts. TRS80 Model 4, works but needs repair. Disk drive for TRS80 Model 100, like new, books, and software. Orginal TRS80 Pocket Computer with cassette and printer interface, software and books. Sinclair ZX80, extra memory module, books and software. Jerry Owen, 1264 T.R. Smith Rd., Ellabell, GA 31308.

TCJ ADS WORK!

Classified ads in *TCJ*
get results, FAST!
Need to sell that special older system - TRY *TCJ*.
World Wide Coverage
with Readers interested in what **YOU** have to sell.
Provide a support service, our readers are looking for assistance with their older systems - all the time.
The best deal in magzines,
TCJ Classified
it works!

FOR SALE: Kaypro hard disk controller cards, WD series for 2/4/10s. Motherboards for all models now in stock. Complete replacement monitors and other new items for your Kaypro needs. Mr. Kaypro, Chuck Stafford. (916) 483-0312, eves/weekends.

THE LOWEST COST!!
PLD PROGRAMMER



Get started with Programmable Logic without the high costs! Partial kits start as low as $40.00 Complete kit as shown $179.00 Plans and SW on disk (PC format) $10.00. Programs the GAL16V8, 20V8 and 22V10. Call or use our FAX-BACK for information.

ALTA ENGINEERING
(860) 489-8003 VISA/MC

DIBs

Electronic Design

Dave Baldwin

6619 Westbrook Dr.
Citrus Heights, CA 95621
Voice (916) 722-3877
Fax (916) 722-7480
BBS (916) 722-5799

TCJ Library Subscriptions

Thank you to our subscriber's that have donated subscriptions to their public libraries around the world.
Paul MacDiarmid
has contributed a subscription to the
Rotorua Public Library
in Rotorua, New Zealand.
This is an excellent way to support *TCJ* and spread the word.

Discover
The Z-Letter

The Z-letter is the only publication exclusively for CP/M and the Z-System. Eagle computers and Spellbinder support. Licensed CP/M distributor.

Subscriptions: $18 US, $22 Canada and Mexico, $36 Overseas. Write or call for free sample.

The Z-Letter
Lambda Software Publishing
149 West Hilliard Lane
Eugene, OR 97404-3057
(541) 688-3563

Advent Kaypro Upgrades

TurboROM. Allows flexible configuration of your entire system, read/write additional formats and more, only $35.

Replacement Floppy drives and Hard Drive Conversion Kits. Call or write for availability & pricing.

Call (916)483-0312
eves, weekends or write
Chuck Stafford
4000 Norris Ave.
Sacramento, CA 95821

TCJ MARKET PLACE
Advertising for small business
First Insertion: $25
Reinsertion: $20
Full Six Issues $100
Rates include typesetting.
Payment must accompany order.
VISA, MasterCard, Diner's Club, Carte Blanche accepted.
Checks, money orders must be US funds. Resetting of ad consitutes a new advertisement at first time insertion rates.
Mail ad or contact
The Computer Journal
P.O. Box 3900
Citrus Heights, CA 95611-3900

CP/M SOFTWARE
100 page Public Domain Catalog, $8.50 plus $1.50 shipping and handling. New CP/M 2.2 manual $19.95 plus shipping. Also MS-DOS software. Disk Copying including AMSTRAD. Send self addressed, stamped envelope for free Flyer, Catalog $1.00.

Elliam Associates
Box 2664
Atascadero, CA 93423

Kibler Electronics

Hardware Design &
Software Programming

8051, 6805, Z80, 68000, x86
PLC Support and
Documentation

Bill Kibler
P.O. Box 535
Lincoln, CA 95648-0535
(916) 645-1670
e-mail: kibler@psyber.com
http://www.psyber.com/~kibler

MORE POWER!
68HC11, 80C51 & 80C166

- ☑ More Microcontrollers.
- ☑ Faster Hardware.
- ☑ Faster Software.
- ☑ More Productive.
- ☑ More Tools and Utilities.

Low cost SBC's from $84. Get it done today! Not next month.
For brochure or applications:
AM Research
4600 Hidden Oaks Lane
Loomis, CA 95650
1(800) 949-8051
sofia@garlic.com

S-100/IEEE-696

IMSAI Altair
Compupro Morrow
Cromemco
and more!

Cards • Docs • Systems
Dr. S-100
Herb Johnson,
CN 5256 #105,
Princeton, NJ 08543
(609) 771-1503
hjohnson@pluto.njcc.com

THE FORTH SOURCE

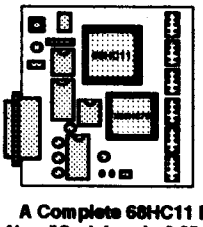
Hardware & Software

MOUNTAIN VIEW PRESS

Glen B. Haydon, M.D.
Route 2 Box 429
La Honda, CA 94020

(415) 747-0760

$79.95 68HC11
8K EEPROM for More Program Space!
SER-8C, Optional 8K Serial EEPROM $10
Single Board Computer
SBC-8K



- Small Size, 3.3"x3.6"
- Low Power, <60 ma
- 8192 Bytes EEPROM
- 256 Bytes RAM
- DB-9 RS-232
- 24-TTL I/O Bits
- 8-A/D Inputs
- Power Reset Circuit
- 8 Mhz Clock
- Log Data with SER-8C

A Complete 68HC11 Development System.
New "CodeLoad+ 2.0" and Sample Programs.
No EPROMs or EPROM Programmers!
500 Pages of Manuals, 3.5" Utility Disk.

LDG Electronics
1445 Parran Road
St. Leonard, MD 20685
Voice / Fax
410-586-2177