The COMPUTER JOURNAL

Programming - User Support
Applications

Issue Number 64 November/December 1993 US$4.00

Small System Support

Z-System Corner

DR S-100

Real Computing

Support Groups

IDE Drives part III

Small-C?

Moving Forth Part IV

Mr. Kaypro

ISSN # 0748-9331

68XXX COMPUTER PRODUCTS From Peripheral Technology

68000 System Boards with 4 Serial/ 2 Parallel Ports, FDC, and RTC.

PT68K4-16 with 1MB **$299.00**

PT68K2-10 w/ 1MB (Used) **$149.00**

REX Operating System Included

OS9 V2.4 Operating System **$299.00**
With C, Editor, Assembler/Linker

SCULPTOR V1.14:6 for Business Software Development - requires any version of OS9/68K. **$79.00**

Other 68XXX products available!

1480 Terrell Mill Rd. #870
Marietta, GA 30067
404/973-2156

Cross-Assemblers as low as $50.00
Simulators as low as $100.00
Cross-Disassemblers as low a' `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.

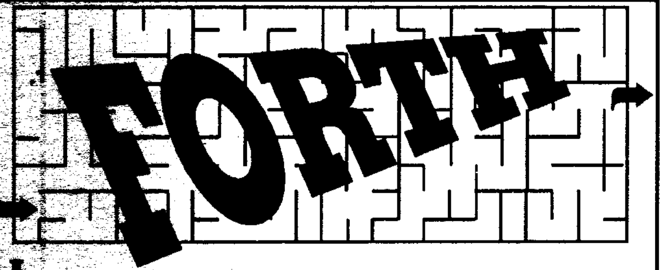
Quality Solutions
PseudoCorp has been providing quality solutions for microprocessor problems since 1985.

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
716 Thimble Shoals Blvd, Suite E
Newport News, VA 23606
(804) 873-1947 FAX: (804)873-2154



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.

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 (pw=DDT)
(MABOS on PC-Pursuit)

The Computer Journal

Founder
Art Carlson

Editor/Publisher
Bill D. Kibler

Technical Consultant
Chris McEwen

Contributing Editors
Herb Johnson
Charles Stafford
Brad Rodriquez
Ronald W. Anderson
Tim McDonough
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 535, Lincoln, CA 95648, (916) 645-1670.

Subscription rates within the US: $24 one year (6 issues), $44 two years (12 issues). All funds must be in U.S. dollars drawn on a U.S. bank.

Send subscription, renewals, address changes, or advertising inquires to: *The Computer Journal*, P.O. Box 535, Lincoln,CA 95648.

TCJ The Computer Journal

Issue Number 64 November/December 1993

EDITOR'S COMMENTS

Welcome to issue number 64, a big issue in many ways. There are many new items to discuss and lots of regulars as well.

The newest member of *The Computer Journal* writing staff is Ronald Anderson. His letter of introduction starts the Reader to Reader section. Ron comes from the 6809 world by way of the old *68 Micro Journal*. Many of our readers may remember him form his regular column that appeared in that magazine. Since that time Ron has continued his 6809 work while porting many of his Flex and OS-9 products to MSDOS machines.

I have called Ron's column "Small System Support", since he indicated that his many years of designing and working on embedded controllers will be his primary beat. Like many of our other writers, he will actually cover many topics, from reviewing and helping others to keep their old 6809 platforms running, to broaching old ideas but using new platforms for their solutions.

If you have any doubts about his ability to meet our needs and challenges, I am sure they will be all gone after reading both his letter and first article. Welcome aboard Ron!

Our Reader to Reader has more than Ron's letter. There is plenty to keep you interested and educated. Brad Rodriguez has an announcement of a contest that should perk up your attention if your do embedded controls. All starts on the next page...

I have finally gotten around to adding more information to JW Weavers column, so that you can find all of *TCJ* author's contact information in one place instead of their own articles. So check out the Support Groups section and see if your local group needs to have their added.

Charles Stafford shows us how to upgrade some Kaypro's, while Jay Sage provides part 2 of using PMATE MACROS. For more regular support, Rick Rodman fills us in on how well LINUX worked for him. Rick also explains some insides of LINKING and a possible idea doing with linking all our computers together.

On the special side we continue with part 3 of Tilmann Reh's series on IDE drives. We get more information on their commands and a sample PASCAL program to see if we can read our drives identification information. This is all part of his making sure we understand enough about IDE drives to be able to write our own interface device drivers.

We follow Tilmann's article with more disk drive interfacing, by having Herb Johnson continue his series on developing your own disk drive software or BIOS. This time we review some of the CP/M commands and hardware arrangements needed to develop the BIOS code.

The long awaited Small-C article is here. I must confess that I feel a part 2 coming on. Like any complex topic, choosing a universal language is no simple task. The article covers some good and bad points of C and other alternatives. I conclude the article with short descriptions of The C Users Group disks that have a version of Small-C on them. Since not all versions were covered, I hope that some of you can send in more reports (like J. G. Owens) on other versions you have personal experience with.

Fortunately I asked Brad to keep it short this time (he also had to present some studies at the University) so his article this issue deals with moving Forth and some decisions you need to consider. My objective over the next few issues is to try and get not only Brad but everyone else to give me about 4 printed pages per issues.

It appears I have several articles I would like to use but seem to be short a page or two. I had to add two extra pages just to get our regulars. My position is to print whatever it takes, but if possible I think about 4 *TCJ* pages each should be enough. That means of course that some writers will do a 3 pager one time and 6 the next. Hopefully not all will do 6 pages at the same time or I am in big trouble.

Speaking of troubles, I have made some important decisions based on other happenings that all our readers need to find out about. They are all covered in my Computer Corner. The most earth shaking is finally deciding to support PC/XT machines! That support will be limited and clearly defined as explained in my column. So check out our new position and more.

That is about all your going to get this issue, and if it isn't enough, it is only because your comments and letters haven't been received yet. *The Computer Journal* has always been your magazine, and if you don't see or like something, just drop me a note and we will consider our options. I can't guarantee every request will be granted, but then I can't grant anything I don't know about. So write and thanks for supporting *TCJ*!

READER to READER

Letters to the Editor

All Readers

MINI Articles

Dear Bill,

I received the three issues of *TCJ* yesterday. I ve read the tenth anniversary issue through cover to cover, and read a lot of the others as well. I wonder why I hadn't run across your publication before. Perhaps it was because of the emphasis on CP/M and FORTH that nobody called it to my attention. No matter. While I was waiting for the back issues I began to write an introductory column. It turned out to be a history of the 6800/6809 systems that parallels the one in the tenth anniversary issue covering the 8080 /Z80 machines. I had already done it as an ASCII file when I discovered yesterday that you use WordStar 7.0, which I am using today to write this letter. I have a package called Emulaser with which I can do wonders with Post Script fonts etc. The printer is a Citizen GSX-140.

The material I've submitted (on the enclosed disk) is really two subjects. First there is some history, but I found myself reading all my old stuff in *'68' Micro Journal* and remembering all that good software, most of which I still have around, for the 6809 FLEX system (which I also still have around). I couldn't help preparing a list of suppliers and their last known address. Maybe this will help some of your readers who have stumbled onto 6809 systems and wonder what to do with them. If you would like to separate these two subjects, just do so, or let me know and I will do it for you. I would hope the subject matter won't undermine or disturb anyone else you have lined up to do 680X material too. I'm open to any and all suggestions.

I noted a bit of dissatisfaction on the part of one reader who sent all his back issue back to you. Being an outsider as far as the CP/M world is concerned, I can easily agree with that reader. One rule ought to be not to use a mnemonic without spelling it out at least the first time it is used. You have new subscribers all the time, and even in the third part of a multi-part article, the author ought not to assume that the reader understands a mnemonic automatically.

I have some mixed feelings about your mix of articles and your mission (which I may not understand correctly). I couldn't agree more with the idea that these old 8 bit systems are easier to understand and better computers on which to learn what computers are all about. If these are really to be found for under $100 at swap meets, they would certainly be ideal for someone on a small budget to use to learn about computers. On the other hand I detect something between disdain and contempt for those "IBM" boxes. Come on, folks. Believe it or not, programming an 8250 (16450 in the newer serial applications) is no harder than programming a 6850 serial interface, or, I'm sure, the Intel equivalent. Give me a chance with some articles and I'll prove that. I recently needed to write a pair of programs to allow a consulting client to transfer a large amount of data files from a SWTPc system using 8" floppies, to a new 486 system, on 3.5" 1.44 Mbyte disks. I of course chose a serial interface at each end, in order to keep it simple. The 6809 end had a 6850 which can be initialized in a couple of lines of code. The baud rate is set with jumpers on the serial board. The IBM end didn't yield to my attempts to set the port up using the MODE command from DOS nor did it work as indicated when I tried to use some of the Borland Turbo C library functions to set it up. When I found an 8250 data sheet, I found it no more difficult than the 6850, except of course that it has more steps since the baud rate is programmable too. I was even able to use the RTS handshake (Request TO Send) from the IBM to the CTS (Clear To Send) of the 6809 so I could stop the data flow while writing data to the 3.5" floppy. It just isn't that much harder and you don't have to go through high level library functions OR BIOS functions to do it.

When I translated my editor PIE to Turbo C for IBM use, I found that writing to the screen using BIOS calls was unacceptably slow. (This was back on an old XT about 42 times slower than my present 386DX-40). It was not much of a project to write directly to VIDEO RAM. The routines were easier than trying to do a screen oriented editor on a serial terminal and I didn't have to resort to Assembly code to do the job, and do 20 terminal configuration files for 20 different terminals.

I also detect contempt for "C". C, I've found just as viable as a procedural language as is Pascal. I cut my teeth on Pascal years ago, and at first found C to be quite cryptic. I have news for you. When I translated PIE from Pascal to C. I found that I could do it line for line, though not getting into the "true spirit" of C. You don't have to use the ridiculous features. I read about the question mark operator and promptly forget it. In Pascal you have to pre-read a character from an input file before you can go into a read loop to read "while not end of file". In C you can combine the getting of the character with the test for end of file, and you don't have to have two places where a character is read from file.

I guess, my point is that surely the good old simple little computes are nice. My PAT editor for the 6809 is around 29K of object code, about 50 pages of source in PL/9. The C version for the IBM ended up being about 49K of object code. A screen editor with reasonable features doesn't have to be 500K of object code. Pat DOS version has more features such as the use of color for marking blocks. It runs very much faster on a reasonably fast clone. I can edit a huge file and have it all in the edit buffer at once as opposed to pulling in a section at a time and having to write it to disk before pulling in another part. Hardware progress is hardware progress. Let's not ridicule it.

On the other hand regarding software, I do see a conspiracy on the part of the software suppliers to make their programs so large as to wipe out the competitors who can't put 20 or 30 programmers on a project at one time. Buy one of the Borland products and I guarantee that before you can get it installed you will receive an offer for an upgrade for only $89.95 plus $12.95 shipping and handling. At this point, though I may have moved along a bit farther than some of you, I draw the line at Windows. Windows is an unnecessary complication. I'd much rather type a command and hit enter, than have to point and click my way through 6 menus to do the same thing!

Regarding cost, you guys out there who are now finding the bargain oldies are really in luck. I figure my 6809 system had about $4000 invested in it in its heyday. I have three IBM clones that haven't cost that in total, thanks to finding old hard drives at bargain prices, and not throwing away old motherboards when they are upgraded, etc.

Well, I've more than said my piece. If you can live with a cantankerous old opinionated grouch (when it comes to computers particularly), I would be pleased to write a regular column for you. Comments and criticisms will be greatly appreciated. I'd like to do what you want and need for the most part, though I reserve the right to needle a bit as I have done in this letter. I see as usual, I will end up with my signature on a fresh page!

Yours truly, Ron Anderson.

Welcome to TCJ Ron!

Your line about being a bit opinionated (notice I ignored the OLD and Grouch part) fits right in with most of us at The Computer Journal. As you will find out, I am a one person staff, so when I talk about "US" it is the readers who are the other members of my staff. I rely on them to correct me if I error (or our writers) too far in any direction. Your background and comments (your history is just wonderful) are going to be greatly appreciated.

Let me say you have correctly understood a number of TCJ's positions. Learning about hardware and software interfacing on any machine is transferable to any other platform. My problems about recommending IBM Clones for learning is based on experience. I have had BIOS's turn off interrupts while trying to do I/O. My OS/2 platform crashes two or three time a day. Yes the hardware is much better (who can complain about having a mainframe on your desk), but more problems and a much higher level of understanding came with them. I guess the main issue for TCJ readers is just cost. The older machines are cheaper still (but clone parts are getting very close) and the test equipment is by far cheaper. Now if all you want to work on is 4 Mhz clones a simple 35MHZ oscilloscope will do just as well here as on a 4Mhz Z80. It will prove useless on a 35Mhz 486 Clone however.

The Computer Journals main objective is to teach our readers how to use and repair (or keep running) ANY machine they chose to use. Since all other computer magazines in the world have refused to support or even acknowledge that people still use Z80 and 6809 based machines for every day tasks, we have decided to support as best we can these wonderful old machines. Many of our readers have found these old machines to be works of art and are fascinated with them from a collectible aspect (much as an antique car collector enjoys his collectible old machine). We are limited in the amount of coverage we can give any platform however, due mostly from lack of advertisers and limited number of pages. I felt strongly that several other magazines provided (or said they provided) technical support of PC clones and as such I have no desire to compete head on with them.

It appears now that the support is for the new 386/486 machines only, and not the older and obsolete PC/XT platforms. So starting now we are adding the obsolete versions of PC Clones to what we support. This means that you can digress and talkabout those machines as well as the better 6809 and other embedded systems of which you are so familar. We can provide support for all the older systems (with your help Ron) that can be found no where else. Over time I expect our readership to grow as the word spreads that this is the place to come for real hands on support and learning information that can be used on any platform.

Thanks for not giving up completely on the old machines Ron! Welcome again to TCJ. Bill Kibler.

Mr. Kibler,

I am impressed with *TCJ*. I recently started publishing a Tandy Color Computer, OS-9, and OSK (OS-9/68000) magazine myself. So far, all is doing very well. "*the world of 68' micros*" will support other 68xx(x) series microcomputers and controllers and other operating systems for these devices as it grows.

Enclosed you will find ad copy for *TCJ* along with payment for the first two ads. I've also enclosed the latest copy of "*68' micros*".

Would there be any objection to my printing a story about the 6809 multi-processor system? All I would like to do is print a description (no schematics) and who to contact for the circuit boards. I realize the boards aren't in production, and may not be produced... part of the reason I'd like to run an article, to help get enough boards to do the project. It is a little beyond my capabilities and interests at the moment, but some of my other readers may be interested. It is possible that OS-9 could be patched to operate on such a system.

I enjoy classic computers a lot myself, having my CoCos and a pair of T/S models (ZX-81 and T/S 1500). I got the CoCo after discovering what would be involved in connecting a "real" printer to the 1500.. I intended to use it for word processing. After researching the home market at the time (1984-85), I decided the CoCo offered the most bang for the buck and cheapest expandability. I didn't learn much from Tandy, but from magazines and users.

Keep up the good work!
Francis G. Swygert
Editor, "*68'micros*"

Thanks Francis for the copy of your magazine. I was impressed that you have been able to get so much going in just a couple of issues. Your action supports my position that it is only a period of time before "collectible" computers becomes an industry of itself.

We do not allow full reprinting of our articles, however reviews or synopsis are allowed. The Z-Letter prints a review in each of his issues about what other support magazines and newsletter have done in their last issue. Since most of our support is by word of mouth, any plugs we can get and give each other will be greatly appreciated.

You will enjoy the next issue, as it is a special on the ZX-81. I have received lots of material on the ZX-81 and will try and review it all (not an easy task!) next issue. So, welcome to TCJ as both a reader and advertiser, not to mention a supporter of classic systems. Bill.

Dear Bill,

I had almost decided not to renew *TCJ* - I'm not a big Intel/Zilog/CPM fan. So issue number 61 was going to be my last; not because the magazine wasn't well done or lacked meat, it just wasn't useful to me.

Then I picked up that issue 61, and turned to the end of the magazine, and found just what the doctor ordered! Yes, yes, yes, do it! Your ideas concerning the use of Small-C and Forth are right on. I suggest you use mainly Small-C because, while I personally use Forth, most people find it objectionable. And a Small-C complier can probably be made to generate tighter code than Forth.

I would also encourage you not to go overboard on picking up the pieces of has-been publications (e.g. *68xxx Micro-Journal*, et. all). Here again, I prefer 68xxx architectures, but I want to move forward. Let's do the Small-C tools set, and then move to the development of an OS that meets todays standards (real-time, multi-threaded, multi-tasking). If it's done modular, with different (small) kernels for different folks, and written in our Small-C, it can be easily adapted to everyone's needs.

The critical thing in my mind is to keep things from becoming so complex that one person of average intelligence can no longer get wrapped around it all. Keep the compilers and OS reasonably simple, and let the complexity be added at the application level by only those who need it.

Finally, besides your very admirable objective of education, also be aware of major problem areas of your readers. Oh why oh why can I not find a supplier that will sell me by mail order a couple MC683xx chips? Why do I have to still be stuck with articles that use almost two decade old 6809 chips when 683xx are so much more desirable? One 68306, a couple of DRAM chips, and one EPROM is all you need to make a fantastic system; it even has the DRAM refresh built in! As far as I'm concerned, finding friendly suppliers is right at the top of my problem list; there are many publications that help providing me with the information I need, but getting the materials I need to make use of that info is a serious problem. Addressing this problem is worth a monthly column in my opinion.

Well, you did ask for our input! Seriously, I very much like your suggested new direction. I'm willing to continue my support.

Sincerely, David F. Klink

Thanks for the support David. Actually my search for an universal operating system is a very old project. In considering using Small-C you had better read the Small-C article in this issue. Those with experience using it have made some good comments and expressed their concerns. It is beginning to look like if we are to use Small-C some form of multi-pass compiling will be needed (as done with OS-9 C compiler). I think any major operating system project should be handled the same. Break the project and the pieces into small parts for different people to work on. The results too would be in small units that can be loaded only if needed.

*My getting rights to back issues of other magazines is just making sure they vanish for good. Lots of very important work was done back then which I would hate to see in the trash dump. Yes we are moving forward, but looking back on the past and building on what was done is important. That goes for the latest 68K chips which I think (and hope) we will have an article about soon. Peter Stark of SK*DOS is considering doing just such an article, but is just not sure our readers want it.*

You correctly identified the problem with using new chips and why Brad is using 6809s in his project. Let me say however, that the 6809 is considerably better than most users give it credit for being. But Brad is using it mostly because they are cheap and available. One of my constraints on articles, is the parts must be easily available. I have been personally looking for a source of 68306's with out any luck. I am sure I could get samples, but that does our readers little good. Since I can't find a source I can't even determine cost. This problem explains one of the reasons we push using older systems for projects. With old units, the parts are still avail-

able and inexpensive as well. Also remember that what you learn can be transferred to the new systems. Like the 68306 which I know Brad would rather use if it was easily available.

We like things simple at TCJ, and your letter and support is greatly appreciated. How about some comments on the Small-C project from you? Thanks for the letter! Bill.

Dear Bill,

Attached is the copy of my reply to John Butler in London. I have a fairly extensive collection of some of the older computer magazines and may be able to help out anyone looking for articles in them.

Dear John:

Noting your letter in the current issue of *TCJ*, asking for help in locating several articles on old issues of *Microsystems* and *Interface Age*, I searched my chaotic attic and came up with the following:

Relocating Assemblers and Linkage Editors: part1, *Microsystems*, June 1983. Ibid., Part 2, October 1983. Ibid., Part 3, January 1984.

Structured Programming with Microsoft M80 Assembler: *Micro/systems Journal*, July/August 1985.

I'm enclosing photocopies of the articles, and hope you find them useful.

I'm pretty sure I also have the issue of Interface Age with the article you're looking for, but so far haven't been able to put my hands on it. If it turns up, I'll copy the article and send it along to you.

With all good wishes, Norman F. Stanley, Rockland, ME.

Thank you for helping out John, Norman! It is not often that I actually find out if someone got help or not. Keeping track of all the back issues and many of those very important articles has actually been suggested by a few of our readers, as something TCJ should be doing. since my space and time is limited, I have to rely on people like you. So again thanks for your help! Bill.

Dear Bill,

I am working on a project to put together a public domain/shareware CD-ROM containing articles on the subjects of Macintosh Computer Data Acquisition and Data Analysis to be published by the Macintosh Scientific and Technical Users Association. The CD will be promoted to the MacSciTech members (1500), the readers of the SciTech Journal (10,000 per issue) and sold at various MacSciTech conferences.

I am reviewing previous issues of your publication looking for articles I'd like to include. I will contact you later with specific requests. Please let me know if you are willing to grant permission for reprints.

Would it be possible for you to give me articles in computer-readable format? I would prefer Macintosh text files, but I can deal with DOS text files if necessary.

You may contact Mike Duncan of MacSciTech if you have any questions.

Gus Calabrese.

Glad to hear that your putting together a Mac CDROM, Gus. Our CP/M CDROM should be done about now. I have been using CDROMs lately and find them just wonderful. I was unable to get more than a sample of TCJ into our ROM. My time and amount of computer ready data is almost non existent. I plan to contact both previous editors and get what ever disks they have, but for now, I only have printed back issues and am not about to retype them all. As far as rights to reprint the entire article, no, because I still sell all back issues and they are a major reason for TCJ breaking even (and keeping cost down). You can print a synopsis or review with information on how to get back issues without my permission if that will help. Please let us know how the CDROM comes out, and thanks for supporting TCJ. Bill Kibler.

To: B.KIBLER
Sub: news release

Could you be...

THE WORLD'S FASTEST PROGRAMMER?

On March 8, 1994, the ACM Special Interest Group on Forth (SIGForth) will invite fifty programmers to joust with computer and compiler, to vie for the title of "World's Fastest Programmer." Individuals and teams will compete to program a physical "gizmo" in the shortest possible time, using the computer and language of their choice.

First held in 1988 under the auspices of the Forth Interest Group, the World's Fastest Programmer contest has traveled from California to Europe, and now arrives in Phoenix, Arizona for the 1994 Symposium on Applied Computing. This symposium is jointly held by six Special Interest Groups of the Association for Computing Machinery:

SIGAPP	(Applied Computing)
SIGAPL	(APL)
SIGBIO	(Biomedical Computing)
SIGCUE	(Computer Uses in Education)
SIGSMALL/PC	(Personal and Small Computers)
SIGFORTH	(Forth)

Now's your chance to prove the worth of your platform, language, or programming methodology! Any computer that supports a parallel printer can be used. Entrants (individuals or teams) need not be members of ACM or SIGForth, but only fifty can compete, so register now!

REGISTRATION

The contest registration fee is $25 (U.S.), payable to ACM SIGForth (U.S. checks or money orders only, please). Send your name, address, and a check or money order to the contest chairman:

Brad Rodriguez
Box 77, McMaster University
1280 Main Street West

Hamilton, Ontario L8S 1C0 Canada

email: B.RODRIGUEZ2 on GEnie, or b.rodriguez2@genie.geis.com on Internet.

(U.S. entrants note: first class postage to Canada is 40 cents!)

For team entrants, only one member need register. You may register by email; however, your registration will not be accepted until the $25 fee is received. Participation is limited to the first fifty registrations received by December 31, 1993. Cancellations after December 31, 1993 will forfeit the registration fee. (The contest organizers reserve the right to extend the registration period at their discretion.)

The contest will be held at the Phoenix Civic Plaza, in conjunction with the 1994 Symposium on Applied Computing (SAC '94) and the 1994 ACM Computer Science Conference (CSC '94). While in Phoenix you may wish to attend SAC '94. For registration information, contact:

Ed Deaton, Conference Director
Department of Computer Science
Hope College
Holland, Michigan 49422 USA

email: deaton@cs.hope.edu

SAC '94 can also provide information on housing in Phoenix.

CONTEST RULES

The object of the contest is to solve a real-time programming problem in the shortest time. This problem will involve a hardware ''gizmo'' to be controlled by a computer. Rules:

1. Entrants may be individuals or teams. Teams may have any number of members.

2. The ''gizmo'' will be supplied by the contest organizers at the commencement of the contest. Only one gizmo will be supplied to each entrant (i.e., only one gizmo per team).

3. Entrants may use any programming language(s).

4. Entrants may use any computer(s), and any number of computers. Entrants must supply their own computer(s).

5. Entrants must ensure that their computer has an interface suitable for the gizmo, as follows:

a. The computer must provide 8 bits of parallel output, and 1 bit of parallel input which can be read by software (i.e., not an interrupt input).

b. The gizmo will use a standard Centronics-type 36-pin female connector, and will electrically resemble a standard ''IBM PC compatible'' parallel printer. Pins 2 through 9 on this connector (D0-D7) will be the 8 data inputs of the gizmo. The data output of the gizmo will appear simultaneously on pins 11 (BUSY), 12 (PAPER END), and 13 (ON LINE). Ground will be pins 19 through 30. TTL levels will be used. All other pins will be unconnected.

c. Entrants must supply their own cables. A cable that connects the computer to a standard parallel printer should be satisfactory; however, it is the responsibility of each entrant to verify that the signal assignments given above are compatible with their computer's parallel port, and to provide any necessary adapters.

d. The gizmo will not require power from the computer.

e. The contest organizers will exercise care in the design and construction of the gizmo. However, the organizers assume no responsibility for any damage to any entrant's computer(s), caused by connection to the gizmo.

6. No other information about the gizmo will be provided until the start of the contest.

7. No information about the problem to be solved will be provided until the start of the contest.

9. The winner of the contest will be the entrant who completes the assigned problem in the shortest time after the start of the contest. Completion criteria will be provided in the problem description; but no entry will be deemed complete until so pronounced by the Judges.

10. Entrants must provide their source code to the contest organizers at the conclusion of the contest. Entrants will retain full rights to their work. However, by entering this contest, each entrant agrees to grant the contest sponsors and contest organizers unlimited right to use, publish, or distribute their contest entries. (In particular, the contest organizers intend to publish the winning entry in a suitable journal.)

11. All disputes about the interpretation of these rules, and all other matters pertaining to this contest, will be decided by contest judge(s) to be named by the contest organizers. The decision of the judge(s) will be final.

12. Participation will be limited to the first fifty (50) entrants whose applications are received by 31 December 1993.

=END=

Thanks for the notice Brad. I was unable to attend the gizmo contests, but one of our local Forth members did. He said it was just great to watch how all the different teams worked. He really learned a lot and enjoyed himself as well. The next couple of Forth meeting we had our own mini-contests as we all brought one item or another. I had a stepper motor driven by 68HC11 using Forth. The object was how little code was needed to make it move to any given location. Was simply fun and livened up the meeting.

Now Brad, I expect to get an article from you on this, correct? Bill.

Dear Bill,

Please change my address to......
Note that Sound Potentials, which used to distribute a collection of public domain CP/M software, is no longer in business. The software collection was acquired by Lambda Software Publish-

ing, i.e. Mr. David A.J. McGlone, who publishes the *Z-Letter*, and remains available from him.

I have subscribed to *TCJ* since issue #48, primarily for articles relating to CP/M and Z-Systems. My CP/M systems include a hopped-up Kaypro 4-84, which has a 1 MB Advent RAM disk, two quad density drives (782K) and one double density drive. I also own two Epson PX-8 Geneva CP/M portable computers with the wedge attachments for extra RAM. I now frequently use an 80386DX-40 Mhz IBM Clone, on which I run a registered version of MYZ80. This makes a high performance CP/M system, especially when running ZCPR34. I would be interested in any information or articles concerning MYZ80.

My wife, Diana, has been promoted in her company, hence our move to Richmond. I will seek computer programming work there. I recently completed a B.S. degree in Computer Information Systems from Regents College, 1450 Western Ave., Albany NY 12203-3524. This is a fully-accredited program of the University of the State of New York that can be completed without ever going to Albany or even to New York State. If any *TCJ* reader has been contemplating a B.S. in computing, I recommend this "distance learning" program. If you have any existing college credits, they can be applied to the program. I already had a B.A. degree with a freshman year that was filled with engineering courses. I was able to complete the B.S. degree in one year by taking seven courses plus G.R.E. in Computer Science. The Computer Information systems degree differs slightly from a standard Computer Science degree in that the mathematics requirement is a bit lower, and there are additional requirements for Systems Analysis and Design, and Database Management Systems.

Very truly yours, Richard E. Brewster.

Thanks for the information Richard. I had seen in The Z-Letter that you had transferred Sound Potential's software to David. David is doing well in his new Eugene location.

It looks like you have a good assortment of CP/M systems. Yes, MYZ80 is great, especially running ZCPR. We have only had one past article on it, but would love to see more!

The degree idea sounds great and hopefully some of our readers will give them a ring to find out more. Thanks. Bill.

Dear Bill:

Re: Superbrain Hard Disk Boards, Schematics, ETC.

I was pleased to have a chat with you a couple of days ago when I phoned to commence my first subscription to *TCJ*.

I use Intertec Superbrains and, unlike most people, do not use an MSDOS, or any other, computer. The Superbrains are very heavily used for DBASE II programmes for agricultural research, for word processing with Magic Wand/ Peachtrext, and to a lesser extent for computer communications. Some day when time permits, I will be converting a number of BASIC statistical programmes I wrote for the Hewlett-Packard 9830 to run under MBASIC on the Superbrains. I have just bought Z-SYSTEM and am struggling to install it.

I use CMC controller and interface boards to run two 10 MB Hard disks on the Superbrain, each configured as two logical disks with 1024 directory entries.

Since losing one of my hard disk controller boards when my hard drive power supply blew up, my major problem has been one of trying to find a good spare CMC controller board or two. So far I have had no luck.

The CMC boards I am looking for were originally used with CMC OEM Superbrains called "Super Five", "Super Ten", or "Super 20" and were also used to upgrade many factory standard Superbrains and Compustars.

The CMC controller board set consisted of 2 boards, usually green. Both bore the trade name "act". The smaller board also said "HOP-8". The larger board said "5MSDC" and, immediately beneath, "REV-3".

I am also interested in Superbrain (or Compustar) boards and accessories of all kind, schematics for the CMC hard drive boards, schematics for a Superbrain II, information for adding IDE hard disks, adding a parallel port, etc. for the Superbrain.

If anyone knows of available CMC or Superbrain boards (or the whole hard drive unit or computer), or has information regarding any of the above, I would appreciate being informed. I can be reached as follows:

Brian Murphy, Fidonet 1:153/151, or 12521 Cathy Cr, RR#7, Mission, B>C> V2V 6H5, Canada. Call (604) 462-7057.

I am eagerly looking forward to a year of receiving *TCJ*.

Sincerely yours, Brain C. Murphy.

OK Brain, I can help for many of what you need. I have several Superbrain schematics and a whole Superbrain QD with parallel port. I did the parallel port some time ago and wrote an article in issue #32 as part of my Computer Corner. One problem with my 10 years of Corner's is that they are not indexed by topic. I did several projects and have to thumb through them to find which issue the discussion is in. I guess that is a good reason to just have all of the back issues. I never seem to be amazed at what I find while looking for something else.

I had intended on keeping the Superbrain, but lately I am a bit over loaded and might consider selling it. How much and then how to get it to Canada is another problem. Lets hope you can find help a bit more locally. As for hard drives support I have none, either drives or support. Let us know if you find help however.

Thanks for still using classic machine! Bill Kibler.

Dear Bill:

Enclosed is my check in the amount of $44.00 to renew my subscription for another two years. though my ''EXP'' is 65, I am renewing early to avoid any chance of missing an issue.

I am operating under TurboDos in an S100 configuration. My system is presently set up for eight users and three printers. TurboDos could handle sixteen of each. Would you be interested in an article about this alternate CPM multiuser, multifunctioning system? I may even have schematics on my Teletek and Earth boards.

Sincerely, Rosenthal & Associates PC, Harold Rosenthal.

Yes and thanks Harold. I wish more people would renew on their own early. Early renewal saves me money and time which I have so little of these days. As for an article on TurboDos a big YES. I once worked for Teletek and had chance to use TurboDos before I left. Their multiuser systems were the best, unfortunately I can't say same for the management. What made them so great was the single CPU for each user, something the MSDOS folks haven't learned about yet. So hopefully you can do an article and explain what I just said to our readers! Thanks again. Bill Kibler.

Dear Mr. Kibler,

Enclosed please find my renewal for another year.

I run CP/M 2.2 on an Apple//e (unenhanced) (Microsoft Softcard) and also CP/M 3.0 using Digital Research Goldcard. I also use MSDOS Clones at home and at work.

It never ceases to amaze me that each new version of a program requires a doubling of the amount of disk space and a 50% increase in the amount of RAM required for an almost functionally equivalent program. A program that used to require 48K of RAM and fit on a single-sided, double-density disk (not compressed) now comes on 8 highly compressed high density disks and requires 20 megabytes on a hard drive when installed. I realize that memory is relatively cheap now and that CPUs and hard disks are faster, but this increase in space and requirements offends my aesthetics.

Do you know of a source of CP/M software on 5 1/4 inch Apple Softcard formatted disks? All the references to CP/M software seems to be for Kaypro and Radio Shack computers and this cuts out many of the ''old-timers'' who have the Apple - Softcard configuration. Does the software you distribute come in this format?

Thank you for your assistance.

Sincerely, Jim Moore.

Thanks for the renewal Jim. Lambda Software I believe can copy software onto that format for you. In fact David just told me he is becoming the Borland CP/M software distributor. So he may soon have all versions of Borland CP/M software for your system. He also has the Z-Letter which might be a help for you. Give him a call, his number is on the back page.

You also correctly mentioned one of the reasons I am thinking about giving up MSDOS systems and going back to CP/M full time. Of course one doesn't have to get the latest, but I usually find that I hope the newest version will have all the bugs I found fixed. Unfortunately that usually is not the case. In adding all those new features, they miss some of the old bugs and put in many more new ones. Seems you can't win, so why join them. Use CP/M software that works and has all the bugs worked out of it!

Thanks for your interest in TCJ and CP/M! Bill Kibler.

Articles Neded

We need articles on subjects that are of interests to our readers. Those interests now span small and older eight bit systems, through the obsolete IBM PC/XT style of computers.

The subject matter of interest are mostly those which explain and teach readers how to perform intermediate and advanced improvements and modifications to their systems.

All of TCJ's readers are not intermediate in skill, many are beginners. Articles need to take any reader of any skill level through your project, as if they were begining on this subject for the first time.

Areas of current interest are using older and obsolete systems for new embedded control situations.

Embedding operations in ROM and running the entire operations for remote sensing over a telephone line would be a great article of interest to our readers.

First hand reports on the history of early and classic systems is always a topic which our readers enjoy.

Projects which use surplus parts available from current vendors, showing how to debug and develop the needed knowledge of the used system, is something of interest to our readers and advertizers as well.

Short reports on projects that are currently under way, belong in our Support Groups section, where letting others know of what is being done has become a major focus.

Send your letters to:

The Computer Journal
P.O. Box 535
Lincoln, CA 95648-0535

Regular Feature

Classic Support

Group Reviews

SUPPORT GROUPS FOR THE CLASSICS

By JW Weaver

Well here it is, another issue due, and par for the course, I'm in a mad rush to get this article ready for Bill.

On my request for help with information and/or history of computer companies, I've received a response from a fellow in Iowa, with information concerning the Morrow system. Did not know that Morrow produced so many variations. I thank the gentleman for his letter, and hope to include a bit of this information in the next column.

Had some suggestions from Tilmann Reh and Bill Kibler towards changing the direction of this column. Will try to include their suggestions, and if you the reader have any suggestions for improvements or other directions or subjects, you would like to see covered, please drop me a note.

I Would like to include within this column, projects that readers are doing for the classics, information on these projects, if these projects will be available to other readers, or if your project might need some help. Firmly belive that there are readers with projects, un-aware that other readers would like to become involved, might like to duplicate it, or might like to aquire kit / finished product.

With the onset of winter, my outdoor time will be dwindling, that means I will have more time to put in on a few of my own projects. One of these projects, is designing a replacement board for my Kaypro, basicaly a Z181 processor with a SCSI interface, higher speed serial ports, hopefully support for 8" floppy (external of course), 1.2 meg , 1.44 meg as well as the orignal formats. Another project is designing a simple but fuctional computer, built from mostly descrete logic, to aid in teaching my two young daughters, fundamentals of soldering and testing circuit boards, and the concepts of computers, how they work, and how to program.

Well enoght of my rambling, Need to wrap this up and post to BBS for Bill K.

An after note, any persons trying to contact me, via my BBS, in the last 3 weeks of October, I must apologize, when I replaced the crashed hard drive, I also changed the BBS software, and due to my NOT thoroughly wringing out the package, I overlooked a few options which placed the BBS into some strange modes. Sometimes not anwersing incoming calls, sometimes anwersing but not allowing access. Please try again, Hopefully I have corrected the problems.

Keep Hacking.

Contact by US Mail:
TCJ Support Groups
Drawer 180
Volcano, California

Contact by BBS:
(916) 427-9038
up to 2400 baud 8 bits N parity 1 stop

TCJ Staff Contacts

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

Z-System Support:
Jay Sage,1435 Centre St. Newton Centre, MA 02159-2469, (617)965-3552, BBS: (617)965-7259(pw=DDT), MABOS on PC-Pursuit, E-mail: Sage@ll.mit.edu. Also sells Z-System software, see inside front cover.

32Bit Support:
Rick Rodman, BBS:(703)330-9049 (eves), E-mail: rickr@virtech.vti.com.

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

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.

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

Users Groups and Project Reports:
JW Weaver, Drawer 180, Volcano, CA 95689, BBS: (916)427-9038.

Regular Contributors:
Brad Rodriguez,Box 77, McMaster Univ., 1280 Main St. West, Hamilton, ONT, L8S 1C0, Canada, Genie: B.Rodriguez2, E-mail: b.rodriguez2@genie.geis.com.

Frank Sergeant, 809 W. San Antonio St., San Marcos, TX 78666, E-mail: fs07675@academia.swt.edu.

Tilmann Reh, Germany, E-mail: tilmann.reh@hrz.uni-siegen.d400.de. Has complete MS-DOS disk emulation program for CP/M+, contact Jay Sage.

USER GROUPS

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

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

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

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

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

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

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, (503)688-3563. Bi-Monthly user oriented newsletter (20 pages+). Also sells CP/M Boot disks, software, and new versions of Borland CP/M 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, Paul F. Herman Inc., 9317 Amazon Drive, New Port Richey FL 34655, (800)346-2152. Publication and products for Z-100 and S-100 machines.

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.

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, 2751 Reche Cyn Rd. #93, Colton, CA 92324. $9 for 6 bi-monthly newsletters on Amstrad CP/M machines.

Other Support Businesses

Sydex, PO Box 5700, Eugene OR 97405, (503)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.

Davidge Corp. 94 Commerce Dr. PO Box 1869, Buellton CA 93427, (805)688-9598. Z80 support of Davidge and Ampro Z80 Little Board.

Star Technology, 900 Road 170, Carbondale CO, 81623. Epson QX-10 support and repairs. New units also avialble.

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

Peripheral Technology, 1480 Terrell Mill Rd. #870, 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)202-0150. SS-50 6809 boards and systems. Very limited quanity, call for information.

Regular Feature

Kaypro Support

Customization

Mr. Kaypro

By Charles B. Stafford

THE NATURE OF THE BEAST

Wherein we digress from the transmogrification of a mouse into a Lion, to acknowledge the existance of (gasp) the "universal" mother-board and use the dreaded DDT.

MEA CULPAs

When we did the 2 MHz (did I get it right, Lee ?) to 5 MHz speed-up, somehow a digit was dropped when specifying the clock divider chip. It was labeled as a 74LS29 but should have been 74LS293. To those of you who have been sweating blood trying to find one (See it says right here 74LS29), I apologize. You should have better luck finding a 74LS293.

MUSINGS

There is something insidious about summertime, It may be the gentle breezes wafting in from the delta, or it may be the temperature, but invariably toward the end, visions of sailing and surfing push all other thoughts into the far distant background, from whence they return only after the first rains, or the first chilly (below 60 degrees) evening. Sometimes sanity follows, but only on occasion. During one of these dream sequences an 84 Kaypro owner asked,"Can we speed up the 84s like the 83s ?" A great voice sounded through the mists "perhaps."

Several hours later, while investigation of the proposition proceeded, the answer changed to "probably". The entire sequence reminded me that the world of CP/M Kaypros consists not only of 83 K-IIs and K-4s but also of K-10s and those machines using the "universal motherboard", i.e. all the 84s. So far, this series has neglected those late model machines, but NO MORE. Herein we will look at fixing the clock on those that have it installed, and installing it on those that don't.

THE LONG AWAITED TRom decoder board

Sorry, this is only an update. The layout of the printed circuit board is finished and the prototypes ars being etched, I am told. If they are not available in time for the next issue, we will build one from scratch, using a piece of perf board with solder pads on both sides and through-plated holes. It works well, but point to point wiring is never as neat as printed circuits.

"THIS COMPUTER WAS STOLEN FROM ..." Customizing your sign-on message or "here comes the dreaded DDT."

When I started using a UNIX variant, I was introduced to the Vi editor, which is even more convoluted and user-unfriendly than Ed or the infamous Edlin. I did learn to use it, however, because there was nothing else that would do the same job, i.e. imbed executable control and escape characters in executable scripts.

The same is true of DDT, nothing else will quite do the job that DDT will. Having said that, I will admit the existance of NDDT, a variation that does more, but the underlying operating scheme is the same. Here we will just use the D(isplay) and the S(et) functions to alter the CP/M sign-on message.

Here's what you'll need; one bootable diskette, with
MOVCPM.COM
DDT.COM and
SYSGEN.COM, all on the diskette,
another formatted diskette,
and an ASCII table.

For those of you who use the TurboRom, you'll need the analogous programs, and those required by them. MOVTURBO, for instance requires the original MOVCPM.COM and the BIOS file, refer to your TurboRom manual for the exact requirements.

Here's how we do it:

1. Turn on your computer, and watch it boot up, preferably from the diskette, and write down the exact sign-on message that appears on the screen. It will be something like "Kaypro CP/M 2.2d" or "61.00K CP/M - TURBO-BIOS".

2. Use MOVCPM to create an image of the operating system on diskette. To do this, issue the command MOVCPM xx.xx where xx.xx represents a number between 55.00 and 64.00 which will be the resulting size of the TPA. (Transient Program Area, where all those neat programs reside and run) Movcpm will exit with an admonition to save the resulting system by issuing the command: "SAVE 36 CP/Mxxxx.sys", so do it. For those of you running the TurboRom, Movturbo

is the analogous command and it saves the new system all by itself.

3. Now call DDT and the new system into memory with the command: DDT CP/Mxxxx.sys (or DDT Turbxxxx.sys as appropriate). DDT will sign-on with a message and then give you a "-" prompt on the next line. Give the command "d" and DDT will display the first 16 lines of memory beginning at location 100h, the bottom of the TPA.

The display will look something like figure 1, with the addresses matrixed on the left side and the top line, and hexidecimal values in the field. There will also be a column of ascii equivalents on the right side.

```
-d0900
0900  18 FE 80 C2  80 C2 33 00  1A 07 35 36  2E 30 30 4B  ......3...56.00K
0910  20 43 50 2F  4D 20 2D 20  54 55 52 42  4F 2D 42 49  CP/M - TURBO-BI
0920  4F 53 24 20  20 20 20 20  20 20 20 20  20 34 18 B3  OS$          4..
0930  17 2B 19 D2  17 F3 17 D2  00 B1 FF 60  00 00 00 00  .+.......'...
0940  06 32 19 06  32 19 06 32  19 06 32 19  07 02 07 00  .2..2..2..2....
0950  00 05 06 14  06 44 06 13  06 E1 06 05  06 EB 06 01  ....D.......
0960  06 00 08 05  0E 14 0E 44  0E 13 0E E1  0E 05 0E EB  .......D.....
0970  0E 01 0E 00  0D 0A 57 61  72 6D 20 42  6F 6F 74 24  .....WarmBoot$
0980  31 80 C2 21  F8 FF E5 AF  06 08 86 23  10 F8 18 2A  1..!.....#...*
0990  27 11 A0 C2  06 03 1A BE  20 1E 23 13  10 F8 18 2A  '..... .#....*
09A0  50 50 53 1A  07 52 65 71  75 69 72 65  73 20 54 55  PPS..Requires TU
09B0  52 42 4F 20  52 4F 4D 21  21 A3 C2 06  15 4E 23 E5  RBO ROM!!....N#
09C0  C5 CD 45 00  C1 E1 10 F5  F3 76 0E 00  1E 01 CD 0F  ..E.....v.....
09D0  00 11 08 00  19 5E 23 56  D5 DD E1 2A  F0 FF 7C ED  ....^#V...*..\.
09E0  47 ED 5E DD  75 4F 11 D0  DB 73 23 72  3A FB FF 07  G.^.uO...s#R|...
09F0  F5 DD E5 E1  11 4C 00 19  EB F1 F5 3E  0B 30 02 3E  ....L....>.0.>
```

Figure 1

Successive "d"s will display successive 16 line groups of hexidecimal numbers (the contents of memory) with the ascii equivalents on the right side of the screen. Continue to scan through the memory using the "d' command until you see the same message, on the right side in the ascii equivalents column, that you carefully wrote down when the machine booted up.

4. Count over from the left side of the right column to the first character of the sign-on message, and then count over the same number on the address line at the top of the display. Combining the number that you landed on at the top of the display with the line number that the sign-on message started on will give you the starting address of the sign-on message. For instance 07A0 + 05 = 07A5. Write down the address you came up with. Count the number of characters in the sign-on message, and you can include trailing spaces, 20h, and write this number down. This is the time to figure out what you want your new sign-on message to be. You can use the same number of characters as the last number you wrote down. When you have your new message figured out proceed to step 5.

5. Using the S(et) command move to the starting address of the old sign-on message. You can do this by issuing the command "sxxxx" where the xxxx represents the address we calculated two paragraphs ago. For instance, using the address above the command would be s07A5. This will give you a numeric prompt, indicating the memory location and its contents. Using your ASCII table, translate the first character of the new sign-on message to a hex(idecimal) number and enter it with a return. The display will reflect the change and the prompt will change to the next memory location. Continue entering your new message in hex and when you finish enter a period(.). If you made a "misteak" enter the "s" command again with the appropriate address and fix it, entering a period when your finished.

6. To exit DDT, use a ^C (control C) and then issue the command "SAVE 36 NEWCPM.SYS". This writes the first 36 pages of memory beginning at 0100h to disk, in a file called "NEWCPM.SYS". Now all we have to do is install your new CP/M on the boot tracks of a diskette.

7. Issue the command SYSGEN NEWCPM.SYS and the response, after much whirring and clicking will be; " Enter destination drive or return to reboot". Type in the letter of the drive where you have a formatted diskette and press "return", and follow the prompts. I usually write the system to the disk twice, just to make sure.

Those who use a TurboRom would use TURBOGEN instead of SYSGEN, and the commands are otherwise the same. Now put the newly "sysgened" diskette in the "A" drive and reboot to see your new handiwork.

A SMALL CAVEAT

Some times the original system file that you generated with MOVCPM will have what looks like a sign-on message in several places, because several different people wrote parts of the "loader" routine. If you don't use the right place the first time, just go back to the beginning (step 1) and look further. You won't do any permanent damage, because we just modified a copy of the original file, and floppy boot tracks can be rewritten many times.

THE CASE OF THE MISSING HOURGLASS, or in this case real-time clock

All of the so-called "Universal mother boards" are configured for a real-time clock, and a modem. Some even had them installed. For those that still lack them, there is good news and mediocre news. The mediocre news is that the on-board modem was a 300 baud device, and you're better off without it, because now you can use an outboard ("external" for you HCWs (Highly Certified Wizards)) modem in your choice of speeds, at least insofar as your pocketbook will allow. The good news is that it's relatively easy to find AND INSTALL the pieces necessary to make the on-board clock operational.

For those who already have the clock installed, read on, during this adventure we will also track down and destroy the entomological cybernoid that runs around resetting these clocks to random times.

"THE INSTALLATION"

BITS & PIECES

40 pin	socket
24 pin	socket
24 pin	component carrier
Z80A-PIO	parallel input output chip
MM58167A	clock chip
32.768 KHz	crystal
1N4148	diodes (2)
100K ohm	resistor 1/8th watt (1/4 watt will do fine)
1 uF	dipped tantalum capacitor
0.1 uF	disk capacitor
22 pF	disk capacitors (2)
3 volt	battery and holder

INTO THE OPERATING ROOM

A quick note for us neophytes before we get elbow deep in the machine. (You HCWs can skip this part.) The holes at locations Y4, U35, U36, CR6, CR7, C54, C64, and C65 are almost certainly filled with solder, for two reasons, 1. MURPHY, and 2. the mother-board is "wave-soldered" during assembly and everything that can be gets soldered. SO we need to clean out the holes at those locations. You can use either a "solder sucker" as described in a previous project or the "soldering iron and toothpick" method wherein the hole is heated momentarily with the iron and the toothpick is plunged into the hole immediately. This method is tough on toothpicks, so you might want to have several on hand before you start.

Another quick note, this time on the battery. A battery is a battery, right ? Well, maybe, the original installation used a lithium cell about the size of a triple-A cell, and a lot of folks have used both lithium and silver button cells and holders very effectively. The current drain is small, mostly when the computer is off, and there is a recharging circuit on the board for the battery (CR6 & CR7). Because I couldn't find an appropriately sized battery holder within a reasonable time,(the true meaning of reasonable can only be found with a friend at the bottom of a bottle of very good wine), I used a holder from Radio Shack, which holds 2 AA alkaline cells (which should last almost forever) and has a pigtail (2 wires) to which I soldered 2 femaleconnectors (1 each). I also soldered 2 single pin connectors into the board at BT1, and used double stick foam tape to attach the battery holder to the side of the drive cage beneath the motherboard. Next time I remove the motherboard, I'll just have to unplug the "battery cables".

HERE WE GO

The Z80A-PIO goes in at U35. If there isn't already a socket there, (there wasn't on my motherboard) I'd solder in a 40 pin socket first. The two diodes go in at CR6 and CR7, the polarity is marked on the motherboard, just match up the arrows on the diodes with the symbol silk-screened at the location. The crystal should be CAREFULLY soldered in at Y4, and the 0.1 uF disk capacitor is installed at C54. The remaining 2 disk capacitors are for C64 and C65, and the battery is put at BT1.

The clock chip goes in at U36. I also put a socket there and then plugged in the chip.

TAKE NOTE

This is where you can cure or prevent the random resets. Pin 23 of the clock chip is the Power-on signal and was originally tied high directly. There has been some speculation that the high transient appearing here on power-up has been acting as a "reset signal" indicating that some buffering is needed. That notion is further reinforced by the experiences of others who have retrofitted clocks. Borrowing from power supply theory, an inline resistor (analogous to a choke) and a capacitor to ground (analagous to a filter capacitor) would seem to be called for. A 100k ohm resistor and a 1 uF dipped tantalum capacitor appear to be appropriate, and in fact do the job very well. Advent Products had another solution, involving a voltage regulator circuit using a transistor and a zener diode, but this is far more cost effective.

The "universal" mother-board is not really universal, it would appear. The K-2x has an 81-295 and the 84 K-4 and K-4x have an 81 184/5 depending on whether you believe the schematics or the sticker on the mother-board. The difference is that the 81-295 has pin 23 tied directly high and the 81-184/5 does it through a 1 k ohm resistor (R53) with a 47 uF capacitor to ground (C87). It further appears that on the 81-184/5 the resistor (R53) is bypassed by a trace. In either case, the following assembly will work.

Find the 24 pin component carrier and solder one end of the 100k ohm resistor to the saddle above pin 23, with the resistor itself lying on the carrier between the rows of saddles, slip a small piece of insulation on the other lead and dress it alongside the resistor so that it returns to the pin 23 location. In a similar manner, lay the 1 uF capacitor between the rows of saddles, so that its leads can be run to the saddles above pins 23 and 12, with the negative lead soldered at pin 12 which is ground. Now place the MM58167A IC on top of the carrier and solder all legs to the carrier EXCEPT pin 23. Bend out pin 23 and solder the remaining resistor and capacitor leads to it.

In the past JDR Microdevices in Los Gatos CA has had all the necessary parts for the clock retrofit into "universal" motherboard.

PREVIEWS

Next time we meet we will build a Personality-Decoder board from the ground up, and the following time we'll address the '84 video ideosyncracies. Unfortunately, I am not on Compuserve yet, nor am I reachable on Internet, so "snail-mail" or landline will have to do. I can be reached through *TCJ* or at 4000 Norris Avenue, Sacramento, CA 95821.

The Z-System Corner

By Jay Sage

Regular Feature

ZCPR Support

Failsafe Scripts Part 2

Techniques for Running Unattended

Part 2:
The PMATE Analysis Macros

In my previous column I presented the control scripts I developed to allow my DOS computer to run a sequence of tasks unattended over a long period of time, during which the computer might have to be rebooted. As I mentioned, I used that approach to carry out a large set of complex electronic circuit simulations while I was on vacation for nearly a month. Although several problems did, indeed, arise and require rebooting of the computer, the computer was able to pick up quite nicely each time where it had left off, and a large volume of useful results awaited me on my return.

What I described last time was the additions to the AUTOEXEC.BAT file that allow a general set of commands to be run in a fail-safe mode. This time I will describe the PMATE (or ZMATE) editor macros that made it possible for my specific circuit analysis tasks to be performed with some "machine intelligence" (that seems to be a newer term for what used to be called artificial intelligence). Specifically, a set of control programs analyzed the results of the simulations and adjusted a circuit parameter automatically to determine the margins, that is, the range over which the circuit would function properly. Next time, in the third installment, I will describe the glue scripts that held the whole operation together.

The Extension to DESQview

Before getting into the main subject matter, I would like to make one addition to the discussion in the previous column. I am a great fan of DESQview, a DOS multitasker that is in the spirit of Bridger Mitchell's BackGrounder-ii under CP/M. It is, naturally, much more sophisticated and elaborate, but I use it mainly the way I use BGii, to switch between tasks. It is handy to operate my fail-safe system under DESQview so that someone can pop into another window to examine logs of system activity or the results-to-date.

To boot up in DESQview, I add the command "DV" to my AUTOEXEC.BAT file. Then I need a way to get DESQview to operate in a fail-safe start-up mode. To accomplish that, I define a special DESQview keystroke macro -- the one that DESQview runs when it first loads -- as follows:

```
{Learn ! "!DV Startup"}
{Enter}
fs
{Finish}
```

This is the text form of the definition. DESQview has a utility to convert between the text form and an executable form. Even if you don't know DESQview, you can probably guess that this launches the DESQview task identified by the letter pair FS (all DESQview tasks have two-character IDs). The FS task is set up to run the command FAILDV.BTM, the dummy version of which is shown in Listing 1. It is basically the same as the dummy FAILSAFE.BTM that we described last time.

Text Analysis and Automation Using PMATE

We now turn to the main subject of this column: the use of PMATE (or ZMATE) to carry out an analysis of text that can be used to control an automated process. The complex set of commands that carries out the simulation of my electronic circuit will be described next time. For now I will show only the core set of commands where the simulation and the analysis of the results take place. The following are the two essential commands:

```
c:\pspice\pspice1.exe alexsr1.cir alexsr1.out
edit $ b9e xi alexsr1.mat $ .9
```

The first command invokes the PSPICE circuit simulator on the circuit defined by the file ALEXSR1.CIR and writes the output to the file ALEXSR1.OUT. The second command invokes PMATE, which I have renamed to EDIT, with a command line that runs a MATE macro.

The macro command passed in the command tail first switches to edit buffer 9 (b9e), then reads in the file ALEXSR1.MAT (xi), and finally executes the contents of ALEXSR1.MAT as a macro (.9). Obviously, we have to look at ALEXSR1.MAT to see what really goes on. It is shown in Listing 2. Let's start with an overview of what the macro does and why.

This macro will examine the circuit simulator's output listing, which includes a table of voltages at various points in the circuit at various times, and determine whether the circuit functioned properly. But how can PMATE then convey this information to the world outside PMATE, namely to the batch file that invoked PMATE and has to decide what to do next?

If PMATE could write to environment variables, that would be a very good way, but PMATE cannot do that. Under Z-System we might use ZMATE's ability to poke values into absolute memory addresses and then use ARUNZ's ability to paste the contents of memory into a command script to get the result. Fiddling with memory is not such an easy or safe thing to do under MS-DOS, because one can never be sure of the absolute address at which a program will be loaded. For that matter, it's probably not such a smart thing to do in CP/M either.

A really sophisticated method would be to have PMATE compose the text of a BTM file and write it out to disk. The master script that invoked PMATE could follow that command with the invocation of the script to be created by PMATE. I have done things like that on occasion, but I chose a simpler approach here. I just made PMATE create a file with a particular name to serve as a flag or semaphore. One file indicates that the circuit functioned properly, another indicates that the circuit failed, and a third signals some kind of error in the analysis. The controlling BTM script need only test for the existence of these files.

Now let' turn to the details of the macro. The first thing it does is to delete any of these semaphore files that might exist from before. It first tests for the existence of the file using the MATE command @Ffilename$, which returns a Boolean value of TRUE (-1) if the file exists and FALSE (0) if not. If the test is TRUE, then the file is deleted using the macro XXfilename$.

The file ALEXSR1.MAT does not contain all the information or commands needed to perform the entire task, so it proceeds to load additional files into other editing buffers. The first three will only be read; the last one will be edited.

Into buffer 7, the master macro loads the file ALEXSR1.TPL, which contains a template that indicates the desired circuit response. More about this later. A subroutine macro, READNUM.MAT, is read into buffer 8. This is a macro that I use in many other analyses, and so it was convenient to store it in a separate file rather than incorporate it into the master macro directly. The next step is to read the simulation results (ALEXSR1.OUT) into buffer 1.

The last step is to open the file ALEXSR1.RES, the cumulative results files, for editing in buffer T. Since this file grows larger and larger as time goes on, it is read in last. That allows PMATE to use it automatic disk buffering to scroll through the file to the end with the UZ macro without risking an out-of-memory error condition. If it were loaded earlier, PMATE might run out of memory when trying to load the other files. That's just what happened when I was away in Germany. My colleagues described the screen displayed by the hung computer, and I was able to diagnose the problem. I had them copy the large ALEXSR1.RES to a floppy diskette and then delete it from the hard disk, allowing a new one to form. As I recall, they had to do this one additional time before I returned.

Now the analysis can begin. To understand what is happening, you need to know what the output from PSPICE looks like. Listing 3 shows a condensed version. The beginning of the file has a complete listing of the circuit definition. Only after that come the results of the computation.

The analysis begins in buffer 1, with the cursor at the end of the file. We search backwards for the string "0.000E+00". From experience I know that this exact value of zero occurs only as the value of time in the initial time step. (If that could not be relied on, I could have searched back for the second occurrence of "TIME".) If the search fails, there is something wrong with the data file. We then go to buffer T and add an error message to the result file. We then create a one-line semaphore file, ALEXSR1.UNK, to signal an unknown result and exit from PMATE.

If the search succeeded, we loop through the output data several times, each time looking row-by-row at a particular column of data. The circuit is a digital shift register, and the data table shows the voltages at three stages of the shift register after each clock cycle. We want to determine for each voltage whether it represents the high digital state, the low digital state, or an intermediate, invalid state. The circuit can fail either by having a node that does not settle into a valid digital state or by having a node end up in the wrong digital state.

Variable 2 is used to store the column number where the character 'E' appears in the data for a particular node. The loop is traversed for column values of 20, 32, and 44. If an 'E' is not found, then we know we have reached the end of the table, and we skip out of the loop to pick up the next column value.

If the 'E' is present, we back up to the beginning of the number and execute the subroutine macro (READNUM.MAT) in buffer 8. It loads variable 0 with 1000 times the value of the number in the data table. For my circuit, a value under 800 (0.8 V) constitutes the low state, and a value over 1600 (1.6 V) constitutes the high state. Depending on the outcome of the comparisons, either an 'L', and 'H', or an 'I' is added to the line in the result file in buffer T.

After each column has been scanned, a carriage return is added to the result file so that the results for the next column of data (shift register node) appear on the next line of the result file. A section of the result file is shown in Listing 4. When the three loops through the macro have been finished, the result file will end with lines like the following:

```
L H L L H H H H L L L L
L L H L L H H H H L L L
L L L H L L H H H H L L
```

The top line shows the sequence of output states of the first stage of the shift register. The second line does the same for the second stage, and so on. You can see that the data pattern propagates through the shift register.

The next step is to see if this pattern agrees with the intended pattern, which we loaded into buffer 7 from the file ALEXSR1.TPL, which I had created manually. The macro command @h^A@7$ (where ^A represents control-A) compares the text at the cursor (in buffer T) to the text represented by the indirect string "^A@7", i.e., the contents of buffer 7. The result of the comparison is stored in variable 0, which is then tested to determine which message line will be added at the end of the result file and which semaphore file -- ALEXSR1.GD or ALEXSR1.BAD -- will be written out. The edit buffer is then closed (macro XE$), and the edit session terminated (macro XH).

Having shown you last time the general structure of the failsafe setup and this time the innermost structure where the evaluation of the results of a single simulation is made, next time I will show you how these two pieces are connected to carry out the complete, complex computational task.

Listing 1. The dummy failsafe script for DESQview.

```
@echo off

REM   This is the initial FAILSAFE program that will be run
REM   when DESQview is first booted.

text

This is FAILSAFE for DesqView! Right now it is doing
nothing. Add code to C:\DV\FAILSAFE.BTM if you want
something to be performed on a coldboot into DesqView.
endtext
```

Listing 2. The MATE macro contained in the file ALEXSR1.MAT.

```
; ALEXSR1.MAT

; This macro analyzes the data in ALEXSR1.OUT and
; generates a table of the output states for each of the
; three latches. The results are compared to a template
; supplied by the file  ALEXSR1.TPL.  A flag file is written
; out to indicate the result of the analysis.  These files are:
;    ALEXSR1.GD    results were good (matched template)
;    ALEXSR1.BAD   results were bad (did not match
;                  template)
;    ALEXSR1.UNK       results unknown; analysis failed

; clear any exiting flag files

@falexsr1.gd${xxalexsr1.gd$}        ; IF alexsr1.gd exists,
@falexsr1.bad${xxalexsr1.bad$}      ; IF alexsr1.bad exists,
@falexsr1.unk${xxalexsr1.unk$}      ; IF alexsr1.unk exists,
                                    ; if exists... erase it

; load support files into buffers

b7e                  ; goto buffer 7
xk                   ; clear it
xi alexsr1.tpl $     ; load template file

b8e                  ; goto buffer 8
xk                   ; clear it
xi readnum.mat $   ; load the macro for reading numbers

b1e                  ; work on data in buffer 1
xk                   ; clear it
xi alexsr1.out $     ; read in the data file

bte                  ; goto buffer T
xk                   ; clear it
xf alexsr1.res $     ; edit the result file
uz                   ; make sure we start at the end

b1e                  ; back to buffer 1 for analysis
e                    ; turn off error trapping
-s0.000E+00$         ; find starting time value
@e{                  ; IF error
bte                  ;   goto buffer T
i *** bad data in ALEXSR1.OUT *** $
13i                  ;   write error message
1xo alexsr1.unk $    ;   write flag file
xe$xh                ;   close the edit file and quit
%                    ;   end macro (not really needed)
}                    ; END IF

t                    ; remember the location

20v2                 ; process first column of data

:L                   ; mark entry point for looping

[                    ; REPEAT
l                    ;   goto next line
@2qx                 ;   goto exponent of value to read
(@t="E)'{_}          ;   if character is not "E", exit loop
-5m                  ;   back up to beginning of number
.8                   ;   read the value into variable 0
bte                  ;   goto buffer T
@0<800{              ;   IF result is low state
i  L$                ;     write "  L"
}{                   ;   ELSE
@0>1800{             ;     IF result is high state
i  H$                ;       write "  H"
}{                   ;     ELSE
i  I$                ;       write "  I"
}                    ;     END IF
}                    ;   END IF
b1e                  ;   return to buffer 1
]                    ; END REPEAT
bte                  ; goto buffer T
13i                  ; add carriage return
b1e                  ; come back to buffer 1
#                    ; return to starting point
@2+12v2              ; calculate column for next data
@2<45{jl}            ; if less than 45, loop back
                     ; above is { j l }
bte                  ; goto buffer T
-3l                ; go back to beginning of output table
@h^A@7$v0          ; compare to template, put result in v0
z                    ; back to bottom of file
13i                  ; insert blank line
i  Results are $     ; insert header text
@0=0{                ; IF results matched template
iCORRECT$            ;   insert "CORRECT"
1xo alexsr1.gd $     ;   write out "good" flag file
}{                   ; ELSE
iWRONG$              ;   insert "WRONG"
1xo alexsr1.bad $    ;   write out "bad" flag file
}                    ; ENDIF
13i                  ; finish the line
xe$                  ; close the edit file
xh                   ; end session
```

Listing 3. A condensed listing of the output file from the PSPICE circuit simulation program.

```
**** 07/22/92 13:22:58 ******** PSpice 4.05 ********

{ Listing of circuit definition deleted }

{ Listing of model parameters deleted }

TIME       V(OUT1)    V(OUT2)    V(OUT3)

0.000E+00  3.059E-01  3.059E-01  3.059E-01
2.000E-10  5.367E-01  5.367E-01  5.367E-01
4.000E-10  5.367E-01  5.367E-01  5.367E-01
6.000E-10  5.367E-01  5.367E-01  5.367E-01
8.000E-10  5.367E-01  5.367E-01  5.367E-01
1.000E-09  5.367E-01  5.367E-01  5.367E-01
1.200E-09  5.367E-01  5.367E-01  5.367E-01
1.400E-09  5.367E-01  5.367E-01  5.367E-01
1.600E-09  5.367E-01  5.367E-01  5.367E-01
1.800E-09  5.367E-01  5.367E-01  5.367E-01
2.000E-09  5.367E-01  5.367E-01  5.367E-01
2.200E-09  5.367E-01  5.367E-01  5.367E-01
2.400E-09  5.367E-01  5.367E-01  5.367E-01

JOB CONCLUDED

TOTAL JOB TIME        1674.95
```

Listing 4. A section of the result file ALEXSR1.RES.

```
07-09-92:  Timer 1 on: 10:42:04

rftime   = 50ps
ratio    = 0.90
couple   = 0.22
ShotArea = 4.00
Vclk1    = 2.45
Vclk0    = 0.00
cross    = 1.00

L H L L H H H H L L L L
L L H L L H H H H L L L
L L L H L L H H H H L L

Results are CORRECT

Timer 1 off: 11:47:29 elapsed: 1:05:24.70

-----------------------------------------

07-09-92:  Timer 1 on: 11:47:36

rftime   = 50ps
ratio    = 0.90
couple   = 0.22
ShotArea = 4.00
Vclk1    = 2.50
Vclk0    = 0.00
cross    = 1.00

L H L L H H H H L L L L
L L H L L H H H H L L L
L L L H L L H H H H L L

Results are CORRECT

Timer 1 off: 12:52:04 elapsed: 1:04:27.91
```

SCSI BACK ISSUES

Here is a list of Back Issues that pertain to SCSI.

#22 - SCSI Introductory Column.
#23 - SCSI troduction to interface.
#24 - SCSI command Protocol.
#25 - Building a SCSI adapter.
#26 - Software for SCSI adapter.
#28 - SCSI for Real Time Control.
#31 - SCSI for General I/O.
#33 - SCSI for the S-100 Bus.
#48 - SCSI/Bernouli Drive for CP/M.

Issue 20 to 25 available in Volume 3, Otherwise all issues available separately. See pages 52/53 for list of back issues and how to order them.

Regular Feature

68xx/68xxx Support

6800/09 Early Systems

Small System Support

By Ronald W. Anderson

History

This is going to be a nostalgia trip. We are going to talk about SouthWest Technical Products Co. and GIMIX primarily.

By way of introduction, I started working with microprocessors and their applications in 1977. I am an electrical engineer by education, and I soon saw the utility of computers in making my job easier. (I must admit that when I saw the first press release in a trade journal describing a microprocessor, my reaction was "What would I do with something like that?"). In December of 1976 I purchased my first computer, a single board KIM-1 that used a 6502 processor. After learning a bit about instruction sets and real machine language programming (i.e. calculate the binary or hex instruction and enter it into the computer memory that way), I decided to go a bit further into computers, so I ordered a 6800 based computer from SouthWest Technical Products Co. (I'll call them SWTPc as nearly everyone did for quite a few years). I bought the basic computer kit with processor board and extra memory so I could have 16K and could run "8K BASIC" with some room to spare for programs and data. My order was placed in February 1977.

The order must have hit SWTPc about the time a run started. An article had just appeared in (I think) Popular Electronics about their machine, and they must have been swamped with orders. The first thing that arrived was a box containing the BASIC and some other software, and the manuals for those. It was June when the final parts arrived. I had ordered a cassette tape interface, a printer, and a "TV Terminal" which consisted of a keyboard and a board with the serial terminal logic and a signal converter so the terminal could use a standard TV set as a monitor (It had 40 character lines, since at standard TV resolution 80 characters would not be readable. All these were in kit form.

Before the computer kits arrived, I spent many evenings studying the BASIC programming manual. I had managed to find a Tiny BASIC for the KIM-1, and to expand it's memory so I could begin to play with BASIC programs so by the time the SWTPc hardware arrived I could at least write a simple BASIC program and make it work.

I assembled the kits and had no problem with the computer itself, but I managed to put a solder bridge on the cassette interface circuit board, which took several hours to track down and repair.

At work, we had seen the potential of microcomputers applied to our products and had contacted Intel and Motorola. The Intel folks told us that we would have to spend a lot of money to get a development system going. The Motorola folks said "We'll be right over". Consequently we went with Motorola processors, which pleased me because I had a nearly compatible computer at home. We bought the Motorola "Exorciser" development system with floppy disk drives and software including a klutzy line editor and an assembler.

As microcomputer technology grew, I eventually bought myself a pair of 5.25" floppy disk drives and the SWTPc interface board for my home system, plus their simple operating system at the time. I used it for such a short time that I had forgotten what it was called. I just ran across a reference to it. It was FDOS (I assume for Floppy Disk Operating System). It assigned disk sectors sequentially and space wasn't recovered when you deleted a file so you had to run a utility called PACK to move all the valid files to the beginning of the disk to make room for further ones. (UCSD Pascal adopted a similar disk operating system that used a PACK utility).

The hardware disk interface didn't work right off, so I sent it back to SWTPc for a fix. It was returned promptly and in good working order. A few months later the mailman nicely folded a 9 by 12 envelope in half and stuffed it in the mailbox. It was a manual and floppy disk containing Miniflex from SWTPc. I managed to unfold the floppy enough so I could read it, and I had Miniflex installed in no time.

I remember the first assembler and my fears about using it, which were quashed in a day or so after I started to experiment with it. The TSC editor was a "line editor" in which you entered a line number and that line was printed on the terminal. By various commands you could edit a line. C/this/that/ for example changed the first occurrence in the line of the word "this" to the word "that". C/about// would delete the first occurrence of the word "about" etc. The slashes were string delimiters and others would work as well. If you wanted to edit a slash, for example, you could use dashes for delimiters.

When the 6809 processor became available SWTPc sold a new processor board with a 6809 on it. I bought one and modified my motherboard to decode addresses more tightly. Since I had so much old 6800 software and so little for the 6809 I installed a switch that would allow me to use either processor board. There were some differences in the address decoding and the switch could accomodate either. With that upgrade came FLEX9 for the 6809. Flex was a product of a company called Technical Systems Consultants, then in Lafayette Indiana (Purdue country). They had sold a number of computer games written in assembler and distributed on cassette tape for the 6800, and then branched into a line editor, an assembler, and a number of other products. Now as I think about it, they sold some simple games for the KIM-1 6502 board as well on cassette tape or in source and object code listing form. Of course computer games in those days were things like "Hangman". The most complex game they sold was "Battleship", just like the old board game in which your opponent tries to sink your ships secretly marked on a grid, by bombing various points on the grid.

TSC was the first major supplier of software for the SWTPc systems, though SWTPc dabbled in software on and off as well. TSC released two versions of BASIC to replace the original S L O W BASIC that used BCD arithmetic. The first release had 6 digit arithmetic and was FAST. The later one had 15 digit arithmetic and was slower but still faster than the old nine digit BCD version. BCD means "Binary Coded Decimal". Most of the 8 bit processors had some facilities for decimal arithmetic in which a byte was divided into two 4 bit "nibbles", each one representing the numbers 0 through 9. Since in binary each 4 bits can represent numbers 0 through 15, BCD is a less efficient representation of a number, but more importantly, binary arithmetic is more natural on a system in which numbers are represented in binary. More natural means more efficient. Binary arithmetic runs much faster than binary coded decimal.

I began to write a column for '68' Micro Journal after dabbling in publishing a newsletter that I advertised by letters to the editor of Kilobaud and a couple of other computer publications of the time. My first column appeared in the June 1979 issue. I seemed to have a knack for finding bugs in new software. TSC's BASIC wouldn't properly REWIND a data file, for example, and their new FLEX version of the assembler had a bug that showed up when I typed in their first 5 line example program.

Those first floppy drives were pretty unspectacular. Each disk was single sided and single density, and the Shugart drives were capable of 35 tracks (40 tracks, 80 tracks, double sided, and double density all came later). One track was taken with boot information and directory, so the disk had 34 user tracks with 10 sectors of 256 bytes each, four of which were used for sector linking. Total bytes per disk were therefore 252 times 10 times 340 = 85,680. Since my computer at the time had 32K of memory (I had expanded it when I could buy a 16K board for around $190), I wondered how I could ever fill a disk that held that much data(!) At the time I was writing a few utilities in Assembler, and most were no more than 1K bytes, four sectors or less, so 85K plus sounded like a lot of room. Early SSSD (Single Sided, Single Density) disks cost nearly $5 each for this single sided single density applicaton. (I now buy them by the hundred in DSDD (Double Sided Double Density) version for about 30 cents each).

Of course I soon wanted a third drive. The best I could do at that time was another 35 track drive for the bargain price of $225. Then came double sided 40 track drives holding around 200K, double density, giving 18 sectors per track, allowing 360K, 80 track drives at 720K. I was still back at the 85K drives when I bought a pair of 8 inch floppy drives that were double sided and double density, and held just under 1 megabyte of data. Wow, 12 smaller disks worth on one large disk. I would surely never need more than my original box of 10 disks (Ha). Incidentally those drives cost $425 each and the controller was around $225. One drive cost more than the original computer kit! (Last week I bought a 170 Megabyte hard drive for $209. I had the IDE controller, but could have bought one for $17)! I also bought an Integral Data Systems printer, a simple 9 pin dot matrix with one draft mode and friction feed for roll paper. It marked page ends (at least some of my software did that) but pages were somewhere between 9.5 inches and 11 inches long due to friction feed variations.

The new 6809 processor boards had what was called a Dynamic Address Translator memory management scheme. The DAT managed four extra address bits, and so could address 1 Megabyte of memory. FLEX couldn't manage more than 64K, however, so along came a bigger version of FLEX called UNIFLEX to handle that. I didn't ever get into Uniflex.

All along a company called GIMIX in Chicago had been selling competitive hardware, undoubtedly higher quality better designed and better tested equipment at substantially higher prices than SWTPc, but unfortunately also incompatible with respect to disk formats. The story was that SWTPc, in order to save one IC package on their disk controller board, had used the density byte to indicate which side, and the side byte to indicate single or double density. It all worked fine if you had a SWTPc system, but if you wanted to swap files with someone with a GIMIX system you had to use a single sided single density disk only. GIMIX FLEX was different too, since the GIMIX ROM monitor software was different and the disk controller hardware had to be different. In all fairness, GIMIX did it per the IBM standards for floppy disks, correctly using the density and side bytes on the disk.

I received a letter from Terry Ritter, one of the designers of the 6809. He was very unhappy for two reasons. The first was that the 6809 didn't have extended memory management built-in. Because of that, each manufacturer handled the extended memory management differently, so software written for one wouldn't work on the other. This fractured the software market so that different versions had to be written for each kind of machine. A company called

Smoke Signal Broadcasting Co. was also selling 6809 machines at the time, and they required yet a different operating system. (I have to say that I never saw one of their computers so I don't know much about them). Anyway, Terry felt that the disparity of extended memory management schemes spelled the death of the 6809. He didn't say it, but I think he felt that was why IBM chose the Intel 8086 with it's built in memory manager.

Terry also felt that the software lagged the introduction of the 6809 by much too long. The combination of the fractured market and the delays were what, in his opinion killed the 6809. Actually there was quite a bit of 6809 software developed for the SWTPc hardware. Looking back, I think most of it was done by individuals out of necessity because they needed it, as a hobby, learning experience, or whatever, the making of money being the lowest priority. The market never really got big enough to be able to support a company that supplied one or two software products. The closest thing to a commercial effort to supply software was TSC with numerous products. At one time or another they sold their standard Assembler, a relocatable macro assembler, Pascal compiler, two BASIC interpreters, a front end processor for BASIC that let you write code that didn't look like BASIC, a line editor, a word processor (reasonably powerful), a super debugger in both 6800 and 6809 versions (for debugging assembler software), a math package, a scientific function package, and undoubtedly a few that have escaped my attention. Oh, yes, eventually they released a version of FLEX that would run on a Motorola Exorciser system, their original "industrial" development system for 6800 software.

Many of the suppliers advertised their products in *'68' Micro Journal* which became the hub of information on hardware and software for FLEX and the 6809. I wrote my column in that publication for ten years plus, and never received payments in the form of money, but because I could write a review of a software package at least somewhat objectively, I became the software reviewer and applications "explainer". I received a copy of every software package that was developed for those systems over an extended period of time. Some of it was downright junk and some was eminently usable. I was always free to express an opinion either way (or I wouldn't have continued writing columns). A few of my reviews were not published, and the package given to someone else with a different outlook than I, and in some cases received a better review. I can't complain about that because I looked at tools from a perspective of whether they would be useful to ME or not. I could never becomeso detached and objective that I could simply praise a software package because it ran as advertised and was relatively bug free. I do remember one C compiler that tested a real number for zero by adding up the bytes of the mantissa and testing the sum for zero. Carry was ignored, and so many combinations tested zero when they really were not. Of course the proper test would have been to OR all the bytes together and I made that suggestion.

A package came along called PIE (Programma Interactive Editor) by a fellow named Tom Crosley at a company called SoftWest. Tom had done a 6502 version for (of course) the Apple (known as Apple PIE). He had done a 6800 version and a 6809 version. PIE was a full screen editor and I liked it a lot. I managed to get the source code so I could customize it for different terminals. It was written in Assembler, not very well modularized. I decided that I would prove to myself that I could write a similar one keeping the good features and adding better ones, in a higher level language. A company called Windrush Microsystems in the UK had produced a compiler called PL9, which was almost Pascal, but allowed some things that Pascal wouldn't. I embarked on a year or two spare time project to write a screen editor called PAT. PAT stands for Program And Text Editor. I thought PATE sounded too much like liver, so I dropped the E. I don't know why I didn't think of calling it TAPE. I still use PAT on my 6809 systems. I wrote a version in C and then in Windrush's PL9 version for the 68000 called plus. The C version ran under OS9 on a 68020 system that used GIMIX hardware. The plus version ran under SK*DOS on a 68000 system supplied by Peripheral Technology. Later I did a C version for IBM, which I am presently using to write this. (Yes, I've switched at least part of the time to IBM clones).

One evening I was busily using a disassembler called DYNAMITE to look at the code in PIE. When my son asked me what I was doing, I said without much thought "I'm disassembling PIE with DYNAMITE". Of course it suddenly occurred to me that such an operation would be pretty messy!

At work we had long ago bought a few SWTPc systems to use for program development, and then eventually built our own. As soon as we started programming our machines in Assembler I started searching for something else. I figured we couldn't write programs fast enough in assembler. I ran across a compiler called STRUBAL+ (STRUctured BAsic Language) by Jack Hemmenway. It worked, but it was terribly inefficient. A four page program would run me out of operating memory on the 6800 system. Lucidata in the UK advertised a Pascal compiler and it worked just fine, and was reasonably efficient though it wasn't very fast because it was a P code compiler. That is, it generated a pseudo code that had to be interpreted at runtime. It was very easy to use, and was bug free. I reviewed it and called it the first non-toy compiler I had seen for the 68XX processors. Windrush PL9 appeared soon and I reviewed it saying that I would use it if only it had floating point arithmetic. A couple of months later I received an update package with floating point arithmetic, and we started using it (and are in fact STILL using it since we still build and ship computers with 6809 processors).

Assuming that some of you readers have old 6809 systems of GIMIX or SWTPc, and realizing that you might be looking long and hard for software, let me list the last known addresses of some of the software suppliers. Perhaps if a few are still in business and enough of you inquire about certain products, they will run off a few copies and manuals and sell them to you. If you write them and find out they are no longer interested in selling copies, you might ask if it is OK to get a copy from someone that has their software. I understand that most of the suppliers have been quite unreasonable, not wanting to supply the software but still maintaining full copyright protection, thus making it impossible to get a copy. Maybe there are a few who now are willing to allow vintage computer users to share vintage software. My best hunch would be that Windrush would approve, and perhaps Lucidata. TSC may not be in business, and some of my last known addresses might be wrong, or the companies out of business.

If you get approvals for public domain release of some of the software the chances are good that I have a copy somewhere. I'll be glad to supply copies to someone wanting to act as a librarian, but manuals will be a big problem. Maybe in these days of quick copy places, someone will want to copy them and distribute software and manuals for cost plus a small profit to cover wear and tear on disk drives etc. I'll donate originals if I can have a set of copies back. All this contingent of course on a duly signed authorization to allow vintage computer users to swap copies.

Perhaps that is enough for a first column. If you care to write me, my full address is:

Ronald W. Anderson
3540 Sturbridge Ct.
Ann Arbor, MI 48105

Please, if you want to communicate, write and don't call. I like to answer letters and questions at my convenience, generally late at night when all is quiet around the house. I don't promise to answer every letter either in column form or personally, but I have in the past generally managed to do so. It will depend a bit on whether I get 5 letters or 100. I type fast and enjoy corresponding with someone as a change of pace from my work use of computers. Please don't ask me to send you software. We must go through the public domain release process first. There is one exception. I'll send source and object code to PAT 6809 version (I'm sorry but there is no 6800 version) along with a complete manual on disk (that can be printed out on any standard dot matrix printer) to anyone who will send me a blank disk. Format will be DSDD 40 track, FLEX (not GIMIX) compatible. If you have a GIMIX system I can send PAT on a couple of SSSD disks and you can copy them to a DSDD on your system. Please, if you want PAT, send blank disks and desired format instructions.

I should mention here that though FLEX is no longer available and last I heard TSC wasn't willing to let it go Public Domain, there is a totally compatible operating system called SK*DOS available from Peter Stark (see below). SK*DOS will run any software package that would run under FLEX.

Addresses:

Windrush MicroSystems Ltd.
Worstead Laboratories
North Walsham, Norfolk NR28 9SA
United Kingdom

PLUS - Pascal like language for 68000 OS9
PL9 - Similar language for 6809 FLEX
MACE - 6809 macro assembler
McCosh "C" - a fairly complete C compiler (pre-ANSI)

Star K Systems
P.O. Box
Mt. Kisco, NY
Peter Stark

SK*DOS operating system for 6809, FLEX compatible
SK*DOS for PT-68K 68000 systems

Lucidata Ltd.
(last known address, 1986)
P.O. Box 128
Cambridge, CB2 5EZ
England
Nigel Bennee

P-6800 Pascal Compiler for 6809 FLEX

Certified Software Corp.
P.O. Box 70,
Randolph, VT 05060
Bob Reimiller

Pascal compiler for OS9 680XX
Pascal compiler for FLEX 6809 (long ago)

Palm Beach Software
12364 CR 223
Oxford, FL 34484-2709
Dan Farnsworth

SPELLB - 6809 spelling checker FLEX
SPELLB - 68000 spelling checker SK*DOS, REXDOS
ASMK - 68000 Assembler SK*DOS, REXDOS
REXDOS - operating system for 68000, FLEX compatible
EDDY - screen editor for 6809, version for 68000
68000 in this context is PT68K-2, -4, -5 from Peripheral Technology, Marietta GA

Talbot Microsystems
1927 Curtis Ave.,
Redondo Beach, CA 90278 (1986 address)
Ray Talbot

Another address from a software package:

Talbot Microsystems
7209 Stella Link, Suite 112
Houston, TX 77025

tFORTH for 6809 FLEX version

Frank Hogg Laboratory (I don't have a current address)
Syracuse, NY
Frank Hogg

FORTH - for FLEX 6809 CoCo
Currently into 680XX hardware and software

Lloyd I/O
19535 NE Glisan
Portland, OR 97230
Frank Hoffman
(may be out of business)

Line of cross assemblers to run under FLEX 6809
and OS/9 68000

Introl Corp.
647 W. Virginia St.
Milwaukee, WI 53204

C compilers for FLEX, OS9, and Uniflex

Mike Randall
32 Upland Road, Kelburn
WELLINGTON 5
NEW ZEALAND

Modula 2 compiler for SK*DOS 68000

If anyone has additions to this list, please send them to me. I had several other compilers and related products from companies or individuals that would now be very hard to locate.

32-Bit Systems

All Readers

Linux & Linking

Real Computing

By Rick Rodman

Linux

I installed Linux on CD-ROM (from Walnut Creek CD-ROM for about $60). The machine I installed it on was a 386 with 16MB of RAM, a SCSI hard drive that I wanted left alone, a SCSI CD-ROM, and an external SCSI 128MB magneto-optical drive. What I hoped to do was install it on the magneto-optical.

Let me say right away, this system is *amazing*. The floppy boot which comes with the CD , automatically detected the Adaptec SCSI board and all of the SCSI drives, and the Western Digital Ethernet card, and configured itself for all of them, *without me telling it anything!* This is a first for the PC world.

Linux comes with TCP/IP, X Window, Ghostscript, and everything else you might need, including full source. It also comes with an 85-page manual, which is not a "DOS for dummies" by any means, but more a collection of pieces of terse text files and E-mail messages. You can install it in three ways: CD dependent, which requires the CD to run, but takes only 2MB of hard disk; binary, which runs from the hard disk, presumably faster, and takes 95MB; and complete, copying everything to the hard disk, which takes 245MB.

At first, I couldn't get Linux to work with the optical drive, either as a single partition or split into four. After that I tried an 80MB Seagate hard drive, and that didn't work either. As it turned out, I got bit by the usual Fundamental Thing They Forgot to Mention (FTTFM).

In this case, the FTTFM was the fact that there are two different types of partitions - Minix and "Extended", which, of course, has nothing to do with the Extended DOS partitions you normally see hanging around disk partition programs. Furthermore, there are also two types of filesystems, Minix and Extended. Minix filesystems only go in Minix partitions, and Extended filesystems only go in Extended partions. Finally, and the killer, although you can create either kind of partition during installation, the installation only lets you run "mkefs", and it appears that this program only works with an Extended partition. I imagine the "e" in the middle was supposed to clue me in.

Once you get it installed, you have to make a boot floppy. The instructions are on page 2 of the manual, but like much of the manual, they're incomplete and don't match the actual version of the software installed. On my system, /dev/fd0 *almost* works, but doesn't. I have to use /dev/fd0H1440 for the floppy.

1. Log in as root (no password required).
2. Format a blank floppy with: fdformat /dev/fd0H1440
3. Mount the system partion, which in my case was the second SCSI drive, PUN 2: mount -t ext /dev/sdb1 /mnt
4. Copy the file vmlinux over the floppy: dd if=/mnt/vmlinux of=/dev/fd0H1440

For those not familiar with Unix, dd is a *wonderful* utility which can copy files, truncate files, translate from EBCDIC to ASCII, and a lot of other really neat things. You can't just copy, as in "cp /mnt/vmlinux /dev/fd0H1440", because the cp command will overwrite the "special file" in the /dev directory which represents the device. (Devices are not actually files in the /dev directory; instead, there are "special files" created by mknod which represent them for common activities.) In the case of character-mode devices (e.g. /dev/tty0), cp seems to work properly, but it doesn't handle block-mode devices such as disks or tape. Dd is more intelligent, but of course is a more complicated program.

Now, once you've booted on the hard drive, you may want to get back to the CD-ROM. The mount command you need (not listed in the manual) is:

```
mount -t iso9660 /dev/scd0 /mnt
```

Once you're over those hurdles, you're up and running, and you can edit with Emacs, compile programs, and anything you might do with Minix. But wait, there's more. Now, you can move on to getting X Window (never "X Windows", remember!) to work. Nearly all VGA and Super VGA boards work very well. For higher resolutions, you might check what "dot clocks" your video board has. I used both ATI and Paradise chip-set Super VGAs with no difficulty. Diamond boards are not supported. There is an experimental driver for the 8514, which I hope to try soon.

Side comment to anyone in the PC world wondering what video board is the best: There are three basic classes for sale today. First, the VGA and Super VGA boards. There are *slight* differences in speed among these. Second, the 8514 and compatibles, such as the ATI Ultra. These are *tremendously* faster than any VGA or Super VGA, but cost more. You're usually limited to a maximum 1024x768 resolution. Third, proprietary, and very expensive, higher resolution boards like Cornerstone, Artist Graphics, Number Nine, Sigma, Matrox, etc. These boards are useful when you want to go 1600x1200 or higher. But remember, you have to get drivers from them.

If the board only comes with a Windows 2.1 driver, you run Windows 2.1 or you run DOS, buddy.

If you try a driver and the screen goes haywire, remember that you can get out "blind" by a Ctrl-Alt-Backspace tridigitation.

The mouse drivers are more of a mess. A Microsoft-compatible mouse connected to a regular serial port works; nothing else I've tried does, though, and the manual's comments on this topic don't match the installed software.

X Window is very nice, although it seems a little sluggish. There is an MPEG viewer which will give your video board a real workout. Full source is on the CD, too.

I haven't yet tried the networking stuff. Supposedly it's TCP/IP. Also, you can apparently mount DOS diskettes and partitions, and run DOS programs from within Linux, but I haven't tried that either.

The version number of the Linux package I got was 0.99.7a, which indicates that the folks don't view it as a finished system yet. In my opinion, the main area that work is needed in is the manual; everything else is very good. If you're a fairly competent Unix user, there's no need to pay big bucks for BSD-386, UnixWare, or Solaris. This package is more complete, lower in price, and includes source.

Some folks are working on porting Linux to the PC-532. I've also heard that a port is underway to the Amiga. I'm sure you'd need an Amiga with an MMU, like the A2000 or A3000. If you've got an A1000 or an A500, you can still run Minix. More interestingly, some Linux folks are working on a package called WINE, which will allegedly allow MS-Windows programs to run under Linux using X-Windows. This more or less resembles the WABI stuff going on at Sun, where they hope to run MS-Windows programs under Motif.

If you're not fairly confident of your Unix thinkology, Minix has a much better manual, and, from what I've heard, Coherent is even smoother to get working. But if you want to jump in with both feet, Linux is an immense package with new things to see at every turn. And you can't really foul it up too bad, because after all, it comes on CD-ROM.

Dynamic linking in Windows and OS/2

Don't turn the page! What I'm going to describe here, and very briefly - I promise! - is the way the DLLs *really work*, which is much simpler than the ponderosities purveyed by PC pontificators would presume to portray.

DLLs really are regular EXE files with a list of function entry points at the beginning. When you load a DLL, it's brought into memory. You can now call the entry points by indexing into the list of entry points. There are two ways to do this: by indexing into the table by number (called an "ordinal") directly, or by looking up the ordinal in a list of names near the beginning of the file.

The SDK, and Windows and OS/2, let you use DLLs sort of like regular code libraries. What happens here is that you actually link with a "stub library" created by reading the name list out of the DLL. LINK then marks your EXE file with a list of DLLs that must be loaded before the program can run. As you can see, there are bad sides to this: not only does your program take a long time to start up, as all of the DLLs you ever use are loaded, but also, if any DLL doesn't exist, your program can't handle it.

It's faster and more powerful to load your DLLs yourself at run-time and call the functions using the ordinals. A neat capability you get this way is selecting the filename of the DLL at run-time, for example, to select different languages or equipment configurations.

When you dynamic-link, nothing is checked - it's all up to you to get your ordinals and parameters correct. If you statically link and use prototypes, you can be fairly safe, as long as the DLL doesn't change - more on this in a minute.

Dynamic linking in AmigaDOS

At the outset let me quickly say that the "DOS" in AmigaDOS doesn't stand for "Dumb Old Stuff" like it does in the PC world. AmigaDOS is a very clever and well-designed multitasking operating system once known as Tripos. It was once portable, but with its destiny chained to the Amiga star, hardware dependencies have slowly crept in.

On the Amiga, there are "libraries" which work very much like Windows DLLs. They are executable files with multiple entry points. Libraries are loaded with a procedure called OpenLibrary, which returns a pointer in register A6. After that, you call the library by offsetting from register A6, like this: JSR _LVOxxx(A6). Rather than using hard-coded constants, usually include files are used to define the offsets. This gives you the convenience of using mnemonic names rather than numbers, but it's still much like the "ordinals" method under Windows.

Like Windows, parameters and offsets are not checked. If you make a mistake, or if the library changes, you're lost in space.

Unlike Windows, which itself consists of three main DLLs (named EXE), AmigaDOS itself consists of a large number of libraries loaded at boot time. It's quite easy to add a new device driver, or replace any of the standard device drivers or libraries. It has its own idiosyncrasies, of course, but is on the whole a very likable OS. It's multitasking, too, and quite stable.

Dynamic linking in SunOS Unix

Under SunOS, the dynamic linking feature is considerably different from Windows and the Amiga. All references are done using names (symbols). When you have linked with the "shared object" library (a ".so" file), a reference is added to your object that the file is needed. At run-time, the ".so" file is loaded before your program begins execution, but the addresses you reference are not resolved until the time of first call. This means that the first time you call a function, the name is resolved in the library. Thus, you can get an "undefined symbol" error in the middle of running a program.

Remember the library-changing problem we discussed in the Amiga and Windows environments? While some programmers are writing programs that call libraries, other programmers are busy making improvements to the libraries. With static linking, the program was combined with a snapshot of the library at the moment that they got along nicely. With dynamic linking, neither the rock nor the hard place are fixed in any relation to each other.

SunOS, though, has an interesting solution to this problem: version numbers, with a major version and a minor version, are stamped on the ".so" file and any executable it is linked with. Then, at run time, the loader (ld.so) looks for a ".so" file with the same major version number and the maximum available minor version number. Library writers must remember to increment the major version number any time changes in functions or their parameters are made. The version numbers are put on the file name. For example, the Xt (X Window Toolkit) library file on my system is "libXt.so.4.10", and the C runtime library is "libc.so.1.6".

Unix folks have the luxury of 31-or-more-character filenames. Under DOS, this same thing would be harder to accomplish. The main drawback of the SunOS approach is the linking of entry points by name at run-time, which is slow; since the library name is determined at link time, again, you can't handle a missing library or specify the library filename at run-time.

For operating system hobbyists, the best scheme - where best means simplest and fastest at runtime - appears to be a variant on the AmigaDOS system; but adding the version number scheme used by SunOS would be easy to do and well worth the extra work.

TCP/IP for small computers

Moving along to another topic: What we really need in our little-machine community is a consistent way to communicate between the machines, share files, and so on. In my Laboratory, I have an Amiga, a DEC Rainbow, four PCs, three S-100 machines, and a Sun, in alphabetic order. Many TCJ readers, including our esteemed Editor, have similar collections. Right now, to transfer files, I have to plug and unplug cables and use a variety of file-transfer programs: Kermit here, Xmodem there, Zmodem there. Wouldn't it be great if we could construct a little network using RS-232 cables, with very simple client software and consistent commands?

Since TCP/IP appears to be the standard and everyone's favorite, we're looking at that first. There are two possibilities for TCP/IP on small computers. Both support use of SLIP (Serial Line IP), which should allow computers with serial ports to interact with other computers on an Ethernet network.

One is the KA9Q package by Phil Karn. This package started out (relatively) small and simple but soon became extremely large and complex. I have two versions, one from 1987, the other from 1991. The 1987 version has 73 source files adding up to 387 kbytes; the 1991 version has 246 source files adding up to 1.7 megabytes. This is not including the drivers.

The second is TNET, an implementation of TCP/IP for Minix by Michael Temari and Fred Van Kempen. This one is not quite as large as the 1991 KA9Q, but still sizable.

Actually, I don't think that these packages are suitable for our network of small computers. They're just too large and complex. But that's because TCP/IP itself is too large and complex.

I am quite fond of ftp's ease of use. Ftp is an application, however, not the protocol it rides on. As an API, I'm comfortable with Netbios, although it could have been cleaner. But again, it actually says nothing about the protocol underneath.

What we might do here is design a very simple underlying protocol, perhaps based on Xmodem or Zmodem, then build an API atop that, and an ftp-like program atop that. The protocol should support forwarding of packets. The program should be usable on single-tasking machines with only a single serial port.

In the computer world a trend has developed toward simplicity, streamlining, low overhead. An example is the low-overhead Frame Relay protocol that is replacing the baroque X.25. And much of the attractiveness of RISC machines comes from this same trend.

Next time

I've also acquired an experimental operating system called "Sprite". This system was designed from the outset as a distributed operating system. If time and space permit, I'll have some impressions of that package.

And from another time and space comes a "fully operational battle station", Windows NT. Will all else go the way of Alderaan? In the meantime, send in your NT puns. (For example, "is the glass half full or half NT".)

Where to call or write

BBS or fax: +1 703 330 9049 (autoswitching)
E-mail: rickr@virtech.vti.com

Walnut Creek CD-ROM +1 800 786 9907 or +1 510 947 5996
1547 Palos Verdes Suite 260
Walnut Creek, CA 94596-2228

CONNECTING IDE DRIVES

by Tilmann Reh

Special Feature

Intermediate Users

Part 3: IDE Commands

In Part II (printed in the previous issue of *TCJ*) we covered the basics of the IDE interface in terms of history, concept, hardware, and register structure. This time we want to dig deeper into the software side of those drives.

Terminology

Using common terminology, I often simply refer to the "drive" when, in fact, I am thinking of the integrated controller of an IDE drive. However, when explicitly talking of an external controller like the WD1010, I always refer to the "controller".

Register Accessing

Let us first recall the Task File. It consists of the data register, a set of six parameter registers, and the command/status register. For those who don't have Part II lying nearby, here is a shortform:

Relative Address	Register	Abbr.
0	Data Register	D
1	Error Reg. / Write Precomp. Reg.	E / WP
2	Sector Count	SC
3	Sector Number	SN
4	Cylinder Low	C
5	Cylinder High	C
6	SDH (Sector Size, Drive, Head)	D,H
7	Status Reg. / Command Reg.	

Also remember that the data register is the only 16-bit register!

Every parameter register of the task file is freely accessible as long as there is no active command. Before loading the command register, all related parameter registers must contain the appropriate values. They may be loaded in any order. After the command register is loaded, the issued command is immediately started. The original WD1010 hard disk controller chip had a flag (bit 1 of the status register) which was set during execution. With IDE drives, the BUSY flag of the status register is simply set until the command execution is completed.

The WD1010 controller chip knew only 6 commands. However, some of the commands have option flags within them. To support additional features, today's drives have many more commands. The following is a list of common commands, options, and needed parameters, with the WD1010 commands marked by an asterisk and the manufacturer-dependent expansions marked with a plus sign:

Command	Type	7 6 5 4 3 2 1 0	Hex	Parameters
Recalibrate	*	0 0 0 1 (Rate)	10-1F	D
Read Sector	*	0 0 1 0 0 M L T	20-27	SC,SN,C,D,H
Write Sector	*	0 0 1 1 0 M L T	30-37	SC,SN,C,D,H
Scan ID / Verify	*	0 1 0 0 0 0 0 T	40,41	D,(SC,SN,C,H)
Write Format	*	0 1 0 1 0 0 0 0	50	C,D,H,(SC,SN)
Seek	*	0 1 1 1 (Rate)	70-7F	C,D,(H)
Exec Diagnostics		1 0 0 1 0 0 0 0	90	D
Set Drive Parameters		1 0 0 1 0 0 0 1	91	SC,(C),D,H
Read Multiple	+	1 1 0 0 0 1 0 0	C4	SC,SN,C,D,H
Write Multiple	+	1 1 0 0 0 1 0 1	C5	SC,SN,C,D,H
Set Multiple	+	1 1 0 0 0 1 1 0	C6	SC,D
Power Commands	+	1 1 1 0 0 x x x	E0-E6	SC,D
Read Sector Buffer		1 1 1 0 0 1 0 0	E4	D
Write Sector Buffer		1 1 1 0 1 0 0 0	E8	D
Identify Drive		1 1 1 0 1 1 0 0	EC	D
Cache On/Off	+	1 1 1 0 1 1 1 1	EF	D,WP
Power Save	+	1 1 1 1 1 x x x	F8-FD	?

Parameters in parentheses are needed with some drives and ignored by others (depending on the manufacturer and age). Any required parameters must be valid before a command is started.

Although most of the commands are manufacturer-dependent, this usually does not raise problems. For normal operation of the drive, only the WD1010's and few of the really common commands are needed. Now let's have a look at the options.

In the Restore and Seek commands, there is a four-bit rate field. This was originally intended to specify the step rate for head movements, with a zero value meaning 35 us per step and all other values representing counts of 0.5 ms per step (so that the range was from 0.5 to 7.5 ms). The hard disk controller had a memory for each drive's step rate, so the same value would be used for implied seeks later. But very soon, even with later ST-506 controller boards, this step-rate field became obsolete (due to handshake mechanisms between controller and drive). With today's IDE drives, the four lower bits of those commands are generally ignored.

The Read and Write commands originally had an option flag (M) for multi-sector (m/s) transfers. Today this flag is nonexistent. For doing m/s transfers, the sector-count register is simply set to the desired number of sectors to be processed.

But today's drives have another flag which originally wasn't there: the "Long" flag (L). When it is set, the ECC (error-correction) data is transferred after the sector's data field. I assume this was meant for error correction when there are too many errors for the drive to correct automatically. However, there is a very odd characteristic: the ECC data is transferred in *bytes* through the data register. This is the only case where the data register doesn't transfer word data! By the way, the number of ECC bytes differs from drive to drive, but can be read out using the Identify Drive command.

All generations of hard disk controllers and IDE drives support the last option flag for Read/Write commands, the Retry Flag (T). Normally, the drive retries to read/write a sector after non-fatal errors. When this option flag is set, automatic retries are disabled. The Scan ID command also supports this option.

The Commands

I will now explain the commands and their parameters in detail. To do this, besides drawing on my own experience in IDE interfacing, I collected detailed information and specifications from three different, independent sources. However, there still might be some slightly different drives or controllers out there. Please inform me if you encounter problems or differences with your disk.

One general feature of both the WD1010 controller and modern IDE drives is implied seeks. That means you don't have to explicitly move the read/write (r/w) heads to the desired cylinder before starting to read from or write to the disk. When the command is issued and the actual cylinder number doesn't match that of the cylinder registers, an implied seek is performed, transparent to the user. This applies to every command where it may be needed (Read, Write, Verify, Format).

Recalibrate (1xh):

This command moves the r/w heads of the selected drive from anywhere to cylinder 0. The controller waits for the drive to complete the seek before the task file is updated and the busy flag is reset. Upon successful completion of the command, the error register and the cylinder registers are set to zero, while SC, SN, and SDH remain unchanged.

Read Sectors (2xh):

This is probably the mostly used command. It will read from 1 to 256 sectors of disk data as specified in the SC register (with an SC value of 0 meaning 256 sectors to be read). The starting sector is defined by SN, C, and H in the task file.

When the task file contains invalid parameters, an error occurs. Otherwise, the r/w heads are moved to the requested cylinder if they are not already there (implied seek). Then the data of the starting sector is read into the sector buffer, and the DRQ (data request) bit in the status register is then set. This informs the host that the sector data can be read from the sector buffer. When this is completed, DRQ is reset and the drive is ready again.

For reading multiple sectors (SC>1), when the sector buffer is completely read and DRQ is reset, the busy flag is set again immediately, and the next sector's data is read into the sector buffer. When DRQ is set again, the next sector's data can be read from the buffer. This is repeated until the SC register value reaches zero (this register is decremented with every successfully read sector).

In any case, after successful completion of this command, the SC register contains a zero value, and the other registers in the task file are updated to contain the cylinder, head, and sector number of the last-read sector. If an error occurs, the task file will contain the parameters of the sector at which the error was detected.

For the Read Sectors command, two options are possible. The "M" option is valid only with the original WD1010 controller (thus with old AT's). When M was not set (0), the SC content was ignored, and exactly one sector was read. When M was set (1), the SC value was taken as the count of sectors to read. Today things are different. With modern IDE drives, if the M bit is set, an error occurs. So this bit must always be cleared!

The other option ("L") is valid only with modern IDE drives. "L" stands for long and means that the additional ECC data, which the drive automatically puts after each sector, is transferred after the net data of each sector. When this option is set, the drive also does not check these ECC bytes, so it won't detect or correct errors. This provides a way to read a sector's (remaining) data even if it is nonrecoverably damaged. When using this option, remember that the ECC data is transferred as bytes through the word-wide data register!

Write Sectors (3xh):

The Write Sectors command is very similar to the Read Sectors command. Of course the data flow direction is different... This command will write up to 256 sectors of data to the disk. All parameters and options are similar to those of the Read Sectors command. After writing the command to the command register, the drive sets the DRQ flag, informing the host that the data can be written into the sector buffer. When all data has been transferred, DRQ is reset, and the drive starts writing the data buffer contents to the disk. The busy flag is set as long as the drive is physically writing to disk. The SC register is decremented, and, if not zero thereafter, DRQ is set again for the next sector. When using this command with the "L" option, the drive will use the ECC bytes delivered by the host

Continued after Centerfold, page 33.

TCJ Center Fold

Special Feature

All Users

XEROX 820

UNLESS OTHERWISE SPECIFIED:
RESISTANCE VALUES ARE IN OHMS, ±5%, .25W

CAPACITANCE VALUES ARE IN MICROFARADS, +80 -20%, 50V

POWER DISTRIBUTION TABLE

REF DESIGNATIONS	GND	+5	+12	-12	-5
U1-8,17-24,37-44 54-61	16	9	8		1
U9-12,14,16,25,26,28 29-34,36,45,49,51,52 53,74,77,78,79,82,83 89,94,95,98,100,102 103,104,108,110,112,114 115-119	7	14			
U13	12	3			
U15,27,48,50,62, 65-73,75,76,80,88,90 91,93,106,107	8	16			
U35,47,81	10	20			
U46	29	11			
U63,64,92	12	24			
U84-87	9	18			
U96	31	9			
U97	11	2			
U98	5	24			
U101,105	11	26			
U109	20	21	40		1
U111,113	7		14	1	

3 REFERENCE DESIGNATIONS

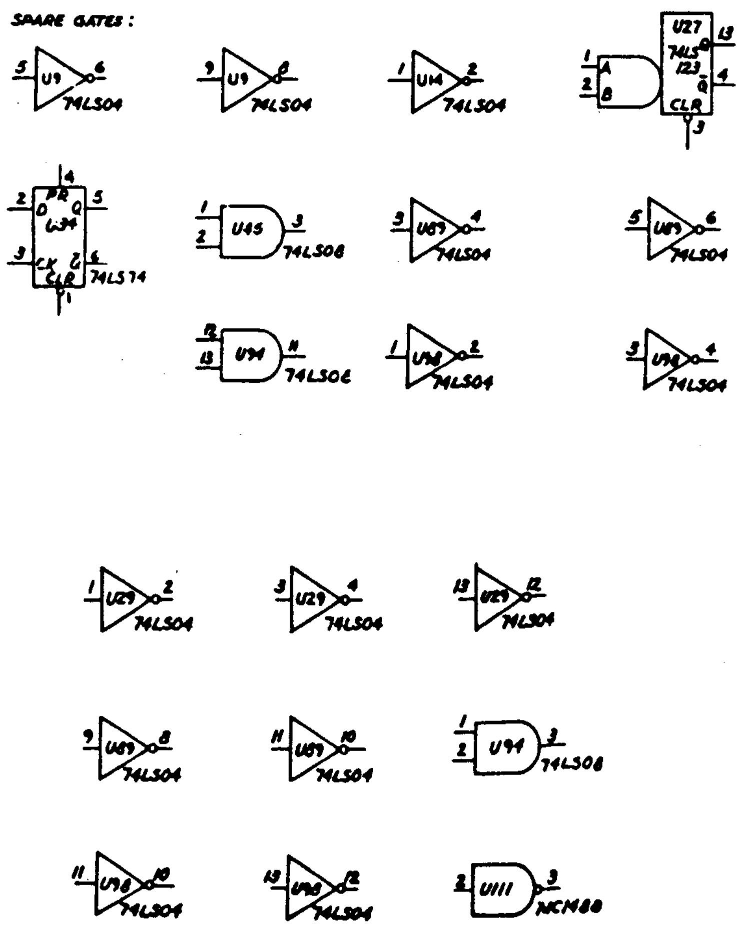
LAST USED	NOT USED
C195	C137
E3	
J11	
R69	R34,49
S1	
U119	U63
Q1	
VR2	

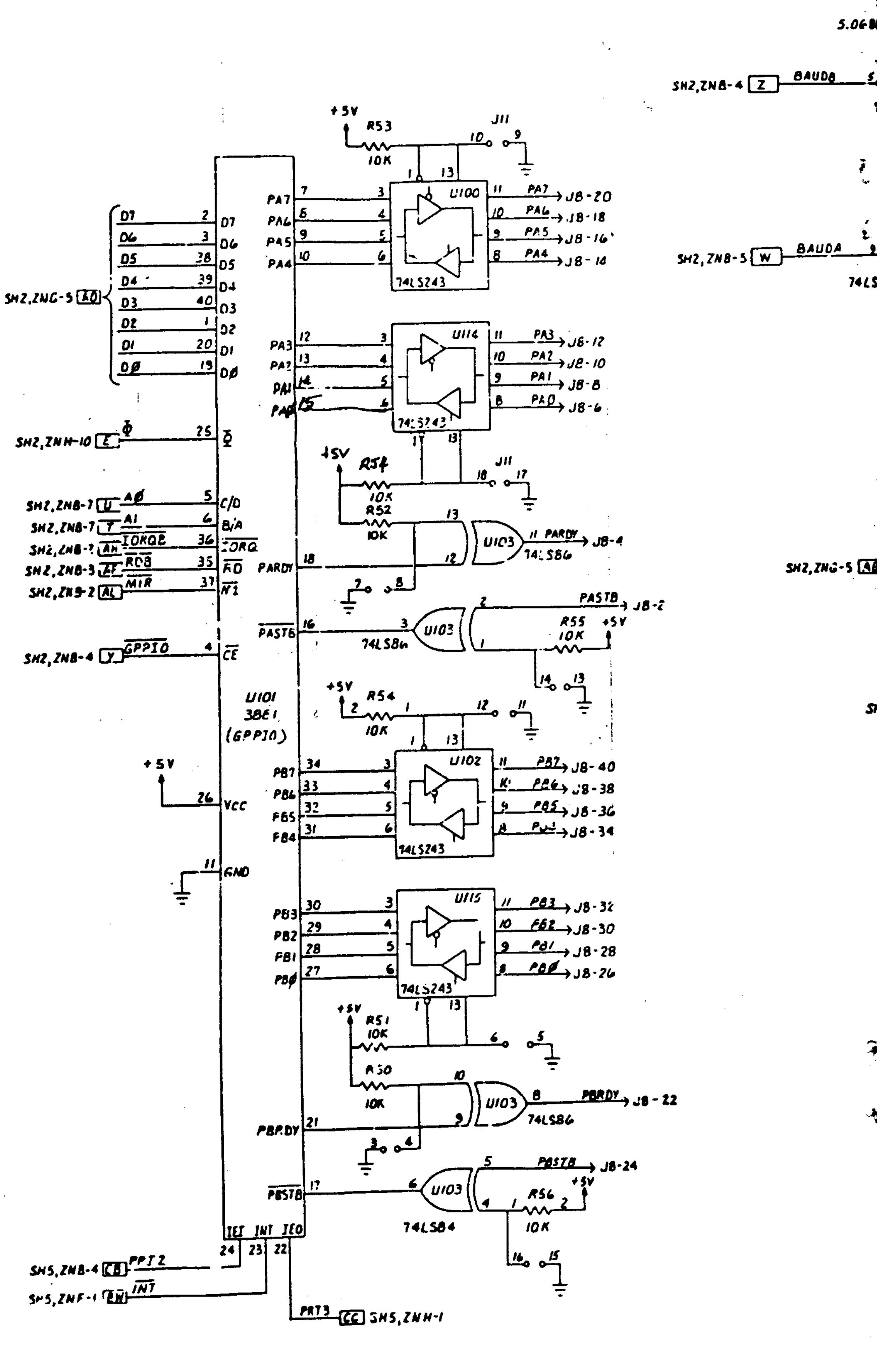
4 FOR NORMAL OPERATION SHUNTS TO BE INSTALLED IN THE FOLLOWING POSITIONS

REF DESIG	BETWEEN PINS
E1	1,2
E2	1,2
J9	7/8,11/12,15/16 19/20,23/24 27/28,31/32 35/36
J10	3/4,7/8

5 LAST INTERCONNECT LETTER USED "CD"

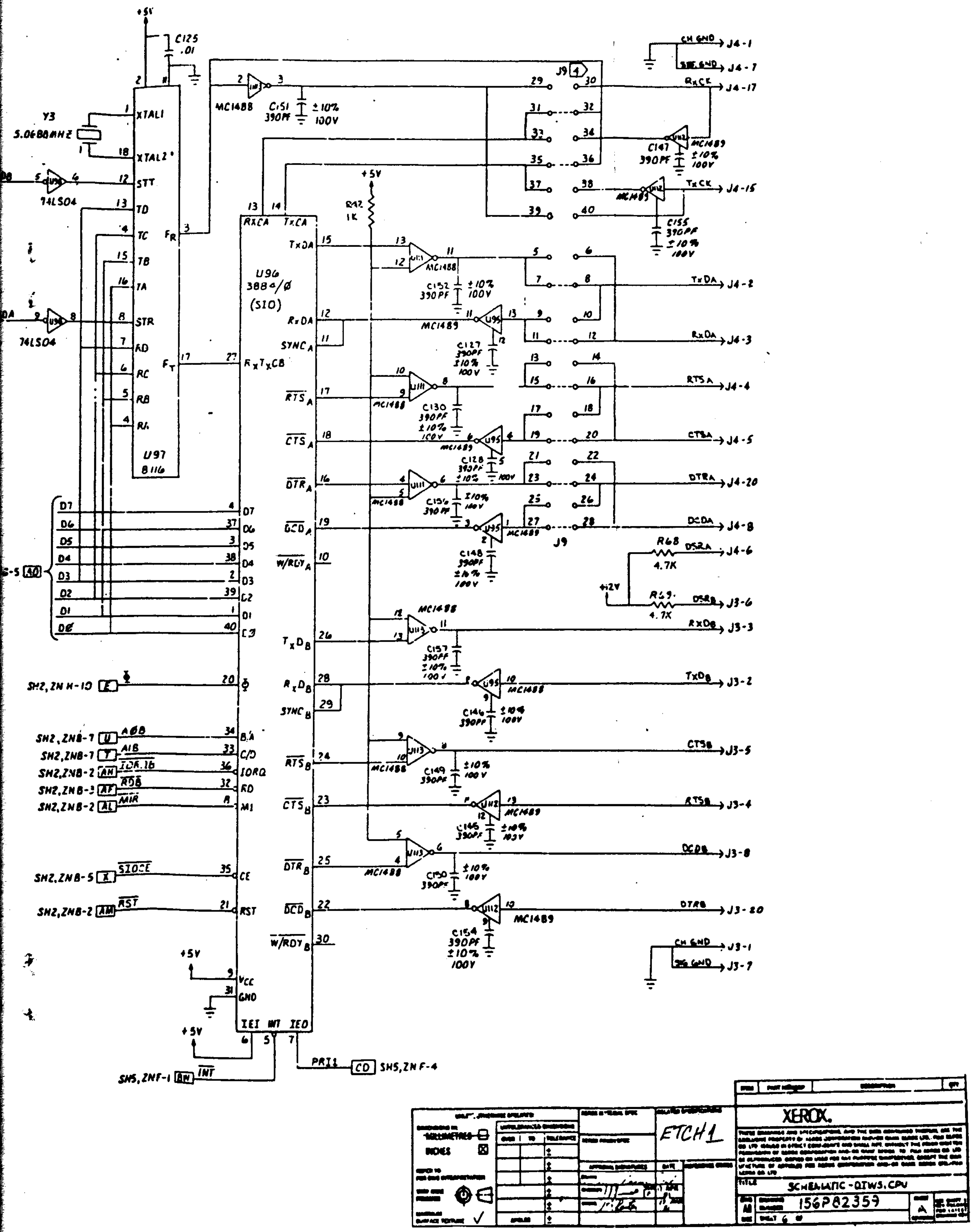
6 SPARE GATES:



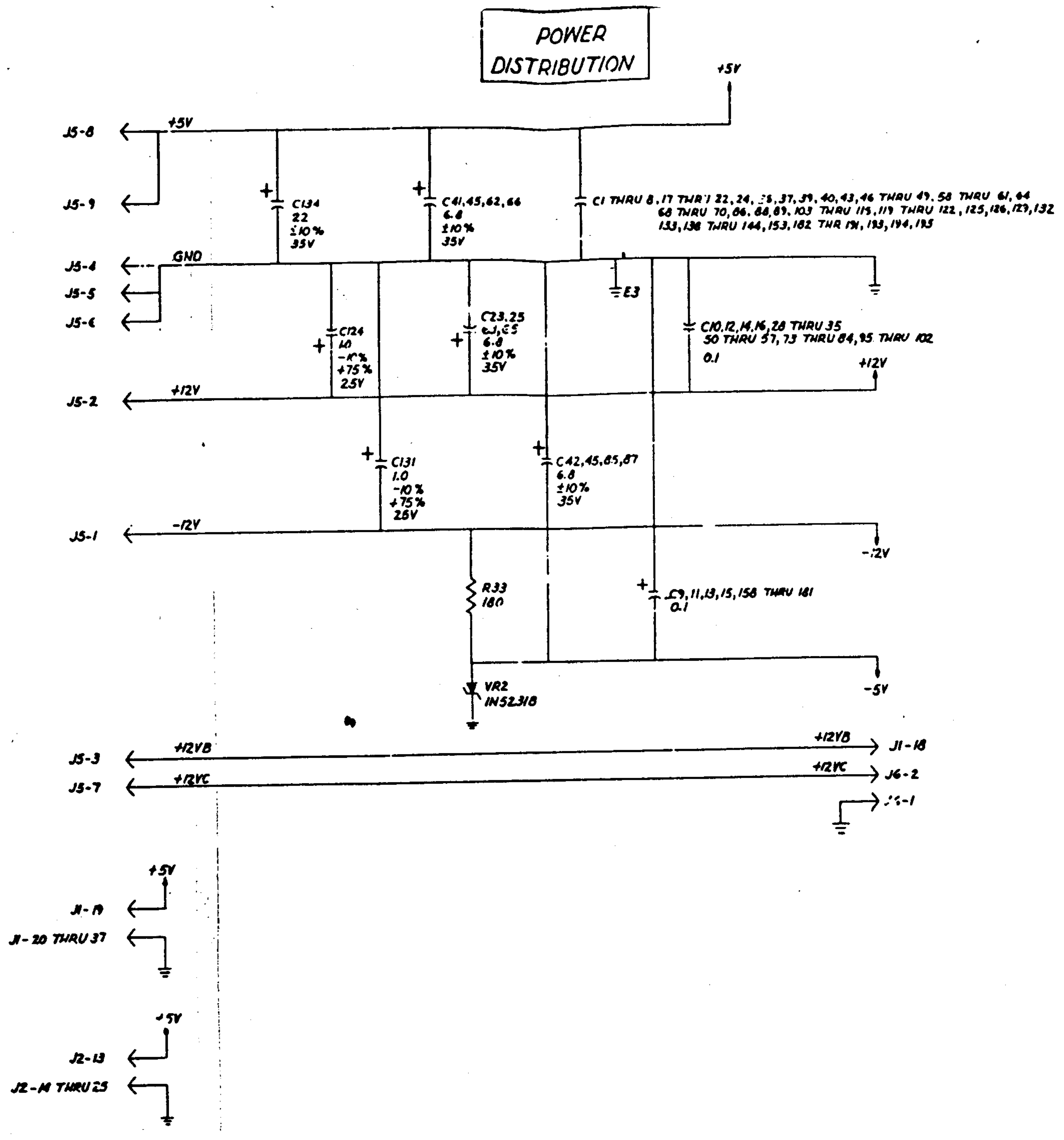


+5V
R53
10K
J11
U100
74LS243
PA7
PA6
PA5
PA4
J8-20
J8-18
J8-16
J8-14
U114
PA3
PA2
PA1
PA0
J8-12
J8-10
J8-8
J8-6
SH2,ZNG-5 AD
D7
D6
D5
D4
D3
D2
D1
D0
SH2,ZNH-10 Z
Φ
SH2,ZNB-7 U A0
SH2,ZNB-7 T A1
SH2,ZNB-? AH IORQ2
SH2,ZNB-3 AF RD8
SH2,ZNB-2 AL MIR
C/D
B/A
IORQ
RD
M1
R54
R52
U103
74LS86
PARDY
J8-4
PASTB
J8-2
R55
SH2,ZNB-4 Y GPPIO
CE
U101
38E1
(GPPIO)
Vcc
GND
U102
PB7
PB6
PB5
PB4
J8-40
J8-38
J8-36
J8-34
U115
PB3
PB2
PB1
PB0
J8-32
J8-30
J8-28
J8-26
R51
R50
PBRDY
J8-22
PBSTB
J8-24
R56
74LS84
IEI
INT
IEO
SH5,ZNB-4 CB PPI2
SH5,ZNF-1 BW INT
PRT3 CC SH5,ZNH-1
5.068
SH2,ZNB-4 Z BAUDB
SH2,ZNB-5 W BAUDA
74LS
SH2,ZNG-5 A

GP
PIO
SIO



C125 .01
Y3 5.0688MHZ
XTAL1
XTAL2
STT
TD
TC
TB
TA
STR
RD
RC
RB
RA
FR
FT
U97 8116
74LS04
MC1488
C151 390PF ±10% 100V
J9
U96 3884/Ø (SIO)
RXCA TXCA
TxDA
RxDA
SYNCA
RTSA
CTSA
DTRA
DCDA
W/RDYA
TxDB
RxDB
SYNCB
RTSB
CTSB
DTRB
DCDB
W/RDYB
RxTxCB
D7 D6 D5 D4 D3 D2 D1 DØ
SH2,ZN H-10
SH2,ZNB-7 AØB
SH2,ZNB-7 A1B
SH2,ZNB-2 IORQ.1B
SH2,ZNB-3 RDB
SH2,ZNB-2 M1R
B/A
C/D
IORQ
RD
M1
SH2,ZNB-5 SIOCE
SH2,ZNB-2 RST
CE
RST
VCC
GND
IEI INT IEO
SH5,ZNF-1 INT
PRI1 SH5,ZNF-4
R42 1K
MC1489
C147 390PF ±10% 100V
C155 390PF ±10% 100V
C152 390PF ±10% 100V
C127 390PF ±10% 100V
C130 390PF ±10% 100V
C128 390PF ±10% 100V
C156 390PF ±10% 100V
C148 390PF ±10% 100V
C157 390PF ±10% 100V
C146 390PF ±10% 100V
C149 390PF ±10% 100V
C145 390PF ±10% 100V
C150 390PF ±10% 100V
C154 390PF ±10% 100V
R68 4.7K
R69 4.7K
+12V
+5V
CH GND J4-1
SIG GND J4-7
RxCK J4-17
TxCK J4-15
TxDA J4-2
RxDA J4-3
RTSA J4-4
CTSA J4-5
DTRA J4-20
DCDA J4-8
DSRA J4-6
DSRB J3-6
RxDB J3-3
TxDB J3-2
CTSB J3-5
RTSB J3-4
DCDB J3-8
DTRB J3-20
CH GND J3-1
SIG GND J3-7
XEROX
ETCH 1
INCHES
SCHEMATIC - DTWS, CPU
156P82359
A
SHEET 6


POWER DISTRIBUTION
+5V
J5-8
+5V
J5-9
C134 22 ±10% 35V
C41,45,62,66 6.8 ±10% 35V
C1 THRU 8, 17 THRU 22, 24, 26, 37, 39, 40, 43, 46 THRU 49, 58 THRU 61, 64 68 THRU 70, 86, 88, 89, 103 THRU 115, 119 THRU 122, 125, 126, 129, 132 133, 136 THRU 144, 153, 182 THR 191, 193, 194, 195
J5-4
GND
J5-5
J5-6
E3
C124 10 -10% +75% 25V
C23,25 63,65 6.8 ±10% 35V
C10,12,14,16,28 THRU 35 50 THRU 57, 73 THRU 84, 95 THRU 102 0.1
+12V
J5-2
+12V
C131 1.0 -10% +75% 25V
C42,45,85,87 6.8 ±10% 35V
J5-1
-12V
-12V
R33 180
C9,11,13,15,158 THRU 181 0.1
-5V
VR2 1N5231B
J5-3
+12VB
+12VB
J1-18
J5-7
+12VC
+12VC
J6-2
J6-1
+5V
J1-19
J1-20 THRU 37
+5V
J2-13
J2-14 THRU 25

Continued from page 26.

and not generate any by itself. For the "M" option, the details described above apply.

Scan ID / Verify Sectors (4xh):

This is a very strange command. As far as I know, it is the only one that is totally incompatible between the old AT's hard disk controller and today's IDE drives. It would appear that this command was never used by common system implementation or application software...

For the WD1010 controller, this is the Scan ID command. It takes no parameters at all (except for the drive and head which originally had to be contained in a register external to the WD1010). When the command is started, the controller searches for the next ID field and reads the contents into the task file. This way the actual drive, head, cylinder, and sector size could be examined. The sector number was also transferred into the task file, so the sector numbering order could be figured out by repeating this command fast enough.

For the IDE drives, this is a completely different command: Verify Sectors. It is similar to the Read Sectors command except that no data is transferred to the host, and the "L" option is not supported. Thus, it needs all parameters in the task file. Up to 256 sectors of data will be read into the sector buffer, and their ECC bytes will be verified. The DRQ flag will never be set. The completion status of the command can be read from the status register.

It is interesting that both types of controller/drive support the retry option - so this is the only compatibility of this command.

Format Track (5xh):

Originally, this command was used to physically format an entire track of the hard disk, exactly as it's done when formatting floppy disks. The Format Track command is started similarly to the Write Sectors command: first the task file must be set up, then the command written to the command register. After that, the drive responds by setting the DRQ flag. The host must then write data into the sector buffer until the DRQ flag is reset. After that, the command is executed.

For the format command, the sector buffer must contain special data. As with the index field array when formatting a floppy disk, it must contain valid sector ID's for every physical sector of the track that will be formatted, beginning at the start of the buffer. Each sector ID in the buffer consists of two bytes. The unused remainder of the buffer is ignored by the format command, but must also be written for the DRQ signal to disappear.

The first byte of each sector ID is a flag byte. The WD1010 knew only two different values for this descriptor:
00h = good sector,
80h = bad sector.
Today's IDE drives offer two more descriptor values:
40h = assign sector to alternate location,
20h = unassign alternate location for this sector.
We'll look further at these values below.
The second byte of each ID is the sector-number byte. It contains the number by which the related sector is referenced later during normal r/w operation. The ID fields in the sector buffer are assigned to the physical sectors (created through formatting) in the order they are stored in the buffer. So it is possible to define an interleave factor by appropriate physical sector numbering. Here is an example:

```
Addr.  00 =  00 01  00 11  00 02  00 12
       08 =  00 03  00 13  00 04  00 14
       10 =  80 05  00 15  00 06  00 16
       etc.
```

Here we see the first 12 (of 32) ID words. The starting sector has number 1 (as usual). The interleave factor is two, since each sector appears two sectors after its logical predecessor. You can also see that sector number 5 (the 9th sector physically) is marked bad.

Due to surface errors on the hard disk, there are some positions where the media won't store magnetic information reliably enough (if at all). The defect list for a particular drive then shows the cylinder, head, and "BFI" (byte from index) value of the defect. People then had to calculate the bad-sector position and number from each of those BFI values. However, it is not commonly known that the relationship between the BFI value and the sector number depends not only on the sector size but also on the interleave factor and the starting sector number...

Again, things changed as the years went by... I already mentioned when introducing the features of modern IDE hard disks, that those drives don't have defect lists any more, due to the usage of internal spare sectors. For compatibility reasons, these drives still accept the Format Track command. However, most drives only simulate its execution – internally they don't really format any track. Modern drives are "hard-sectored" by the manufacturer, with the sector size unchangeable by the user. But by virtually formatting a track, one can assign new sector numbers (for example, starting with 0 instead of 1). However, the sector numbering order is often ignored. Because IDE drives commonly have built-in cache memories, the definition of an interleave factor would make no sense. So, the drive always uses the fixed sector ordering which gives maximum performance in combination with the cache.

To make things still more complicated, the Format Track command of IDE drives allows for the assignment of data sectors to the spare sectors and for the release of those assignments (look at the descriptor bytes above). All IDE drives have some spare sectors to which the data of defective sectors is automatically mapped. Normally, there is one spare sector per track, resulting in about 2-3% spare capacity. This is more than enough. When a sector appears too unreliable during normal operation, the drive simply marks that sector as bad internally and moves the data to the nearest free spare sector. As long as not all spare sectors are assigned, the user won't notice anything. However, these assignments can also be done explicitly by use of the Format Track command. But it is strongly recommended not to do that! First, one will normally get no defect list for an individual IDE drive containing the BFI positions. Second, even if a sector which was assigned to one of the spares is marked good again, the related spare sector can not be used again! So with every unassignment of a spare sector, you loose that irretrievably.

So we come to this result: with standard (i.e., ST-506) drives and external controller (i.e., WD1010) it makes sense to format the drive in order to freshen the surface magnetism, to get a defined state (sector numbering and order), and to mark defect sectors as bad (so that the operating system can behave accordingly). With IDE drives, it's best to leave them just as they are coming from the factory!

Seek (7xh):

This command is used to move the r/w heads to a particular cylinder explicitly. For normal operation of the drive, it is usually not necessary, since all r/w commands perform implied seeks. However, this command can easily be used for benchmarks to determine the drive's seek times. With the WD1010 controller, the four lower bits of the command byte contain the step rate (described above). IDE drives simply ignore these four bits.

Execute Diagnostics (90h):

This command is common to all IDE drives but not available with the WD1010 controller. When issued, the drive performs an internal self-test. If the drive is a master drive, and a slave drive is connected to it, the master also waits a limited time for the slave to complete its self-test. During all this time, it is busy (the according flag in the status register is set). After finishing the test procedure, its results are placed in the error register. In this special case, the content of the error register has to be considered as a single byte value, not as several bit flags. There are the following error codes:

01h	no error detected,
03h	sector buffer error.

(These codes are supported by Conner drives. Maybe other manufacturers use more or different codes.)

If the slave drive diagnostics failed, the MSB of the error register is set, leading to values of 8xh. However, even with single drive configurations this bit sometimes is accidently set. It may be ignored then.

Set Drive Parameters (91h):

An IDE-only command again. After power-up or reset, the drive can immediately be used in its default mode. However, the drive's logical parameters can be changed by setting them with this command. This way, the drive can be set up to different modes in order to emulate the parameters of another common drive. The task file registers which are used with this command, and the way in which they are used, may differ. Some drives are really flexible and allow any parameters that result in no more than the drive's real capacity. Other drives (for example, my Conner CP-3044) support only two or three modes with fixed parameters. So for their selection, only part of the task file's registers are needed. Most, if not all, drives will accept this command with valid parameters in the SC, C, and H registers (even if not all the parameters are required), defining the number of sectors per track, cylinders, and heads.

Because of the differences, it is advisable to first collect detailed information about the supported emulation modes of a particular drive, before defining its operating parameters. Normally, it's best to operate a drive in its native mode (so the logical parameters equal the physical ones). However, there's another strange detail: there are drives which don't support the native mode! My Conner drive again serves as example: the drive has 1053x2x40 sectors (cylinders by heads by sectors) physically, but supports only a pseudo-native mode with 526x4x40 sectors, and an emulation mode with 981x5x17 sectors (which is for compatibility with older 40 MB drives). Additionally, depending on the internal software version, the drive defaults to the emulation mode or to the pseudo-native mode.

As a result, it is recommended that the operating parameters always be defined after power-up or reset. And to define them, you must have detailed information about the drive you want to use. There is a "Product Manual" for every drive type, describing all those details. Unfortunately, these manuals are hard to get. Most dealers are not willing to give them to their customers (and some even don't have them in stock). The other way is to try out some parameters, starting with the information delivered by the Identify Drive command.

The break - a sample program

I realize that I've already filled quite a few pages again. So I'll make a break here and continue the command descriptions in Part IV of the "Connecting IDE Drives" article series. Instead of continuing now, I'll show you a short program which reads the ID information of an IDE drive within a PC/AT. This sample program was written with Turbo Pascal 5.5 but may easily be used with any version above 4.0.

You can try out this program on your AT (if you have one with an IDE drive) and play with it until receiving the next issue of *TCJ* with Part IV of the article. That part will finish the command descriptions and will also contain some more programming examples and shortform tables as a programmer's overview of the IDE interface definition.

Abbreviation list:

BFI	Byte From Index (position of surface defect)
DRQ	commonly used for Data Request (bit flag or signal line)
IDE	Integrated Drive Electronics (hard disk interface type)
I/O	Input/Output
PC/AT	Personal Computer/Advanced Technology (a class of computers)
r/w	read/write
ST-506	older hard disk interface standard, used between separate controllers and MFM/RLL drives

```
program Get_IDE_ID;
(* Q&D 930903 Tilmann Reh *)
(* 930905 MSDOS *)
(* Reads the ID information of IDE drives and displays it. *)
(* Should run with every IDE/AT harddisk drive.        *)
uses crt;
const   SignOn = ^m^j'Read IDE ID Info  V0.1  TR 930905'^m^j;
(* I/O addresses and IDE commands: *)
        IDE_Data        = $1F0;
        IDE_Error       = $1F1;
        IDE_SecCnt      = $1F2;
        IDE_SecNum      = $1F3;
        IDE_CylLow      = $1F4;
        IDE_CylHigh     = $1F5;
        IDE_SDH         = $1F6;
        IDE_CmdStat     = $1F7;
        CMD_Identify    = $EC;
(* Data types and variables: *)
type    WorkStr         = string[80];
        BufType         = array[0..255] of word;
        IDRecord        = record
        Config          : integer;
        NumCyls         : integer;
        NumCyls2        : integer;
        NumHeads        : integer;
        BytesPerTrk     : integer;
        BytesPerSec     : integer;
        SecsPerTrack    : integer;
        d1,d2,d3        : integer;
        SerNo           : array [0..19] of char;
        CtrlType        : integer;
        BfrSize         : integer;
        ECCBytes        : integer;
        CtrlRev         : array [0..7] of char;
        CtrlModl        : array [0..39] of char;
        SecsPerInt      : integer;
        DblWordFlag     : integer;
        WrProtect       : integer;
        end;
var     SecBuf          : BufType;
        IDR             : IDRecord absolute SecBuf;
        Secs            : real;
        i,j             : integer;
(* Convert byte/word values to hexadecimal strings: *)
function  HexByte(x:byte):WorkStr;
const     Nib : array[0..15] of char = '0123456789ABCDEF';
begin
  HexByte:=Nib[x shr 4]+Nib[x and 15];
  end;
function  HexWord(x:word):WorkStr;
begin
  HexWord:=HexByte(hi(x))+HexByte(lo(x));
  end;
(* Swaps the bytes of each "word" in string for correct reading. *)
function  SwapStr(s:WorkStr):WorkStr;
var       s1 : WorkStr;
          i  : byte;
begin
  s1[0]:=s[0];
  for i:=0 to pred(length(s)) do s1[i+1]:=s[(i xor 1)+1];
  SwapStr:='>'+s1+'<';
  end;
(* Show error codes: status register and error register. *)
procedure Error(s:WorkStr);
begin
  writeln('  ',s,'; Status: ',HexByte(port[IDE_CmdStat]),
          ' ',HexByte(port[IDE_Error]));
  halt; end;
(* Wait until drive is ready. *)
procedure WaitReady;
const     TimeOut = 5000;
var       i : word;
begin
  i:=0;
  while (port[IDE_CmdStat]>128) and (i<TimeOut) do begin
          delay(1);
          inc(i);
          end;
  if i=TimeOut then Error('WaitReady TimeOut');
  end;
(* Wait for data request (DRQ). *)
procedure WaitDRQ;
const     TimeOut = 5000;
var       i : word;
begin
  i:=0;
  while (port[IDE_CmdStat] and 8=0) and (i<TimeOut) do begin
          delay(1);
          inc(i);
          end;
  if i=TimeOut then Error('WaitDRQ TimeOut');
  end;
(* Send command to drive. *)
procedure IDEcommand(Cmd:byte);
begin
  WaitReady;
  port[IDE_CmdStat]:=Cmd;
  WaitReady;
  end;
(* Read sector buffer of drive. *)
function  ReadSecBuf(var Buf:BufType):boolean;
var       i : word;
begin
  WaitDRQ;
  for i:=0 to 255 do Buf[i]:=portw[IDE_Data];
  ReadSecBuf:=port[IDE_CmdStat] and $89=0;
  end;

(* MAIN: read drive's ID information. *)
begin
  writeln(SignOn);
  IDEcommand(CMD_Identify);
  if not ReadSecBuf(SecBuf) then Error('Read Identify');
  with IDR do begin
          writeln('ID constant          : ',Config,' (',HexWord(Config),')');
          writeln('fixed cylinders      : ',NumCyls);
          writeln('removable cylinders  : ',NumCyls2);
          writeln('number of heads      : ',NumHeads);
          writeln('phys. bytes per track  : ',BytesPerTrk);
          writeln('phys. bytes per sector : ',BytesPerSec);
          writeln('sectors per track    : ',SecsPerTrack);
          writeln('serial number        : ',SwapStr(SerNo));
          writeln('controller revision  : ',SwapStr(CtrlRev));
          writeln('buffer size (sectors)  : ',BfrSize);
          writeln('number of ECC bytes        : ',ECCBytes);
          writeln('controller model     : ',SwapStr(CtrlModl));
          Secs := int(NumCyls+NumCyls2) * NumHeads *
 SecsPerTrack;
          writeln('total sectors        : ',Secs:1:0);
          writeln('capacity (MBytes)    : ',Secs/2048:1:1);
          end;
  end.
```

Regular Feature

Intermediate

Disk Drives and BIOS

Dr. S-100

By Herb R. Johnson

This month, I'll cover the process of adding a new BIOS and a new disk controller to a S-100 system. But first:

Mail and Messages

Before I answer the mail, I have a request to my readers. I need **Internet** access! There is just too much CP/M activity going on in the Internet that I have no access to. Can anyone offer me legitimate access within the Trenton/ Princeton NJ area?

Several people have written to me in the hopes of finding a "good home" for their S-100 systems. Understandably reluctant to discard them, and unable to sell them in the "IBM" computer world, they believe there is still good life, or at least good tinkering opportunities, with their systems; and they hope I can give them a few bucks too. My general response is to request a list of boards, documentation and software in the hopes I can match up one person's needs with another's. Occasionally I'll buy a few systems that strike me as interesting, or in response to a buyer's requests. Generally I sell replacement cards to people who need a specific board, or in response to someone's need for some additional I/O or memory. With prices low but shipping costs high, especially for heavy cases and power supplies, this strategy makes a lot of sense to me.

Of course, if you need a card, system, or docs let me know! For instance, **Michael Griffin** of Tillsonburg, Ontario Canada writes:

"Some months ago, I was given an old Compupro 8/16 in a pair of 19" rack mount cases. Unfortunately, all of the documentation and floppy disks had been thrown out some time before [a common situation - Herb]. The computer, however, is believed to be in excellent shape. Although it was originally purchased around 1985-86, it saw very little use.

"I would like to obtain information about this system, the cards listed below, and about S-100 Bus computers in general. I don't know enough about this system to even successfully fire it up. It contains the following boards:

Compupro: CPU 8085/8088 [a dual processor card], M-Drive-H [a RAM disk], Ram 23 [static memory card], System Support 1 [a general purpose I/O card, including the console serial port], Interfacer 3 [a hard disk controller], Disk 1A [8" and 5" floppy disk controller]. Also, Illuminated Technologies, this may be a special video card; and SSM IO4 [another I/O card] and a number of custom boards which I do not intend to worry about for now.

"The disk drive unit contains an 8" floppy drive and a 40 Meg Quantum Q540 hard drive [5-inch]. The operating system used was CP/M 2.2, I think [it could also have been CP/M-86, it can run either]."

Michael, I have a system very similar to this and I have some docs I can share with you: I'll contact you shortly in detail. I can provide a CP/M 86 boot disk, I believe. Anybody know about the "Illuminated" card? Also, Michael doesn't mention the hard disk controller in his list. Compupro used a Disk 2 or Disk 3, which may include a second card for DMA (direct memory access, a bus controller to speed up data transfers). Any 5.25" disk controller will have a number of 20-pin connectors and one 40 pin connector.

Thanks to **Kenneth Kutalek** of Evergreen, CO; **Larry Cameron** of Austin, TX; for sending their lists of stuff to me. I'll see what I can do to help.

John Butler of London, England (!) writes to ask if I know about changing the **Osborne Executive** built-in disk formats. He wants to read Kaypro formatted diskettes, but he says the Executive's BIOS has "private codes" in the Disk Parameter Block beyond the typical ones, and he can't find the right combination! He also wants "a driver for hard disks: I know Trantor, etc., did them, but they never published. To import, paying customs duty and insurance is hell - much simpler to buy the code if available. A CD-ROM would be even better!"

Sorry I can't help you, John. It is not an S-100 system, and I know it only by sight. You seem to know the basics of BIOS and disks; you might try to get the BIOS source code and attempt to interpret the private codes. You might try to get on the FidoNet's CPMTECH echo and make your request: there is likely a BBS supporting FidoNet in your general area.

Gregory Nakshim suggested to me that **Stan Veit's** book, **The History of Personal Computers**, is a helpful reference on S-100 computers and related systems of the era. It sells for $19.95 softcover, $24.95 hard. See References for the publisher.

New capability: papertape!

In the old days....when personal computers first came out, there were NO peripherals available, other than whatever could be scrounged from mid-1970's industrial technology. At that time, small data storage was on **paper tape** (figure 0), a sturdy medium consisting of 1-inch wide paper rolls punched with a series of 8 data holes to represent a byte and one "feed hold" of smaller size. The most common type of readers and punches were on Teletype terminals, but mainframe computers and industrial controllers of mills and drills also had these devices.

I attended my first New Jersey hamfest (radio amateur flea market) in mid-October, and I was finally able to acquire a papertape reader/punch from a Heathkit H-11 (a Digital Equipment PDP-11 computer) system. After I test and interface (parallel) it, I should be able to provide papertape to all those classic system owners who have been waiting for **4K BASIC** for the last 15 years! By the way, does anyone out there have a box of papertape for me?

Tutorial Topic: methods for building a BIOS for a disk controller.

Last issue, I described the workings of a disk drive and disk controller, and I provided skeletal code for accessing a disk. Briefly, a diskette has many tracks, each divided into sectors. The BIOS (Basic Input/Output System) code in CP/M provides routines for selecting a drive, track and sector; and for reading and writing to a sector. It also has a "translation table" to convert sector numbers from "physical" (as written into the sector itself) to "logical" in order to improve access times by spreading out the physical distance between logically sequential sectors. (Bill Kibler reviewed very briefly the operation of CP/M and the building of new systems in his "Computer Corner" article in issue 62.)

I had hoped this month I could get into the details of the SD Systems disk controller card and how to use a combination of CP/M and shareware utilities to create a new system from an old disk system. However, after scanning previous issues of *TCJ*, and after reviewing my hours of work on my IMSAI-based S-100 development system, I realized I need to cover the background of CP/M system development and of the bootup process, as well as describe some "secrets" of CP/M development and some key shareware utilities.

So, I will try to work "backward", from the issues of tracks and sectors on the diskettes; to the layout of CP/M on disk and in memory; to how CP/M gets loaded and started. I'll also cover some of the features of CP/M programs like ASM, DDT, SYSGEN and LOAD; as well as DU or DUU, the famous disk utility program that you **must have** to do any useful work on disk development! (Check any CP/M system or utilities supplier for a copy.) Throughout this article, I will sprinkle some "hints" about manufacturer's variations in the features and capabilities I'll describe. You may want to ignore these hints if you are new to this material, and reread them later. You can read the **Sidebars** for details of

CP/M UTILITIES

MOVCPM: This utility rewrites code addresses in CP/M, so that the resulting code can run in a new memory location. The commands

```
MOVCPM * 32
SAVE 44 CPM32.COM
```

will create in low memory a 32K version of CP/M, followed by a save of the memory image to a file called CPM32.COM. Use DDT or DU to examine this file. The number "44" is a decimal count of 128 byte segments (sectors on an 8" SSSD disk). This number may vary from system to system. Too small a number will not save all of CP/M

DUMP: a CP/M utility to display the contents of a file in both hex and ASCII.

LOAD: This utility converts a .HEX file to a .COM file, or executes a .COM file. Caution: it will load the file into the memory referred to in the HEX address records, which may wipe out whatever is there. Use DDT and an offset to load into a different memory area. For example, a .HEX file with the record:

```
nn1000nnnnnnnnn
```

will load into location 1000H with LOAD, but with the DDT command

```
rF100
```

it will load into 0100H instead (F100 + 1000 = 10100, interpreted as 0100).

DDT: CP/M debugger. Commands of note are:

ifilen.com	indicate file "filen.com" for loading or writing
r1000	read the indicated file into memory, offset its loading address by 1000H (.HEX files are read into the addresses they contain; .COM and others into address 0100H plus the offset.
^C	exit DDT, return to CP/M prompt.

How does DDT work? Among other things, it uses a software reset instruction, RST 7 (07H), to return control to the debugger at convenient points in your code. We will use this feature to test the boot code.

ASM: CP/M Assembler

ASM file.abc	assemble file A:FILE.ASM, create hex file B:FILE.HEX and listing file C:FILE.PRN.

the basic features of CP/M and of the utilities.

I should say that a series like this could go on forever with excruciating details and asides. The more readers who contact me with their reactions the better, even if to only say "great!" or "yuck!". I will confide to you (don't tell anyone!) that it can be discouraging to write about an 18-year-old operating system without receiving some feedback!

Disk hardware: This article series will presume a typical 64K (RAM) S-100 system with 8-inch, single-sided, single-density disk drives and controller. The 8" SSSD diskette has 26 sectors per track, 77 tracks, and each sector holds 128 bytes for a total capacity of about 256K bytes. Other diskette densities and sizes will perform similarly but have additional complications, such as density detection and a variety of system track layouts.

My current controller is a Morrow DJ2D, with its own serial port and ROM; the "new" controller I'm adding is a SD Systems Versafloppy II controller, with neither ROM nor serial port. I have the sources for the SD Systems BIOS, but they required much rewriting and lifting of code from other controllers, most notably the original Tarbell single density 8" disk controller.

Recommended reading and software: The Digital Research CP/M manuals describe the operation of all their utilities and the process of BIOS development. If you do not have these, search your local libraries for old CP/M books or books on particular CP/M machines. Or, contact me: if demand warrants it, I may write a more complete description; or find a cache of CP/M documents!

In principle, you should only use one copy of CP/M at a time! And, you should have a serialized licensed diskette from Digital Research. If you need one, I have some for sale; other vendors in this magazine will also sell you CP/M. (Hint: some of this series may also apply to the Z-System, but I don't have enough experience to verify this personally.)

Boot-up operations

For most systems, there is a small bit of ROM (read-only memory) buried on some card that is "jumped" to at power-up or reset (see figure 1). For our purposes, this code eventually loads in some "boot" code from a floppy diskette, which will in turn read in the "system" code on the disk containing CP/M and the BIOS that is particular to your system. Because the first track of the disk (track 0) is the easiest to find, the boot code is contained on track 0, sector 1. The system code will follow, on the rest of the sectors of the first track and for all the sectors of the second track. Figure 1 shows a layout of these two tracks for an 8" SSSD diskette. (Hint: different manufacturers may use slightly different schemes! Use DU (See Sidebar) to get familiar with how **YOUR** boot tracks are laid out!) The boot code in the first sector is read by the ROM (1) into a convenient memory area (usually address 0) (2) and executed. The boot code in turn reads the rest of the sectors (3) into the high end of memory (4) and then jumps to the CP/M starting address (of the code just read in) to initialize and run CP/M. (Hint: some systems follow this sequence with the load of a more complicated BIOS from a file.)

This division of tasks is intended to minimize the ROM requirements of the disk controller, and to make it easy to modify the location of the operating system. The boot code on disk is changed to create a new loading address for CP/M, and the operating system and BIOS are changed to support operation in the new area of memory. In the days when memory was not cheap and ROM code features were far from standard, this flexibility was essential.

To modify CP/M, the program MOVCPM created a new "image" of the operating system which ran at the

CP/M INTERNALS

CCP: Console Command Processor. Similar to the COMMAND.COM of MS-DOS and the "shell" of UNIX, this code receives commands from the console terminal and either calls the internal CP/M commands (DIR, DEL, REN) or calls external programs of the form nnnnnnnn.COM, where "nnnnnnnn" is a filename of up to 8 characters. This code is hardware independent, and can be relocated with MOVCPM.

BDOS: Basic Disk Operating System. This controls the opening and closing of files, access to devices, and converts operating system requests to calls to the BIOS. This code is hardware independent, and can be relocated with MOVCPM.

BIOS: Basic Input Output System. A collection of hardware-dependent code for disk access by track and sector, and device access. Each manufacturer or developer must adapt this code to their particular hardware. This code may be relocated by a manufacturer-modified version of MOVCPM: check your version.

jump table: At the start of BIOS is a series of JMP instructions to a defined set of capabilities in a defined order, using the 8080 registers to return appropriate values.

"Synchronization error": When you try to use a utility from one CP/M with another copy of CP/M, particularly the utility MOVCPM, you may see this message. Each CP/M BDOS (CCP?) has a serial number embedded within it that is matched by some of the utilities. A failure to match causes the utility to stop and produce this message. Either use a consistent copy of CP/M or modify the serial number (the latter I leave to the reader for now!).

desired memory size. To modify the BIOS, the user re-assembled it at the desired memory size as well. (Hint: some MOVCPM's will also re-address the BIOS code as well as CP/M, so reassembly is unnecessary.) Traditionally, DDT was used to "glue" these two together in memory, and SYSGEN was used to copy them to the boot tracks. (I'll use DU instead!) Figure 2 shows typical addresses for CP/M's CCP, BDOS, and the BIOS for a 56K RAM machine. (Hint: CP/M systems with double density or larger sector sizes have more room for both a bigger boot program and a bigger BIOS, so these addresses can vary! Read your docs and prowl through your current system with DU.) See the Sidebar for more details on MOVCPM, DDT, DU, and other terms.

Adding a new disk controller

CP/M was a success because of its unique ability (at the time) to be adapted to any Intel 8080-based system with appropriate hardware: RAM memory available from address 0 on up, a disk controller, and a serial port. The operating system itself (CCP and BDOS) were "movable" without re-assembly by the use of MOVCPM, a program that would modify the code addresses within CP/M, to allow the code to run at the top of whatever memory was available. To add hardware, a manufacturer (or even user!) need only write a BIOS of a standard form, including at the start of the code a jump table: a series of JMP instructions to routines that return after doing standard things, including the disk operations I described in my previous article; as well as console (terminal) and printer operations.

For this series, I'm going to avoid the **hard** problem of trying to bring up a "diskless" system with a new controller. Instead, I'll show how to add a **new controller to a system which has another disk controller.** (Hint: if you have a ROM-based system that allows you to modify memory, you can use that to load and test code instead, from another system which acts as a terminal.) How can this be done without interfering with the old controller and "old" CP/M?

1) The two controllers must not overlap in memory or I/O space. Shared addresses will create obvious conflicts. However, they can share resources, say a serial port or ROM code as only one operating system will be running at a time.

2) The original CP/M must run in higher memory, while the "test" CP/M will be loaded and run in lower memory; and these must also not overlap. See figure 2 for a reasonable memory map of such a configuration. Note that DDT resides just below BDOS when DDT is loaded; its size must also be taken into account. Building a 32K "test" system will give plenty of room for both CP/M's, although it will be too cramped for any serious work later.

3) The test controller's boot sector code must not load below location 100 or above the test CP/M's memory areas. Location 100H is a reasonable place to load and execute the test boot code. For convenience, let's call the source file for the boot sector code BOOTDISK.ASM (this name is not in any manuals!).

4) You have (at least) one drive for the original controller, and one drive for the test controller. You can do this on your current two-drive system by disconnecting the second drive from the disk controller cable, changing the drive's address (usually from "drive 1" to "drive 0") and attaching that drive to the second controller. See figure 3 for the layout. (Hint: this scheme allows you the option of flopping the second drive between either of the two systems, as restoring two drives to your original controller will speed up development.)

5) You will need a bit of code to load in the boot code from the first sector on the bootable disk. This code will eventually reside in ROM, so let's call it BOOTROM.ASM. (This name is an arbitrary name too!) It will be similar to the disk's boot code, but it only need read in one sector and to jump to it. Be careful it does not overlap the disk's boot code, or it will try to load the disk's code over itself!

Testing for loading and "good" addresses

With this scheme, here's how you will test your new controller's BIOS and CP/M. First, to verify that your boot code works and that the BIOS and CP/M are properly relocated:

1) Boot up your current CP/M on your current controller. Edit, assemble and load your new BIOS and disk boot code BOOTDISK.ASM . Make sure BOOTDISK will run at 0100H, and change the code to jump to DDT after loading via the RST 7 instruction (see Sidebar). Move the assembled code to the system tracks of your test boot disk (with DU, details on this move later!). Place your test disk on the test controller's drive.

2) Use DDT to load the BOOTROM.HEX file (the assembled boot code) and to execute it. The "ROM" code will run the other controller and drive, load the boot sector and execute it, which will in turn load the new CP/M and its BIOS into memory below the "current" CP/M. Instead of executing the code, it will return control to DDT.

3) Examine memory with DDT and verify that all the code was loaded at the appropriate addresses. (Hint: for front panel machines, or machines with ROM monitors, you might use these capabilities to verify proper and complete loading.) Note any problems. Reboot the old CP/M system and make appropriate changes to the code.

4) Repeat steps 1-3 until satisfied with the operation of BOOTDISK and the layout of CP/M and BIOS. Then, modify BOOTDISK.ASM to give control to the new CP/M after loading it. Assemble

and load the new version, and place it on the test boot disk.

To verify your code is "properly loaded", consider the following:

1) A hex dump of the beginning of the BIOS code looks something like this:

```
C3 21 73 C3 44 73 C3 82 73 C3 01 74
...
```

the "C3" is the jump (JMP) instruction; and the 2 following bytes are the jump address, the low byte and the high byte. For example, the first three bytes shown above are the instruction "JMP 7321". The high byte should be consistent, starting low and changing by one after a few "jumps". If you use DDT to look at these in memory, the code should be located in the same area as the first few jumps. That is, the first few BIOS routines usually reside at the beginning of the BIOS area; in this case at 7300 and above. If you see an inconsistency, you have either loaded the BIOS into the wrong area, or you assembled it for the wrong area.

2) A similar rule of thumb applies to CP/M. If you use DU or DDT to look at the first bytes of CP/M on the boot tracks, you'll see something like the following:

```
C3 nn nn C3 nn nn ........      *CxxCxx.......*
(more code)                     *  COPYRIGH*
                                *t (c) 1979 DIG*
                                *ITAL RESEARCH *
(zeros)
```

This marks the beginning of CP/M CCP. Look at the jump addresses in this area, and see if they are consistent with where the CCP will eventually be loaded. BEWARE that DDT will overwrite the CCP area, so you can't use DDT to look at the CCP in memory! However, you can use DDT to look at BDOS and BIOS.

Once you are confident that loading and addressing is correct, use a similar scheme to the above to test and execute the new BIOS and CP/M. Run simple CP/M commands like DIR, then STAT. PIP a large collection of files, using the [v] option to verify them. Then, ASM a few files. I like to use the shareware utility CRCK (checksum) to read all the files and write a file of checksums as a test of CP/M operation.

Summary

We have reviewed the features of CP/M relevant to BIOS development, "walked through" a development cycle and described the layout of memory. Next time, I'll show more specifics of the two disk controllers, particularly the SD Systems Versafloppy II card, and its BIOS. I have several of these cards available if anyone is interested in following along on their S-100 system!

References

I am curious: does anyone use these addresses? Please contact me and tell me! They are here to encourage contact and assistance: use them!

John S Butler, 16 Uphill Drive, London, NW9 0BU England. phone 081-204-7203.

Michael Griffin, Apt 207, 182 Lisgar Ave, Tillsonburg, Ont. Canada N4G 4L2

The History of Personal Computers by Stan Veit. Published by Walt-Comm Press, 65 Macedonia Rd, Alexander NC 28701. (704) 252-9515.

Shareware utilities

DU or DUU: Ward Christian's Disk Utility. This program allows you to read individual or sequential sectors on a disk, gather them into a memory buffer, change disks, and to write the buffer out to another disk! This utility will save you the trouble of writing a SYSGEN program, plus enable you to move assembled code from disk to disk. The DU commands needed are illustrated as follows:

la;	load diskette on drive A as the current disk
t2;s3;r;d	Track 2, Sector 3, read it, dump to screen
r;<<;+;/10	read it, save sector to buffer (as newest) and accumulate, go to the next sector on disk, and repeat this line of commands 10 times
>>;w;+;/10	recall the first (oldest) sector in the buffer, write it, go to the next sector on disk, and repeat this line of commands 10 times
?	display all the commands

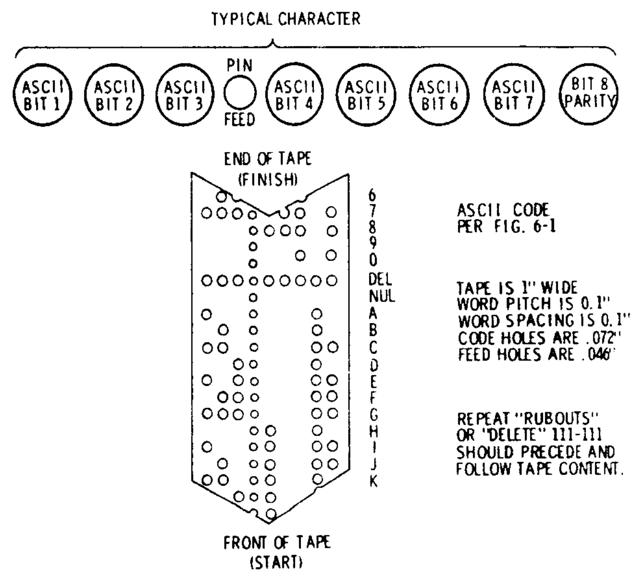
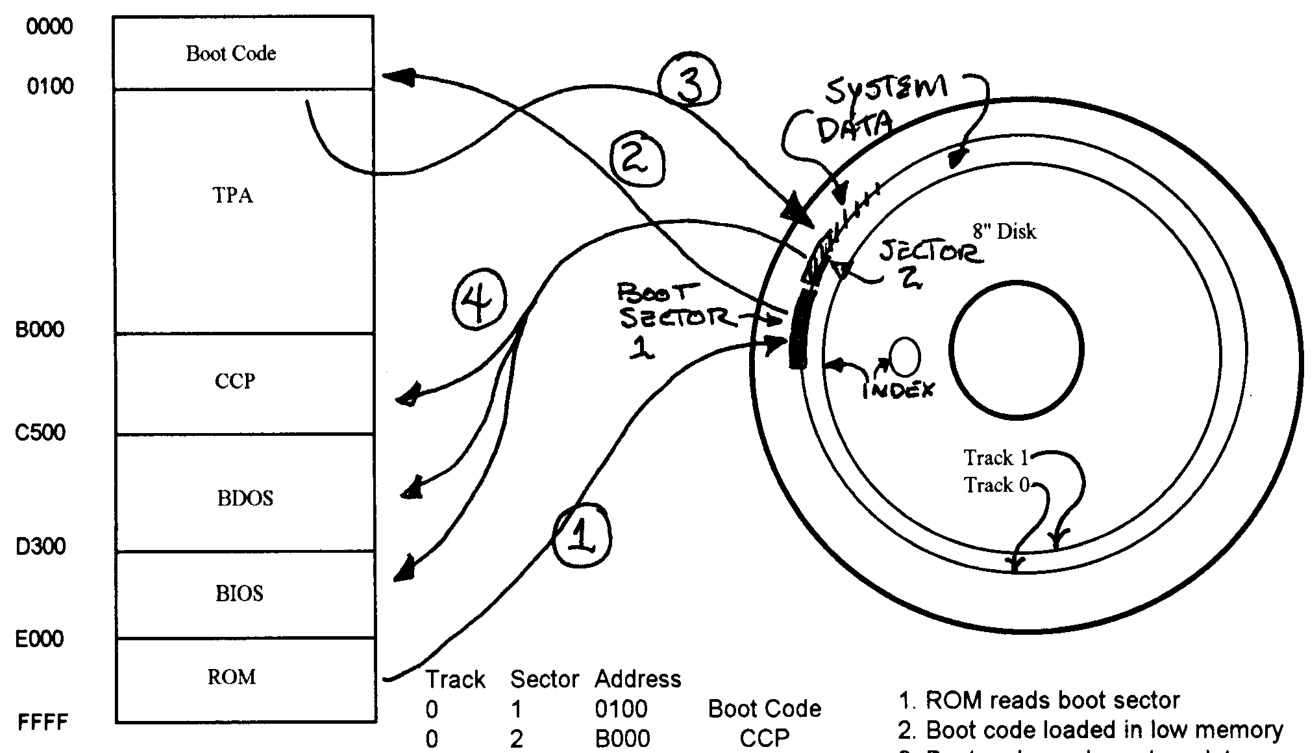


Figure 0: Papertape.



Track	Sector	Address	
0	1	0100	Boot Code
0	2	B000	CCP
0	18	C500	BDOS
1	20	D300	BIOS
1	26	E000	End of BIOS

1. ROM reads boot sector
2. Boot code loaded in low memory
3. Boot code reads system data
4. System loaded in high memory

Figure 1: The boot-up process, with representations of code in ROM and the sequence of system code on disk.

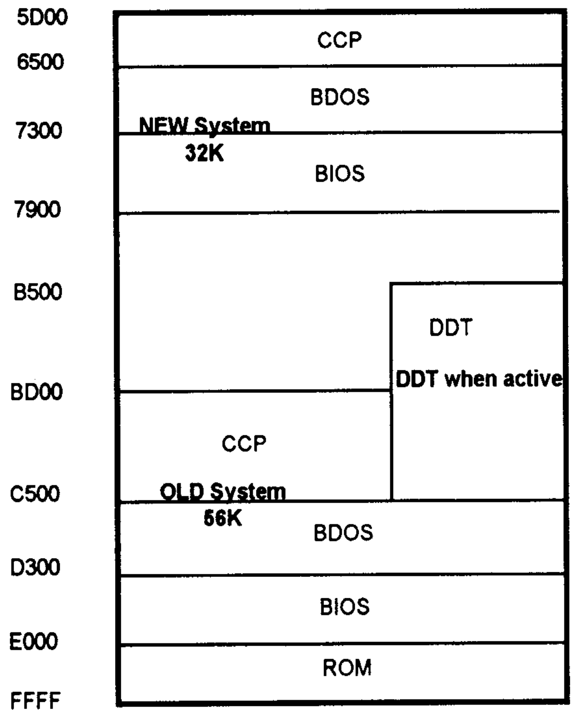


Figure 2: The memory map of CP/M. Also the BIOS JMP table. Two memory maps, one for 32K development and one for maximum memory with the DJ2D.

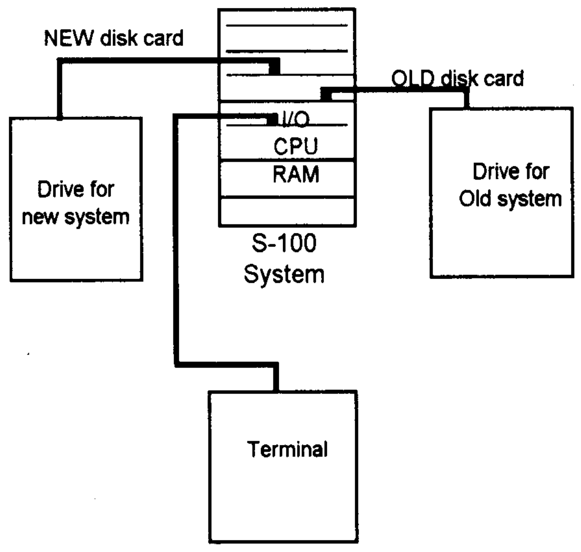


Figure 3: hardware configuration for development. Two disk controllers, two drives.

Special Feature

Intermediate Users

Part 1: What is Available

SMALL -C ?

Introduction by Bill Kibler

Some issues back, I broached the subject of a magazine endorsed programming language. The idea was to use some standardized programming environment and language for all articles. The magazine would make it possible that this choice would be available for all platforms we support. Thus, it would provide all our readers with a learning and working set of language tools at an affordable price. Currently, there is no known standardized language available across all platforms.

Forth is probably the closes language that actually attempts to be similar across all platforms. Even still, there is not a single version of Forth that works completely the same on all platforms. Many of the older classic systems came with BASIC in ROM or on disk. The BASIC's were not anything approaching implementations of a standard. You can expect each BASIC platform to require major changes in coding to move from one platform to another.

For teaching, PASCAL is considered superior. The structured design of PASCAL was intended to teach programming language concepts and practice. Several implementations have proven that the language can also be used quite successfully outside of the educational environment. The company Borland rode to success on their Turbo Pascal implementation for CP/M and later MS/DOS. Turbo Pascal's success in part is also the integration of programing and editing tools in one package.

Unix users have come to enjoy using C and find many features they consider superior to any other language. Of interest to us, is the migration of C, back to smaller platforms. Ron Cain implemented a version for small platforms with the intention of teaching users about C. That version was called "Small-C" and was explained in a book about the language. Since this version was very small and self compiling, many options and features were not provided. The basic set of functions however was easily ported to other platforms.

Since the direction at *The Computer Journal* is to teach our readers how to perform simple and basic tasks using tools suited to their platform of choice, finding a language has become an important decision.

One language that has been broached is of course Forth. I am a long time user of Forth and have seen many items available which might make it our best choice. There are BASIC and PASCAL interpreters that run under F83 Forth. I am aware of projects to create C interpreters using Forth platforms. Forth is also available in several versions of C source for use on UNIX and LARGER platforms. A new version of Forth for embedded controllers, EFORTH, is just about ported to all platforms. This might give our readers an across the board single implementation.

Since all users are not especially fond of Forth, an alternative language and options should be considered before making any long term choices. I proposed that Small-C might be that alternative. Current Small-C implantations are fairly extensive, but have the same problem as Forth, major variations between platforms. Since my experience with these variations of C is limited, I have turned to our readers and writers for input. Not all our readers feel strongly that even C would be a good choice. Some experiences have shown the problems that need to be addressed.

What follows is most of what has been sent to me to date. I have added index summaries of the C Users Group listings of the disks on which Small-C versions are to be found. I did not list any of the utility programs that have versions for the Small-C language. I found at least 10 to 15 disks that have separate tools for Small-C users. This last fact should be considered as a plus for using Small-C. Accessory tools are very important to any language and having tools already done must not be over looked.

Lets hear then from our writers and readers in this first installment of "to C or not to C?" (Sorry for the pun..but have fun.... Bill Kibler.)

Dear Bill:

I've been thinking about this controversy that seems to have come up regarding the code for the serial gizmos Walt wants to design. While the Forth that Brad Rodriguez and Frank Sergeant write is pretty readable to me at least, in the past I have had terrible problems with reading Forth.

But, as you mention, assembly takes up too much space, and Pascal would be too cumbersome for the project to be written in. I don't like pseudocode, because it isn't the real code, and when you actually try to write something, little gotchas always come up.

I have two ideas. First, we write a Pascal-to-Forth or Small-C-to-Forth translator, in Forth, to allow conventional block-structuring, top-down, algebraic coding. Really, it isn't Pascal nor C, though, but a preprocessor/reformatter that does not change Forth at all, but reorders and reformats the statements so that they look conventional.

The bad part about that idea (which is not a new idea, by the way), is that the Forth folks won't like it (''it just isn't Forth''), nor will the Pascaloid folks (''it just isn't Pascal''). It could be argued that it really is Forth, however, if you remember Prolog, there were several different syntaxes for it (''Edinborough'', ''Clocksin-Mellish'') which were quite different-looking. Who says Forth can have only one appearance? The translator could be written in Forth or in M4, awk, C, Pascal, BASIC, whatever.

OK, then, the other suggestion is to find a way to make the assembler take less space. One way is by putting multiple statements per line:

```
    inc dptr : movx a,@dptr : ret
```

The other way is by using macros. As a long-time assembler programmer, you know problems with macros: ''What does it do, again?'' ''What side effects does it have?'' And when you forget what the macro does, the macro code itself is unreadable. And, of course, every assembler has tremendously different macro syntax, so macros aren't portable.

We can solve all of the latter problems by writing our own macro preprocessor in BASIC. Why BASIC? Because BASIC, is the language most of us have, and it's the most portable language we have accessible. You might even be able to run the preprocessor on an 8052 chip with built-in BASIC.

But why stop there? We can write our own assembler, too! OK, now I imagine I'm getting carried away.

Have fun!
Rick Rodman

From Tilmann Reh over numerous E-Mail discussions concerning the choosing of a language.

<From June>
Concerning programming languages, I do not like Forth as a medium to describe some software technique, even if it is very portable. Most people will be unable to extract algorithms out of a Forth listing! So we always should use 'algorithmical' languages to explain our ideas. The ideal language for this purpose is Pascal (or Modula), as explained earlier. Every Forth programmer will be able to port an algorithm from a Pascal source to his Forth assembler. This does not apply vice versa. So, the only really portable software language is the algorithm itself, or anything similar (like Pascal).

BTW(By The Way), your argument that Pascal is too strict for hardware level programming does not apply. Common Pascal compilers all offer some method to access the hardware directly. At least you might define subroutines located anywhere which do some special hardware-related functions, and are called by really readable symbolic names. From my experience of programming (high-level and assembler), there is absolutely no argument against structured high-level languages even when programming at the hardware level, except for some run-time limitations in some cases. Instead, software development times are much shorter when using (for example) Pascal.

<From Late June>
I still am sure that Pascal (or Modula, Ada) are the languages of first choice if you want to EDUCATE. This because these languages express the used algorithms very directly and clearly. If we really want to educate our readers, we must show them HOW to solve a problem, and not print a listing of a special implementation which is unportable to other programming languages! So may I repeat that the structured modular languages (mentioned above) serve this purpose much better than any other, even C. Every reader who does programming himself will then be able to adopt the used method to the language of his choice, which is (nearly) impossible with every other language (I think).

I don't know a Pascal for OS-9 or Flex, probably because I never used those systems. But I am sure there must be something... And if not, see the last sentence of the previous paragraph.

Last but not least, be aware that there are no efficient C compilers for microcomputers, especially for the 8-bit world. The code size and efficiency is so bad that even in this concern Turbo-Pascal is better. (So you won't help people with <64k systems by using C.)

<From July>
I must agree that BASIC is best available for any small computer system. However, BASIC is by far not the right language for teaching anything. The only thing you can teach with BASIC is a bad programming style... May I repeat my ''General Portability Rule'': You can very well port a Pascal program to BASIC, but it will often be very hard work to port a BASIC program to Pascal (when saying ''Pascal'', I also think of other structured languages like Modula or Ada). This is because BASIC is not strict enough, and uses jump instructions (GOTO) very often. The only way to educate though using BASIC, is writing BASIC programs as if it were Pascal! If you have someone who edits all incoming BASIC programs according to this rule, I might as well agree to use this language more often.

<From August>
I agree that BASIC can also be programmed well-structured. But the fact is that *most* BASIC programmers *don't*. I think it's not possible to change this, so I would discard BASIC for our purposes.

I don't think that Forth is better than Pascal. Nor for real work, neither for education. May I repeat again (the third time?) that a Pascal source can be read by everyone and easily translated to any language of his choice, but this is absolutely impossible with a Forth program (only if the reader is well used to Forth). The optimum would be to teach only algorithms, but if we print programming examples, we should use a language which is nearest to the algorithm, so *every* reader is able to understand and translate it, if necessary. I think the only language which reaches this goal is Pascal (or Modula or Ada, which however are less usual; or structured BASIC, which is possible, but in fact even less usual).

Sincerely, Tilmann Reh.

Dear Mr. Kibler;

In your 'Computer Corner' in the #61 issue of *TCJ* you mention Small-C. Last year I had some experiences with Small C on a 6809 and I thought I'd share them with you.

I have 4 Small-C's. The first two are 6809 Small-C's from the C User Group. The third is Byte Magazine's Small-C for MS-DOS. The fourth is F. A. Scacchitti's Small-C for CP/M.

The first one I got was Byte Magazines' MS-DOS Small-C for the 8088. It is a fairly 'complete' Small-C and is reasonably structured. It has a fairly large standard library.

The two for the 6809 from the CUG aren't very good. The earliest one was CUG 132 and is very old and incomplete. The later one is CUG 309. It is a port of CUG 221. It is much better but it doesn't have a lot of library routines. It does however have an optimizer to clean up some of the terrible code that Small-C generates. I think there is a bug in the code, but I can't remember what.

I only 'recently' got F. A. Scacchitti's, so I haven't done anything with it.

Most of my experiences with Small-C come from Byte Magazine's Small-C. It was more complete than the two 6809 ones I had and I didn't yet have F. A. Scacchitti's (CUG 222 & 223). Converting Byte's from MS-DOS 8088 to 6809 OS9 was reasonably straight forward. (Actually I never finished with it. Small-C wasn't complete enough for what I wanted it for). Converting the output from 8088 to 6809 wasn't too hard. Also there were just a few minor changes for my assembler. I also had to change a few things because 6809 OS9 requires the program to be non self modifying and Small-C allocated variables right into the program section. (I wish all of the stuff that needed to be changed was in a single file rather than spread throughout the program.) However, the code that was generated was terrible.

At this point I began looking for other compilers. I eventually found Dunfield Development in Canada, but their C compiler was $50 (including C source) plus $50 for the 6809 generator and it had a few limitations as well. It was better than Small-C, but not enough to be worth my spending $100. It had code generators for several CPU's, including the 6809. The code generated though was vastly better.

I spent quite some time trying to clean up some of the terrible code for the stack but didn't have a great deal of luck. Some things went fairly easily, but since I didn't (and don't) have the book I wasn't sure about some stuff and I was just blundering around in the dark. I also had problems coming up with test suites to test my modifications.

I've been trying for months to talk Microware into releasing the source code for their 6809 OS9 C compiler. They stopped selling it about 5 years ago, don't support it, and barely admit to it's existence. It is a fairly complete compiler, including structures. I haven't had any luck though and I don't really expect to. It is a multi-part compiler which reduces the memory strain for 64k machines. I've examined the disassembly and discovered that it was self compiled so it would be possible to regenerate the C source code. It would be a LOT of effort though. The preprocessor ($27AC), the analyzer ($7B76) and the code generator ($6444) sections total to about 64K worth of code. That would be a LOT of tedious trial and error to regenerate the C source (I did that with the C library, so I know). It also has a couple of bugs that need to be fixed. If the source could be obtained it would be a fairly decent general compiler.

I've also tried to track down several others, but with no luck. That was why in a previous issue of *TCJ* I asked if anybody had a K&R C compiler. I've received two, but they are fairly large. I don't have a modem or I would check CompuServe, BBSs', Delphi, etc. Perhaps somebody might be able to talk the authors of some of the public domain C compiler's (in executables) to release the source code.

If *TCJ* does decide to 'sponsor' a C compiler. I have a few suggestions.

1: Before any work is actually done on Small-C again, I think it would be a good idea to put out a 'call' on major BBSs', CompuServe, computer nets. etc. Looking for a newer compiler base to work with. It would have to be very small, but maybe at least it would be better organized than Small-C is. It would only take a couple of weeks and the results might be substantial.

2: If you do decide to standardize on Small-C, you are going to have to decide which one. There are a lot of them out there and most have been 'extended' in non-portable ways. Also, most of them have changes made inside the compiler itself (cc1x.c-cc3x.c) rather than limiting them to the output part cc4x.c. Most of the changes are for assembler compatibility. Some might be tempted to suggest using the version in the book, but most don't have the book and would have to buy it for $30. I think that particular version in the book has been copyrighted so it can't be shared. If you are going to end up spending money, you might as well buy Dunfield's for $50 and make your own processor output (it comes with 8088, you have to buy others or write your own).

3: The compiler should output P-Code. That way a single compiler can incorporate all improvements for all CPU's. To convert from the P-Code to regular ASM would just require a simple translator. If you also bracket variables with pcodes indicating that it is a variable, that would allow a wide variety of formats. If you also indicate beginning/leaving function then you could also easily add custom code in the translator. The bracketing of stuff would allow for ROM/RAM (after all, you can't put variables in ROM), etc. The translator could also take care of the format of labels etc. That would completely separate the assembly format and the opcodes from the compiler itself. Nice, neat, and generic. Isn't that what you want from a portable compiler? The only 'catch' is there is an extra step in the compile process; converting the pcodes into assembly. A simple shell program would handle that. *TCJ* could do it sort of like the FSF does GNU, improvements are regularly incorporated into the master version. That would require somebody to maintain it though. All of this would also make cross compiling a lot easier. The only problem with pcodes is embedded assembly instructions in it. Again though, pcodes could bracket it and tell the translator about it. At the very least, all changes for porting should be limited to CC4x.C and not spread throughout the whole compiler.

4: It might be nice if the compiler was in parts. A preprocessor, a tokenizer/parser, and a code generator. The C compiler for 6809 OS9 is in 3 parts (listed above) and also has an optimizer, assembler and linker. Doing things in parts does cause a few problems, but it allows it to handle larger programs, more symbols, etc. My OS9 C compiler compiles the Byte Small-C to about 29k of code. Small-C compiling itself would be much larger. That is less than the individual parts of the full K&R C complier of mine.

5: It really should be expanded to include multi-dimensional arrays, as well as arrays of pointers, unsigned integers, multiple levels of indirection pointers, typecasts, sizeof(), etc. Structures would be nice (and not quite as hard to implement as may be thought), but could be worked around if you have other C stuff. The limitations of Small-C are just too great for a person to WANT to work with it. Improvements to Small-C would have to be made.

6: A better preprocessor would also be nice.

7: It is going to HAVE to generate better code than the original Z80 CP/M version. I know that it will be used on a Z80 so better code would have to be an option. Perhaps a pcode 'optimizer' to go through and clean up the stack indexing code for non-Z80 CPU's. I don't know, but I do know that normal Small-C generates such terrible code that most wouldn't want to use it. They'd probably get better performance from Forth.

8: I think that Small-C should be reorganized. That might make it much more readable and modifiable. Small-C has a serious problem in that it has been upgraded and had features added so much that the general structure isn't good. A lot of improvement could be made by defining a few labels for the expression analyzer indexes. Change a few names of some of the routines to make them more self explanatory would also help. Reorganize the source into distinct sections, etc. Nothing major really all of it could be done with a good word processor and a day or two.

9: There should be some form of register assignment in the outputted assembly. What I mean is that the compiler should be able to recognize that not all registers are data registers, some are address registers. One of the problems with Small-C is that it was almost hardwired from the beginning for the 8080 and its descendants, the Z8O and 8088. When I attempted to port it to the 6809 I had some problems because the 6809 has one 16 bit data register and 3 address registers. The compiler expected the secondary register to be a data register with address register capabilities. That caused major problems in most operations because I couldn't do math operations or, the secondary register because it was an address register. Instead, whenever a transfer to the secondary was to be made I pushed it onto the stack. That resulted in even worse code and left the address registers almost idle. As I said before, the Small-C compiler has some MAJOR problems. A replacement should be found, or at least major changes to the expression analyzer should be made.

10: There should be some list of all the features of Small-C as compared to K&R C. If there is a list of features and abilities of each, then people would know what needs to be added or changed. As it is, its just 'shooting in the dark'.

11: the size of the tables should not be fixed at runtime. Instead it should dynamically allocate them as needed. That would involve a number of changes to the tables, but it would allow a more reasonable approach to the 'size' of the programs that it can compile.

To summarize, Small-C would have some major improvements. If possible, an alternative C compiler should be found.

If Small-C does have to be used, I'm not sure what base to suggest using. Either Byte's or F.A. Scacchitti's. I haven't studied FAS's much, but I don't really like the way he did the multiple dimension arrays. Of course, I can't say I could do better. FAS's and Bytes seem to be the same base. FAS's has a few additions and Byte's has more libraries. Byte's is for 8088 MS-DOS, FAS's is for Z80 CP/M. Flip of the coin basically.

I personally have only one computer, a Tandy Color Computer 3 w/512K, a single 360K floppy, and OS9 and its C compiler. I can't really judge what would help other computers. The operating system is in one 64K bank and each process gets its own 64K address space all for itself. That allows larger programs than other computers that have to have the OS in the

same program memory.
Well, I just thought I'd put in my 2 cents worth.

Carey Bloodworth.

P.S. I've managed to learn of the existence of 2 old, portable, fairly small K&R C compilers. So far I haven't been able to actually locate them though. C.B.

Dear Bill Kibler;

Greetings. I see in issue #61 you're thinking about Small C, so here is a little material in this area: a Small C 68HC11 compiler/assembler. I can't say the 6811 would be my ideal embedded processor, but it has its charms.... See README.SC1 on the diskette for a short description of files, and then there's tons of my documentation in there....

At any rate, the interesting thing here is probably the source files CC4*.C and CALL11.MAC. These are supposed to be the machine-dependent part of the compiler, although I won't swear there isn't a little stuck somewhere else in the source. These are the parts you have to change to adapt the compiler to another target. It's not exactly child's play, but it isn't that hard - you need to know at least a little C, and be familiar with the target, and of course have a test system.

As for the thing itself, I discuss this in the docs, but it's really not the world's most wonderful program, at least source-wise. This is partly because of the almost total absence of comments (I mean, Hendrix probably created it on 128K diskettes); to this day I usually can't figure-out what's going on in there and Heaven knows, I've tried. Hendrix or someone apparently referred to it as an "educational" compiler, and this stuck, leading to years of silly magazine remarks about how it's so useful for learning how a compiler works. It isn't. What I think was meant was that it was good for schools etc., because it was cheap/free. Be that as it may, it's still a useful compiler, and it is quite feasible to re-target it....

Language-wise, the most annoying thing about it is the absence of structures (like Pascal records). Otherwise its restrictions are fairly bearable. I'd still like to get hold of a decent public domain K&R source with structs and everything - but then it probably wouldn't fit on my hard drive anyway....

Bulletin: On my quest to find a book describing the PC hardware, I found *one*: *Interfacing to the IBM Personal Computer*, second edition, by Lewis C. Eggebrecht, $24.95 (1990, SAMS). I've shelved my DMA project for a while, so I haven't put it to a real test, but it *looks* pretty good. Genuine timing diagrams, schematics, register descriptions....

J. G. Owen, South Huntington, NY.

Small C 68HC11 compiler, assembler. (to be found on JW Weaver's BBS - SEE page 10 - Support Groups.)

Thu 07-22-1993.
This package includes a lot of material, and zip files within zip files. It is primarily a 68HC11 product, but includes an 8048 assembler.

A good deal of this stuff is derived from products of J.E. Hendrix, and he may well own the copyright to them. Various documentation herein elaborates on this. As noted, the documentation is relatively extensive, including lots of deep thoughts about Small C, embedded systems, and other fascinating topics.... Note however that neither Small C or C language is documented in here.

```
SC11.ZIP

SC11.DOC      describes compiler operation, including
              essays on the meaning of things.
README.SC1    this file.

SC11SRC.ZIP
CC*.C etc.    source for 68HC11 Small C (TC 2.0).
SC11.EXE      the compiler.
GOODBYE.*     demonstration project.
EDIR          describes some of the files, short history
              /notes on project.

XMAC.ZIP
MACOP.DOC     describes assembler, linker operation,
              with much digression.
M6811.EXE     68HC11 assembler.
M48.EXE       8048 assembler.
LNK.EXE       linker.
BS.EXE        symbol/file converter.
68HC11.MIT    op table.
48.MIT        op table.
```

FILES
Files that should be associated with this documentation:

```
MACOP.DOC      This document.
M6811.EXE     68HC11 assembler.
M48.EXE       8048 assembler.
TEST6811.MAC
TEST6811.LB
TEST48.MAC
TEST48.MAC     Test files.
T48.BAT
T68.BAT       Process TEST*.ETC.
68HC11.MIT
48.MIT       Configuration files.
LNK.EXE  Linker which is supposed to work with
         both M6811, M48.
BS.EXE   Link-to-Motorola-S-record translator (for 68HC11).
```

Note that LIB.EXE, a library program which originally was included with the Hendrix Small Mac package, is NOT included here, because it almost certainly doesn't work anymore with the endlessly-hacked XMAC. However, this documentation includes some references to the program, and there are various switches in the linker which relate to it. The LIB program, including the source, can probably still be obtained from Dr. Dobbs.

SUMMARY, HISTORY

This document describes the features of an assembler + linker currently configured for the 68HC11 and 8048 and collectively referred to as "XMAC" herein. This is a major hack of J.E. Hendrix's Small C configurable assembler MAC, which I obtained through Dr. Dobb's magazine offer (see REGISTRATION, below). The programs described here run on MSDOS systems; there used to be CP/M versions, but these have passed into the land of STAT and DISK R/O ERROR. Great quantities of this text consist of MAC documentation.

M48

M48 is the 8048-configured version of XMAC. The 8048 is a very common -- i.e., readily available, cheap, and therefore obsolete -- single-chip microcomputer originally released by Intel but now available from many sources. It is typically used as a micro-controller to provide intelligence for keyboards (IBM PC keyboards are based on this architecture), printers, microwave ovens, etc. The 8748 -- a 2K, EPROM version of the part -- goes for as low as $8 retail.

Developing programs for the 8048 family, even with M48 or some other assembler, isn't much fun without some development equipment in the $1000 to $???? range. Also, an 8748/49 EPROM burner is necessary for small quantity projects. Nevertheless, it is feasible to write simple programs for the 8748 without debugging equipment, providing intelligence for small quantity applications with a start-up cost equal to an appropriate EPROM burner. (Honestly, I haven't done it; I designed and built an 8048 emulator -- I forget exactly why -- which I then used on a few projects. Designing/building the emulator was fun, exciting, and extremely complicated; I had to buy a better scope just to debug it. I'm not sure I'd like to try an 8048 project without an emulator.)

See "48.MIT" -- the assembler configuration table for the 8048 assembler -- for a list of the exact available instructions.[1]

M6811

M6811 is the 68HC11 version of XMAC. Motorola's 68HC11xx is an amazing, HC-CMOS based single-chip microcomputer which includes everything AND the kitchen sink (i.e., timer, A/D converter, UART, radar-range, etc.). It has a somewhat enhanced 6800-type instruction set (actually, 6801), but still reflects the tedious limitations of that architecture (i.e., it's no 6809). Obtaining documentation may be difficult. Also, support equipment is ridiculously expensive (//this was written in 1987 or so; support equipment since then has become comparable to that for other microprocessors; not cheap, but not that offensive -- i.e. $1000 to $10,000).

Somewhere in the vicinity of this file may be 68HC11.MIT, which can provide some guidance as to exactly what syntax I had in mind for this device; I was attempting to follow the Motorola specification, but note that specifically- Motorola conventions are always overthrown for Intel conventions -- i.e., "LDAA #12H", not "LDAA #$12" (actually #$12 probably works now).

BS

BS is a program that translates the output of LNK into a Motorola S-record (i.e., BS, "Binary/S-record", of course). In addition, it can create files suitable for downloading to the TECI 6811 emulator.

James Gregor Owen
Huntington Station, N.Y.

C USER GROUP

The Following is information on what the C User Group has for the Small-C implementation. The information was removed from the Small C Users Group CD ROM.

Important files and directories:

cug_info.txt	Info about the C Users Group
capsule.txt	Capsule descriptions of each file in Volumes 100-299
catalog.txt	Catalog of each CUG Library volume
listings\	Source code listings from The C Users Journal
vol_100\	CUG Library volumes 100-199
vol_200\	CUG Library volumes 200-299
vol_300\	CUG Library volumes 300-364
zipped\	All the CUG Library volumes compressed in zip archives
bbs\	BBS support files

Can be purchased from:

The C Users Group
1601 W. 23rd St. Suite 200
Lawrence, KS. 66046
+1-913-841-1631

Walnut Creek CDROM
1547 Palos Verdes Mall, Suite 260
Walnut Creek, CA 94596
1-800-786-9907, +1-510-786-9907
FAX +1-510-947-1644

================================
CUG104 -

CUG104.27-C1.C By Mike Bernson, Ron Cain. Small C-Part 1. Main line and opening text plus #include, #if, #nif, error summary! dumping extern, and static area for a Small C compiler. Executable image on disk. ->ASSEMBLE.COM, LINK.COM. [CP/M:BDS v. 1.41] This Small C is NOT self-compiling and requires special assembler and linker which are available ONLY in executable form.

================================
CUG132-

This file covers the 6809 implementation of Ron Cain's Small C compiler and a graphics driver/support package for the Radio Shack Color Computer.

To make it:
You must have a functioning version of BDS C on drive A. Move the submit file bldc.sub onto drive A with this disk in drive B. Submit bldc. If all goes without error, you will need to answer the questions at the end of the link phase as to which files must be searched. Input canew in response to the first question, cb to the second, ... , ce6809 to the fourth and the link should complete leaving canew.com on drive B. You will probably need double density drives (1/2 meg) to build on drive b or will have to shuffle the compilation and link operations.

================================
CUG146 -

CUG146.11-SMALLC.C v 2.0 By Serge Stepanoff, Ron Cain. Small C compiler for 6800. A version of Ron Cain's Small-C adapted for the 6800 micro under TSC's FLEX operating system. Initial conversion was done on a PDP 11 running RSX-11 and the DECUS (public domain) C compiler with Small C code from DECUS. Ongoing development of this version is being carried out on a SWTPC 6800 with dual 8 inch floppies and 32K RAM. [Flex v. 2.1:Small C] The TSC assembler accepts any length labels but only the first 6 characters are used and saved in the symbol table. Therefore, if you have either functions or labels of the type MODULE1 and MODULE2, the assembler will generate a multiply defined label error. Make sure that the first 6 characters are unique.

================================
CUG156 -

CUG156.08-C80V.C v. 1.2 By Ron Cain, James Van Zandt. Small C Compiler with Floats. z80 Small C Compiler with floating point math. Executable image is included so that compiler is self-compiling. Produces relocatable assembly for ZMAC & ZLINK (also on the disk). ->CUG104, CUG115, CUG132, CUG146, CUG163. [CP/M:Self compiling]

CUG156.10-CC.DOC C Compiler Documentation. Documentation for the z80 Small C Compiler with floating point math. ->C80V.C. [CP/M:Self compiling]

================================
CUG163 -

CUG163.01-CC11.C By J. Hendrix, Daniel R. Hicks. Small C v. 2. Small C Compiler by J. Hendrix adapted to MSDOS environment. [MSDOS:Small C]

================================
CUG200 -

SCI - Small C Interpreter
This Small C interpreter by Robert Brodt (NJ) is a shareware package available only as an executable image for PC-Clones and is accompanied by two extensive documentation files. A useful learning aid. [share2]

CUG200.05-SCI.EXE 1.5 executable
Small C Interpreter. By Bob Brodt. A small C interpreter, designed to introduce C. Includes a screen editor, & debugger. => USER.MAN PROG.MAN. [MSDOS:] Requires 64K of memory.

CUG200.06-SHELL.SCI 1.5 source
By Bob Brodt. The command shell, written in SCI's dialect of C. => USER.MAN.

CUG200.07-USER.MAN doc
Small C Tutorial. By Bob Brodt. SCI users manual describing shell, editor, language, library functions and debugger. => PROGRAM.MAN.

================================
CUG221 -

6809 C For Flex
A rewrite of Ron Cain's Small C targeted for 6809 processors running under the FLEX operating System. Dieter Flunker (Italy), has expanded slightly on the language subset implemented by Cain and includes a peephole code optimizer. [public]

CUG221.07-CC1 TO CC91 2.3 source
Small-C compiler for 6809 FLEX By Pieter H. Flunkert, Ron Cain. Small-C compiler which produces 6809 assembler out-

put. This volume includes both C source and compiled output (as optimized assembly source). 6809 FLEX: VAX VMS C. Requires TSC relocatable assembler, library generator and linking loader.

CUG221.35-README.DOC doc
Documentation of the disk contents

==========================

CUG222 -

Small C for CP/M Doc and Exec
F.A. Scacchitti's (NY) significantly enhanced version of J.E. Hendrix's Small C v2.1. This CP/M implementation handles global initialization, external statics, the ternary conditional operator, multiple levels of indirection, global multi-dimensional arrays, arrays of pointer, hex and octal constants, and nested includes. Also includes an expanded standard library. Contains the executable COM file, relocatable libraries and user documentation. Source is on CUG223. [public]

==========================

CUG223 -

Small C for CP/M Source The source code for a Small C compiler. A full description appears with the entry for CUG222. To construct the compiler you must have an M80 compatible assembler. [public]

CUG223.02-ABS.C source
By F.A. Scacchitti. Returns the absolute value of an integer. => CUG 222. [CP/M: Small C]

CUG223.139-ZZBUF.MAC source
By F.A. Scacchitti. Used to preserve standard CP/M buffer for input arguments. [CP/M: M80]

CUG223.140-ZZFIO.MAC source
By F.A. Scacchitti. File I/O storage variables. [CP/M:]

This disk contains a total of 140 Source files!

==========================

CUG309-

README.DOC for CC09 MSDOS C COMPILER

Brian Brown, Senior Lecturer, Software Engineering, Central Institute of Technology, School of Electronic Engineering, Private Bag, Trentham, Upper-Hutt, Wellingon, New Zealand

15th November 1989

The Small C Compiler for 6809 running on FLEX (CUG 221) has been ported to MSDOS, and changed to allow creation of embedded target software. It supports the ASxxx group of assemblers (CUG 292) available from the C users Group, as well as the Motorola AS9 assembler.

A host of routines is supplied. These work with particular boards used by our students. The system we have is,

6809 Processor card, 32k StaticRAM 0000 to 7FFF. Addresses 9000-93ff are port mapped to 000 to 3ff for PC type boards. Addresses 8000 to 8fff are mapped to B0000 for PC monochrome cards. On board ACIA, MC6850 at A000 (ControlReg) A001 (DataReg). OnBoard EPROM or StaticRAM C000 to FFFF

The processor card drives an IBM-PC backplane with four expansion sockets, thus can talk directly to standard PC cards. The routines for this board are ACIA.H, MEMORY.H, STRINGS.H

Standard PC Serial Card (SERIAL.H)

Standard Hercs/Monochrome card (HERCS.H, PRINTER.H)

32bit Digital I/O card, plus 8 channel A/D (0808 chip). The mapping arrangment of this board is,

Port 220h	A/D Channel register
Port 221h	A/D Data register
222h	A/D End Of Conversion signal
223h	Simple latched I/O
224h	8255 PPI Port A
225h	8255 PPI Port B
226h	8255 PPI Port C
227h	Intel 8255 PPI control register

In conjunction with this board, a panel comprising,
1 x 7 segment display
1 x 8 LEDS
1 x 8 Digital Switches
1 x 16 hexidecimal keypad
is used to allow students to write software routines. (DIOBOARD.H, MCRDRV.H) A magnetic card swipe reader attaches to port C of the DIOBoard.

The DIOBoards address is configurable via DIP, and the A/D convertor can be:
- strapped for auto restart, EOC signal restarts ADC
- software polled via EOC bit status
- Interrupt driven, by staps which connect EOC to IRQ2 to IRQ7

The POD SOFTWARE is a simple board MC6821 which has a 40PIN DIP which plugs across the 6809 target processor. By writing to the MC6821 (which connects to DIOBoard) and setting certain pins, it is possible to alter the processor state (ie, RESET, HALT) and perform READ/WRITE cycles (by emulating bus cycles asserting pins mapped to processors pins in correct sequence).

All the available items are available from us at reasonable cost, eg, US funds. We are also currently working on MC68000 and i8051 processor boards.

IF YOU RE-COMPILE THE SOURCE, USE A MEDIUM MEMORY MODEL!! AND TURN OFF WARNINGS

****

* Please note these disks have many more files than listed. ***
* Only shown are samples of the data to indicate content. **
****

Special Feature

Intermediate Users

Part 4

MOVING FORTH

by Brad Rodriguez

Part 4: Assemble or Metacompile?

"Keep it SHORT!" was the editorial directive for this installment. So I apologize for postponing the source listings to yet another issue. In the meantime, there is a new decision to contemplate:

How do you build a Forth system for the Very First Time?

You know now that most Forth code is high-level "threads," usually compiled as just a series of addresses. In the early days of fig-Forth, assemblers were often the only programming tools available. This was fine for writing Forth CODE words, but high-level threads had to be written as a series of DW directives. For example, the Forth word

```
: MAX ( n n - n)  OVER OVER < IF SWAP THEN DROP ;
```

would be written [TAL80]

```
        DW OVER,OVER,LESS,ZBRAN
        DW MAX2-$
        DW SWAP
MAX2:   DW DROP,SEMIS
```

Later, as working Forth systems became widespread, Forthwrights began modifying the Forth compilers into cross-compilers [CAS80]. Thus with Forth on your CP/M machine (or Apple II, or whatever), you could write Forth programs for some other CPU...up to and including an entirely new Forth system for that CPU.

Because they create a new Forth from within Forth, these are often called metacompilers. Computer science purists object to this, so some Forthies use the terms "cross-compiler" and "recompiler." The difference is that a recompiler can only generate a new Forth for the same CPU.

Most PC Forths are now produced with metacompilers, but opinion is divided in the embedded systems arena [TIN91,ROD91,SER91]. The arguments for using assemblers to write Forth are:

1. Metacompilers are cryptic and hard to understand, and you must thoroughly understand a metacompiler in order to use it.
2. Assemblers are understood by the average programmer.
3. An assembler is almost always available for a new CPU.
4. Assemblers handle many optimizations (e.g. short vs. long branch).
5. Assemblers handle forward references and peculiar address modes; many metacompilers don't.
6. Assemblers use familiar editing and debugging tools.
7. The code generation is completely visible -- nothing is "hidden" from the programmer.
8. It's easier to tweak the Forth model, since many design decisions affect the internals of a metacompiler.

The arguments for metacompilers:

1. You write "normal" looking Forth code, which is easier to read and debug.
2. Once you understand your metacompiler, you can port it easily to new CPUs.
3. The only tool you need to acquire is a Forth for your computer.

The last is particularly applicable to those who don't own PCs, since most cross-assemblers require PCs or workstations these days.

I've written several Forths each way, so I'm painfully aware of the tradeoffs. I admit a preference for metacompilers: I find the Forth code for MAX much easier to read and understand than its assembler equivalent. Most of the arguments against metacompilers have been overcome by modern "professional" compilers, and if you're using Forth for work I strongly recommend investing in a commercial product. Alas, public-domain metacompilers (including my own) are still behind the times, clunky, and arcane.

So I'm going to take a radical position for a Forth programmer, and tell you to choose for yourself. I'll publish the 6809 code in metacompiler form, and I'll supply a metacompiler for F83 (IBM PC, CP/M, or Atari ST) [ROD92]. The Z80 code will be written for a CP/M assembler. The 8051 code will be written for a public-domain PC cross-assembler.

Forth in C?

No discussion of this topic would be complete without mentioning a new trend: Forths written in C. These have the advantage of being more portable than assembler -- in theory, all you have to do is recompile the same source code for any CPU. The disadvantages:

1. Less flexibility in the design decisions; e.g., direct-threaded-code is probably not possible, and you can't optimize register assignments.
2. You have to recompile the C source to add new primitives.
3. Forth words carry the C call-and-return overhead.
4. Some C Forths use inefficient threading techniques, e.g. a CASE statement.
5. Most C compilers produce less efficient code than a good assembly-language programmer.

But for Unix systems and RISC workstations, which frown upon assembler, this may be the only way to get a Forth up and running. The most complete and widely used of the public-domain C Forths is TILE (TILE_21.ZIP, file #2263 on GEnie's Forth Roundtable). If you're not running Unix, you should look instead at the Genie files HENCE4TH_1.2.A (#2490) and CFORTHU.ARC (#2079).

To continue the previous comparison, here's the definition of MAX from HENCE4TH [MIS90]. I omit the dictionary headers for clarity:

```
_max() {
OVER  OVER  LESS IF  SWAP  ENDIF  DROP }
```

Instead of assembler, C is used to write the CODE words in the kernel. For example, here is HENCE4TH's SWAP:

```
_swap() {
register cell i = *(dsp);
*(dsp) = *(dsp + 1);
*(dsp + 1) = i;
}
```

(Please note: there is quite a variety of techniques for writing Forth words in C, so these words may appear radically different in CFORTH or TILE.)

On a 68000 or SPARC, this might produce quite good code. On a Z80 or 8051, quite the opposite. But even if you plan to write a Forth in C, you need to understand how Forth works in assembler. So stay tuned for the next installment of Moving Forth!

REFERENCES

[CAS80] Cassady, John J.,METAFORTH: A Metacompiler for Fig-Forth, Forth Interest Group (1980).

[MIS90] HenceFORTH in C, Version 1.2, distributed by The Missing Link, 975 East Ave. Suite 112, Chico, CA 95926, USA (1990). This is a shareware product available from the GEnie Forth Roundtable.

[ROD91] Rodriguez, B.J., letter to the editor, Forth Dimensions XIII:3 (Sep/Oct 1991), p.5.

[ROD92] Rodriguez, B.J., ''Principles of Metacompilation,'' Forth Dimensions XIV:3 (Sep/Oct 1992), XIV:4 (Nov/Dec 1992), and XIV:5 (Jan/Feb 1993). Note that the published code is for a fig-Forth variant and not F83. The F83 version is on GEnie as CHROMIUM.ZIP

[SER91] Sergeant, Frank, ''Metacompilation Made Easy,'' Forth Dimensions XII:6 (Mar/Apr 1991).

[TAL80] Talbot, R.J., fig-Forth for 6809, Forth Interest Group, P.O. Box 2154, Oakland, CA 94621 (1980).

[TIN91] Ting, C.H., ''How Metacompilation Stops the Growth Rate of Forth Programmers,'' Forth Dimensions XIII:1 (May/Jun 1991), p.17.

THE SCROUNGEMASTER II

Interested in a printed-circuit board for the 6809 multiprocessor? I've been exchanging email with TCJ reader Andrew Houghton, who wants to build a ''Poor Man's Transputer'' out of the 6809. This prompted some improvement and expansion of the original ScroungeMaster I design. Namely:

a) four RS-485 serial ports using two Z8530s, instead of the 2681
b) two parallel I/O ports, using a 6522
c) memory mapping logic for expanded memory: 32K on board PROM, 32K or 128K on-board RAM, and 384K of off-board (bus) address space

The driving principle is still Cheap Parts: I figure the increased cost of ICs over the ScroungeMaster I is about $9 (Jameco prices). No PALs.

Wire-wrapping one of these boards is enough of a headache -- I'd like to avoid wire-wrapping three more. If we can get commitments for twenty boards, I'll do the PCB layout and get boards fabricated at a local vendor. If interested, send me GEnie mail (B.RODRIGUEZ2), Internet email (b.rodriguez2@genie.geis.com), or drop a postcard to Brad Rodriguez, Box 77, McMaster University, 1280 Main Street West, Hamilton, Ontario L8S 1C0 Canada.

Rate Card

TCJ The Computer Journal

P. O. Box 535, Lincoln, CA 95648-0535
(800) 424-8825 or (916) 645-1670

Mechanical Requirments:

Size	Vertical Format	Horizontal Format
Full Page	7 1/8"H x 9 5/8"V	
1/2 Page	3 1/2"H x 9 5/8"V	7 1/8"H x 4 5/8"V
1/3 Page	2 1/8"H x 9 5/8"V	7 1/8"H x 3"V
1/4 Page	3 3/8"H x 4 5/8"V	
Market Place	2"H x 3"V	

All mechanicals submitted for publication must meet the production standards of the journal. The preferred form is line shot positives such as PMT, negatives right reading emulsion side down or laser printer output. Mechanicals composed with dot matrix or daisy wheel printers are not of acceptable quality. Pasted up pages must be done using HOT WAX or other non-visible means, taped art work is not acceptable. Please contact us regarding ads which do not fall within these specifications.

Advertising Rates:

Size	1 Insertion	2-3 Insertion	4+Insertions
Full	$120	$100	$90
1/2 Page	$75	$65	$60
1/3 Page	$60	$50	$45
1/4 Page	$50	$45	$40
Market Place	$25	$20	$100 for 6 issues

Color Charges: $250 per color other than black or $750 for full color (4C) per insertion. Color separations must be supplied with order. Separation service available at extra cost.

Placement: Advertising position on text pages placed as page layout permits and is not guaranteed. We will make all attempts to honor special requests. Placements on covers guaranteed at time contract is accepted by publisher. Publisher reserves right to change ad costs. Advertising contracted before rate changes will be carried at previously agreed rates. Rates are for camera ready copy. Inquire for typesetting and composition rates.

Magazine Specifications: The Computer Journal is a 48 page saddle stitched book plus cover for a total of 52 pages. Text paper is 60 pound high brilliance offset. Cover paper is 60 pound matte text stock. The magazine is printed offset. Trim size is 8 3/8 inches by 10 7/8 inches with no bleeds. Suggested ruling is 100 to 133 lines per inch and tints above 85% may print as solids.

Production Schedule:

Issue Date	Space Reservation	Copy Required	Printing Date	Mailing Date
Jan/Feb	1 Nov	15 Nov	15 Dec	1 Jan
Mar/Apr	1 Jan	15 Jan	15 Feb	1 Mar
May/Jun	1 Mar	15 Mar	15 Apr	1 May
Jul/Aug	1 May	15 May	15 Jun	1 Jul
Sep/Oct	1 Jul	15 Jul	15 Aug	1 Sep
Nov/Dec	1 Sep	15 Sep	15 Oct	1 Nov

Payment must accompany space reservation for advertisements and is not

TCJ CLASSIFIED

·**WANTED:** info. on the BRD Inc. computers DOLPHIN-78 or PORPOISE-78, a.k.a. FDC-78 (6800 based machines). Also, I need an Apple IIe emulation disk for my 256K Apple III (I have the Aplle II+ emulation disk). Roger Olson, 2304 West 4th, North Platte NE 69101.

For Sale:
KayPro 4-all stock as orginally made. System disk, WordStar 3.3, and some other software. Printer cable. No manuals available. Works fine! Asking $50

Kaypro 2X-Advent TurboRom and drive decoder board added; now has two HD drives (96TPI) and one DD drive (48TPI). Advent system disk, utilities, and docs-on-disk, WordStar 3.3, etc., manuals and one-piece cover. Runs great! Asking $75.

Kaypro 1 -(last model made) Advent TurboRom and drive decoder board added; has one DD (48TPI) drive and one HD (96TPI) drive; also Minnie Winnie 20 meg external drive (the ST-506 drive sold by Advanced Concepts and Engineering). Loads of software--running NZCOM, all the Z-system utilities, programming stuff, WordStar 4.0, 3.3, CalcStar, DataStar, and ReportStar, and more -- proably 7 to 10 megs of stuff. Manuals, one-piece cover, etc. included. A really nice machine! Asking $200.

Kaypro printer (Juki 6100 daisywheel) and Tractor Feed - with cable. Works fine! Asking $60.

Kaypro Technical manual - in 3-ring binder. $10.

Shipping extra, Contact: Dave Templin, 2978 Spruce Way, West Sacramento, CA 95691, (916) 371-2964.

For Sale: ***Collector's Guide to Personal Computers and Pocket Calculators.*** Prices and illustations included. 36 pages, $14.95 plus $2.00 shipping. Fred Hartfield, Box 52466, New Orleans, LA 70152. Digitial Cottage BBS (504) 897-5514, help, support, sales, of old systems.

Avaialble: Legal copies of CP/M, Borland Software (Turbo Pascal, DBase II, and more). Many bootable CP/M disks formats avaialble. Disk copying, most formats inculding Apple CP/M. Manuals and more! Lambda Software Publishing, 149 West Hilliard Lane, Eugene, OR 97404-3057, (503) 688-3563.

NEW CLASSIFIED RATES!

NOW $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.

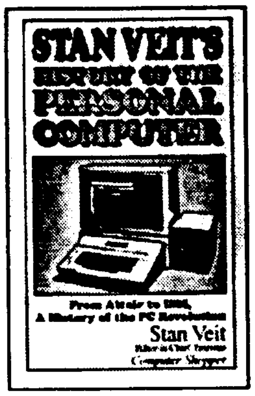
Send your items to:

The Computer Journal
P.O. Box 535
Lincoln, CA 95648-0535

NOW AVAILABLE!

"...a must addition to the library of every computer enthusiast. Highly recommended."

—John C. Dvorak, PC Magazine



Stan Veit's HISTORY OF THE PERSONAL COMPUTER

Order from: **WORLDCOMM®** **1-800-472-0438**
65 Macedonia Road, Alexander NC 28701
(Visa and MasterCard accepted)

Please Send:	**N.C. Residents Add 6% Sales Tax**
____**Hardback Copies of *History***	**@ $27.95 plus $3.00 S&H**
____**Paperback Copies of *History***	**@ $19.95 plus $3.00 S&H**

Name ______________________
Address ______________________
City __________ State____ Zip __________

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 25**
- 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 ISAM 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

Issue Number 26:
- 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

Issue Number 27:
- 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

Issue Number 28:
- 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

Issue Number 29:
- 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
- The ZCPR3 Corner

Issue Number 30:
- 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
- The ZCPR3 Corner
- The CP/M Corner

Issue Number 31:
- 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:
- Language Development: Automatic Generation of Parsers for Interactive Systems
- Designing Operating Systems: A ROM based OS for the Z81
- Advanced CP/M: Boosting Performance
- Systematic Elimination of MS-DOS Files: Part 1, Deleting Root Directories & an In-Depth Look at the FCB
- WordStar 4.0 on Generic MS-DOS Systems: Patching for ASCII Terminal Based Systems
- K-OS ONE and the SAGE: System Layout and Hardware Configuration
- The ZCPR3 Corner: NZCOM and ZCPR34

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: Operating system extensions to BDOS and BIOS, RSXs for CP/M 2.2.
- Macintosh Data File Conversion in Turbo Pascal.
- The Computer Corner

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.
- The Computer Corner

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 hardware for experimenter, CPUs in series, software options.
- SPRINT: A review.
- REL-Style Assembly Language for CP/M & ZSystems, part 2.
- Advanced CP/M: Environmental programming.
- The Computer Corner.

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 date variables.
- Resident Programs: A detailed look at TSRs & how they can lead to chaos.
- Advanced CP/M: Raw and cooked console I/O.
- Real Computing: The NS 32000.
- ZSDOS: Anatomy of an Operating System: Part 1.
- The Computer Corner.

Issue Number 38:
- C Math: Handling Dollars and Cents With C.
- Advanced CP/M: Batch Processing and a New ZEX.
- C Pointers, Arrays & Structures Made Easier: Part 2, Arrays.
- The Z-System Corner: Shells and ZEX, new 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 Hewlett Packard 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.
- The Computer Corner.

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
- The Computer Corner.

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.
- Programming disk and printer functions with C.
- LINKPRL: Making RSXes easy.
- SCOPY: Copying a series of unrelated files.
- The Computer Corner.

Issue Number 42:
- Dynamic Memory Allocation: Allocating memory at runtime with examples in Forth.
- Using BYE with NZCOM.
- C and the MS-DOS Screen Character Attributes.
- Forth Column: Lists and object oriented Forth.
- The Z-System Corner: Genie, BDS Z and Z-System Fundamentals.
- 68705 Embedded Controller Application: An example of a single-chip microcontroller application.
- Advanced CP/M: PluPerfect Writer and using BDS C with REL files.
- Real Computing: The NS 32000.
- The Computer Corner

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: Graphics routines for the IBM PC, and the Turbo C graphics 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.
- Real Computing: The NS32000.
- The Computer Corner.

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 format emulator for CP/M.
- Advanced CP/M: ZMATE and using lookup

and dispatch for passing parameters.
· Real Computing: The NS32000.
· Forth Column: Handling Strings.
· Z-System Corner: MEX and telecommunications.
+ The Computer Corner

Issue Number 45:
· Embedded Systems for the Tenderfoot: Getting started with the 8031.
· The 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.
· The Computer Corner.

Issue Number 46:
· Build a Long Distance Printer Driver. Using the 8031's built-in UART for serial communications.
· 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, Counter/Timers, 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 Multi-tasking & Distributed Systems
· Long Distance Printer Driver: correction
· ROBO-SOG 90
· The Computer Corner

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
· Real Computing
· Z-System Corner: Patching MEX-Plus and TheWord, Using ZEX
· Z-Best Software
· The Computer Corner

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

The Computer Journal Back Issues

· Controlling Home Heating & Lighting, Pt. 1
· Getting Started in Assembly Language
· LAN Basics
· PMATE/ZMATE Macros, Pt. 2
· Real Computing
· Z-System Corner
· Z-Best Software
· The Computer Corner

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
· The Computer Corner

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 as a Technique for Intelligent Real-Time Embedded Control
· Real Computing, the 32CG160, Swordfish, DOS Command Processor
· PMATE/ZMATE Macros
· Z-System Corner, The Trenton Festival
· Z-Best Software, the Z3HELP System
· The Computer Corner

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 Single-Board Computer
· The Computer Corner

Issue Number 53:
· The CPU280
· Local Area Networks
· Am Arbitrary Waveform Generator
· Real Computing
· Zed Fest '91
· Z-System Corner
· Getting Started in Assembly Language
· The NZCOM IOP
· Z-BEST Software
· The Computer Corner

Issue Number 54:
· Z-System Corner
· 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
· The Computer Corner

Issue Number 55:
· Fuzzilogy 101
· The Cyclic Redundancy Check in Forth
· The Internetwork Protocol (IP)
· Z-System Corner
· Hardware Heaven
· Real Computing
· Remapping Disk Drives through the Virtual BIOS
· The Bumbling Mathmatician
· YASMEM
· Z-BEST Software
· The Computer Corner

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

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

Issue Number 58:
· Multitasking Forth
· Computing Timer Values
· Affordable Development Tools
· Real Computing
· Z-System Corner
· Mr. Kaypro
· DR. S-100
· The Computer Corner

Issue Number 59:
· Moving Forth
· Center Fold IMSAI MPU-A
· Developing Forth Applications
· Real Computing
· Z-System Corner
· Mr. Kaypro Review
· DR. S-100
· The Computer Corner

Issue Number 60:
· Moving Forth Part II
· Center Fold IMSAI CPA
· Four for Forth
· Real Computing
· Debugging Forth
· Support Groups for Classics
· Z-System Corner
· Mr. Kaypro Review
· DR. S-100
· The Computer Corner

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

Issue Number 62:
· SCSI EPROM Programmer
· Center Fold XEROX 820
· DR S-100
· Real Computing
· Moving Forth part III
· Z-System Corner
· Programming the 6526 CIA
· Reminiscing and Musings
· Modem Scripts
· The Computer Corner

Issue Number 62:
· SCSI EPROM Programmer part II
· Center Fold XEROX 820
· DR S-100
· Real Computing
· Multiprocessing Part II
· Z-System Corner
· 6809 Operating Systems
· Reminiscing and Musings
· IDE Drives Part II
· The Computer Corner

SPECIAL DISCOUNT
15% on cost of Back Issues when buying from 1 to Current Issue.
10% on cost of Back Issues when buying 20 or more issues.

	U.S.	Canada/Mexico		Europe/Other	
Subscriptions (CA not taxable)		(Surface)	(Air)	(Surface)	(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
Software Disks (CA tax)	add these shipping costs for each 3 disks ordered				
MicroC Disks are $6.00ea	+$1.00	+$1.00	+$1.25	+$1.50	+$2.50

Items: ____________________ Back Issues Total ________

____________________ MicroC Disks Total ________

California state Residents add 7.25% Sales TAX ________

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 535, Lincoln, CA 95648-0535
Phone (916) 645-1670

The Computer Journal - Micro Cornucopia Kaypro Disks

K1
MODEM PROGRAMS

K2
CP/M UTILITIES

K3
GAMES

K4
ADVENTURE

K5
MX80/GEM 10X GRAPHICS

K6
TEXT UTILITIES

K7
SMALL C VER 2

K8
SOURCE OF SMALL C

K9
GENERAL UTILITIES

K10
Z80 AND LINKING ASSEM

K11
CHECKBOOK PROGRAM &
LIBRARY UTILITIES

K12
KAYPRO FORTH

K13
SOURCE OF FIG-FORTH

K14
SMARTMODEM PROGRAMS

K15
HARD DISK UTILITIES

K16
PASCAL COMPILER

K17
Z80 TOOLS

K18
SYSTEM DIAGNOSTICS

K19
PROWRITER GRAPHICS

K20
MICROSHERE'S COLOR
GRAPHICS BOARD

K21
SBASIC & SCREEN DUMP

K22
ZCPR

K23
FAST TERMINAL &
RCPM UTILITIES

K24
KEYBOARD TRANSLATOR &
MBASIC GAMES

K25
Z80 MACRO ASSEMBLER

K26
EPROM PROGRAMMER/TOOLS

K27
TYPING TUTORIAL

K28
MODEM 730 SOURCE

K29
TURBO PASCAL GAMES I

K30
TURBO PASCAL GAMES II

K31
TURBO BULLETIN BOARD

K32
FORTH-83

K33
UTILITIES

K34
GAMES

K35
SMALL C VER 2.1

K36
SMALL C LIBRARY

K37
UTILITIES PRIMER

K38
PASCAL RUNOFF WINNERS
FIRST - THIRD

K39
PASCAL RUNOFF WINNERS
FORTH & FIFTH

K40
PASCAL RUNOFF WINNERS
SIXTH PLACE

K41
EXPRESS 1.01 TEXT EDIT

K42
PASCAL RUNOFF-GRAPHICS

K43
PASCAL RUNOFF-GAMES

K44
PASCAL RUNOFF-PRINTERS

K45
PASCAL RUNOFF-UTILITIES

K46
PASCAL RUNOFF-TURBO UTILS

K47
256K RAM SOFTWARE

K48
C CONTEST WINNERS I

K49
C CONTEST WINNERS II

TCJ The Computer Journal

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

Micro C Disks are $6.00 each plus shipping costs.

Shipping Cost to	U.S.	Canada/Mexico		Europe/Other	
		Surface	Air	Surface	Air
Added these costs	$1.00	$1.00	$1.25	$1.50	$2.50

Shipping costs are for GROUPS of 1 to 3 disks.

Computer Corner

By Bill Kibler

Regular Feature

Editorial Comment

History and Changes

I have three items for this corner and for a change some space to cover the items in. Thanks Brad for keeping it shorter! It really has been nice that *TCJ*'s writers have been so prolific these days, but unfortunately it meant I (or my column) suffered. Lets see if I can make up for it.

Stan Veit

Several weeks ago I received a call from Stan Veit. For those who don't know Stan, he was instrumental in taking *Computer Shopper* from a small classified monthly and making it the worlds largest computer and advertising magazine. I have since found out, Stan's background is considerably more involved with classic computers than that.

It seems Stan was the First owner/manager of a computer store in New York city. Since only one other store existed in Los Angeles at the time, these two owners had lots of first hand knowledge about the computer industries starting days, which they shared with each other.

Stan wrote about these early days while being editor of *Computer Shopper*. Those columns are now in a book called, "Stan Veit's History of The Personal Computer", published by WorldComm Press, 65 Macedonia Rd., Alexander, NC 28701, (704) 252-9515.

My only complaint about the book has to do with the original column format. My writers should all get this book, as it shows a good column writing style and Stan does a good job of making sure all the facts are repeated every issue as needed. That repeating however gets a bit annoying in the book. I would rather Stan had had the time to re-write some it and take the sections that repeat previous discussions and expand them or include them in previous sections. This is something I will be keeping in mind if we ever publish *TCJ*'s columns in a book.

Stan's personal involvement comes across excellently in the book. He is a good writer and his experiences are well illustrated in the book in both words and pictures. I found the book a great source of which companies made which systems. His personal introductions of people that are now very famous gives you an insight into their humble beginning. Steve Jobs getting his jeans sewed up by Stan's mother-in-law is but one sample of insight and humor combined.

For anyone who is serious about collecting and using older systems, the book is an absolute must. My only hope is that Stan decides to write more about those early days. Since each chapter is one columns worth, I am sure he had lots more to say about each company and their products. What about it Stan, a book for each major product? I can see it now, Stan Veit's history of the S-100 computers. Or.."Apple: from torn jeans to three piece suits, in one year!"

Changes At *TCJ*!

If you are subscribers of *Computer Craft*, you will by now know that they are changing their name to be more like our's (*MicroComputer Journal*). They also are dropping any support for older PC/XT platforms. *The Computer Journal*'s readers are mostly eight bit system users, but many of you have been a bit upset that we have not supported the early PC clone models. I have stated since taking over that I felt others were providing support for the PC/XT platforms.

Since *Computer Craft* is no longer supporting them, and their used price is now below $100 for used mother boards, I feel my previous position is no longer valid. My main concern, and I am sure the concern of most readers, is that *TCJ* does not go the way of other magazines and start phasing out all other system support. To make sure that doesn't happen, I am selecting one author to provide PC/XT support.

Out of the ten *TCJ* regular writing staff, having one provide support is allocating only 10% to the cause. That leaves all the other writers to cover their regular beats. This will also make our regulars feel better when they provide not only their normal support but show how it relates to the PC platforms as well.

I do not have a commitment from Frank Sergeant, but I am trying to get him as our first PC support person. Frank certainly has the credentials, wants to help the magazine, and is well known by most of you for his excellent program PYGMY Forth. Some ideas I plan on explaining to Frank, are: porting PYGMY to ROM on a PC; building embedded controllers using PC/XT boards; serial communications on a PC/XT using the keyboard port; and 6809 (as in COCO design) or Z180/ZCPR expansion boards.

His first assignments if he agrees to the task at hand, will be explaining the PC/XT architecture for those who have not been previously indoctrinated. I dare say that many users have little true understanding of the PC/XT hardware designs, and thus don't know why so many of us at *TCJ* hate the hardware with a passion. I have spent the last three years working with 68000 boards that reside as expansion boards on PC based platforms. The 68000 side has always been easy and straight forward, the PC side anything but, which another way of saying Frank will have his work cut out for him.

This move should also help our readers find platforms for new projects. The flood of passed over PC/XT designs and expansion products is becoming enormous. Vendors are starting to practically give them away. They are looking for places to advertise not only these older PC items, but a few old Z80 or 6809 based items as well. With our past position on no official support, these advertiser were passing us by. No more!

New Advertising Strategy

The Computer Journal has been doing well for the last year. Although not running in the red, we are not making money either. We need advertisers to make the difference. Our readers are also complaining constantly that they can not find support or products to meet their needs. Our advertising rates were reflective of days gone by. All this has come to indicate time is right to lower our rates. And lower we have.

Effective with this issue, we are dropping back our rates in hopes that many of the mom-pop type vendors can start advertising and providing the reader with sources to meet their needs. How much of a drop, does 70% DROP sound about right. See page 50 for a complete revised rate card.

The major idea with this reduction is to give our readers names and address of vendors who have products they are looking for. David Klink's letter in our Reader To Reader section is typical of our reader's problems. David would like to build something using the new Motorola 68306 but is at a loss to fine one. I could call our local Motorola rep and get one, but one doesn't help our readers.

Since spending so much of my time doing the magazine, I have lost contact with many vendors that provided small quantity sources of products. Hopefully they will see the changes in our rates and reconsider about advertising with *TCJ*.

WHAT NEXT?

Now that *The Computer Journal* is moving to PC based platforms, I guess I can admit that I am working on OS/2 doing C programming. The move was somewhat of my employers choice, but I did agree to it. The upshot of these changes is that I have become familiar with OS/2 in considerably more detail than most. Our company has had a single product on OS/2 for some time now. It has had poor acceptance by our users. The programmers have had more problems with their tools than they want to think about. It is getting better however.

The biggest surprise I have had is in finding books to support OS/2. There are so little it is amazing the product is being used at all. In one major book store chain, the Windows section is many feet in length. The OS/2 books however, fill about 10 to 12 inches of space. The new Windows NT section is already longer than OS/2 and the product has only been out a few weeks.

I must admit to finding a few good books and will discuss them next time. I think there are two reasons for the shortage, one is just lack of OS/2 sales, the other is OS/2's compatibility with DOS. So much of how OS/2 works is like PCDOS and thus many users have not decided to look deeper into the system. For programmers, IBM's own series of RED BOOKS is quite good and complemented by ON-LINE support, such that using outside books may be a thing of the past.

The most disturbing thing is actually stepping back in time again. The CP/M world has many useful utilities that have never made it to the MSDOS world. It has always amazed me how users can talk about how great MSDOS is yet the advanced tools and utilities that I came to love in the CP/M world have no equivalent. The worse part is the users have little understanding of their loss, but instead just work around the problems or make their own crude tools to do the job.

Well for new users to OS/2 welcome back to the dark ages. Some MSDOS tools have been ported, but in many cases only half heartedly. As time moves on I hope this situation will change. OS/2 2.1 is a very impressive platform and as such deserves better press and support. I hope to explain that position next time.

Now when talking about programming, I expanded a simple main.c file with all the listing options turned on. That means all the included and header files would be printed out in the listing. Well my 35K source file became 6,750K long! That means over 6 megs of data was included during the assembly process. Try assembling one of these OS/2 C modules on a small memory machine. Our machines now have 16 Megs of memory and 350 MByte hard drives as minimum setup.

Speaking of drives, it will take at least 100 megs of hard drive for the system and a few programs. The manual says you can squeeze it onto a 60 meg drive, but don't try. Now I do understand that Windows NT is about twice as big as OS/2, so if you really want to wing it up to the big time, up might go an NT it...

The main problem you will face if you try a small disk, is swap space. The OS/2 system uses virtual memory, which means programs (many programs) can share the same memory space. The system just moves parts of the program out onto disk when not being used. Like when I was compling the 6 megs. I assume that at least half of that ended up temporarily on disk

Oh well more next time....and...

Welcome to the ever improving *The Computer Journal*. Have fun hacking....

Market Place

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
(503) 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 535
Lincoln, CA 95648-0535

CP/M SOFTWARE

100 page Public Domain Catalog, $8.50 plus $1.50 shipping and handling. New Digital Research CP/M 2.2 manual, $19.95 plus $3.00 shipping and handling. Also, MS/PC-DOS Software. Disk Copying, including AMSTRAD. Send self addressed, stamped envelope for free Flyer, Catalog $1.00

Elliam Associates
Box 2664
Atascadero, CA 93423
805-466-8440

NEW MAGAZINE
the world of 68' micros
supporting
Tandy Color Computer
OS-9 & OSK

$23/year for 8 issues
$30/year Canada/Mexico
$35/year overseas

Published by:
FARNA Systems
P.O. Box 321
Warner Robins
GA 31099-0321

S-100/IEEE-696

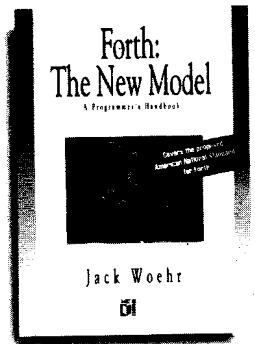
IMSAI Altair
Compupro Morrow
Cromemco
and more!

Cards • Docs • Systems

Dr. S-100

Herb Johnson,
CN 5256 #105,
Princeton, NJ 08543
(609) 771-1503

New from
M&T Books!



$44.95 1-55851-277-2

M&T BOOKS
Technical Books for Technical Times

Available at bookstores everywhere
or call 1-800-688-3987
RCJ3

PCB's in Minutes From LaserPrint!*

8 1/2" x 11" Sheets
100% MBG

* Or Photocopier
Use household iron to apply.

PnP BLUE or	**PnP WET**
For High Precision Professional PCB Layouts	Easy Hobby Quality PCB's
1. LaserPrint	1. LaserPrint
2. Iron-On	2. Iron-On
3. Peel-Off	3. Soak-Off w/ Water
4. Etch	4. Etch
An Extra Layer of Resist for Super Fine Traces	Transfers Laser or Copier Toner as Resist

20Sh$30/40Sh$50/100Sh$100 Blue/Wet (No Mix)
Sample Pack 5 Shts Blue + 5 Shts Wet $20
VISA/MC/PO/CK/MO $4 S&H -- 2nd Day Mail
Techniks Inc. P.O. Box 463 Ringoes NJ 08551
(908)788-8249