The COMPUTER JOURNAL

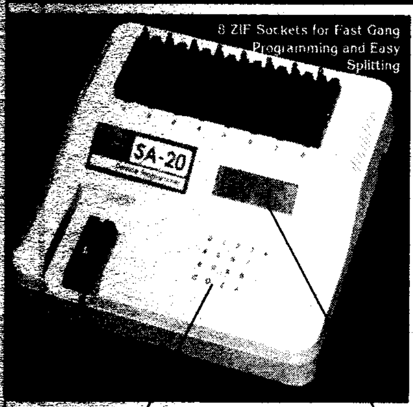
Programming - User Support
Applications

Issue Number 59 January/February 1993 US$4.00

ISSN # 0748-9331

EPROM PROGRAMMERS

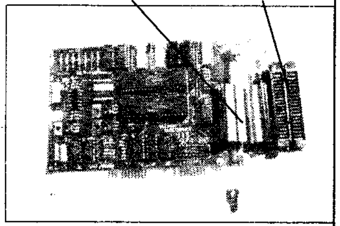
Stand Alone Gang Programmer — $750.00


8 ZIF Sockets for Fast Gang Programming and Easy Splitting
SA-20
20 Key Tactile Keypad (not membrane)
20 x 4 Line LCD Display

- Completely stand-alone or PC driven
- Programs E(E)PROMs
- ***1 Megabit of DRAM***
- ***User upgradable to 32 Megabit***
- ***.3/.6" ZIF socket, RS-232, Parallel In and Out***
- 32K internal Flash EEPROM for easy firmware upgrades
- ***Quick Pulse Algorithm (27256 in 5 sec, 1 Megabit in 17 sec.)***
- 2 year warranty
- Made in U.S.A.
- Technical support by phone
- Complete manual and schematic
- ***Single Socket Programmer also available. $550.00***
- Split and Shuffle 16 & 32 bit
- 100 User Definable Macros, 10 User Definable Configurations
- Intelligent Identifier
- Binary, Intel Hex, and Motorola S

Internal Programmer for PC — $139.95

New Intelligent Averaging Algorithm. Programs 64A in 10 sec., 256 in 1 min., 1 Meg (27010, 011) in 2 min. 45 sec., 2 Meg (27C2001) in 5 min. Internal card with external 40 pin ZIF.



- Reads, verifies, and programs 2716, 32, 32A, 64, 64A, 128, 128A, 256, 512, 513, 010, 011, 301, 27C2001, MCM 68764, 2532
- ***Automatically sets programming voltage***
- Load and save buffer to disk
- Binary, Intel Hex, and Motorola S formats
- ***Upgradable to 32 Meg EPROMs***
- ***No personality modules required***
- 1 year warranty • 10 day money back guarantee
- Adapters available for 8748, 49, 51, 751, 52, 55, TMS 7742, 27210, 57C1024, and memory cards
- Made in U.S.A.

NEEDHAM'S ELECTRONICS
4539 Orange Grove Ave. • Sacramento, CA 95841
Mon. - Fri. 8am - 5pm PST

C.O.D.

Call for more information
(916) 924-8037
FAX (916) 972-9960

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

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

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

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

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

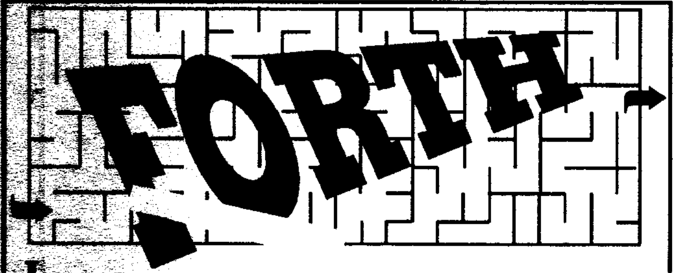
Quality Solutions
PseudoCorp has been providing quality solutions for microprocessor problems since 1985.

BROAD RANGE OF SUPPORT
- Currently we support the following microprocessor families (with more in development):

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

- All products require an IBM PC or compatible.

So What Are You Waiting For? Call us:
PseudoCorp
Professional Development Products Group
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 $49 each)
XBIOS for SB180 ($50)
ZMATE text editor ($50)
BDS C for Z-system (only $60)
DSD: Dynamic Screen Debugger ($50)
PCED: ARUNZ and LSH for MSDOS ($50)
ZMAC macro-assembler ($50, $70 with printed manual)
Order by phone, mail, or modem and use
Check, VISA, or MasterCard.

Z-System public domain software by mail.
Regular Subscription Service
Z3COM Package of over 1.5 MB of COM files
Z3HELP Package with over 1.3 MB of online documentation
Z-SUS Programmers Pack, 8 disks full
Z-SUS Word Processing Toolkit
And More!

For catalog on disk, send $2.00 ($4.00 outside North America) and your computer format to:

Sage Microsystems East
1435 Centre Street
Newton Centre MA 02159-2469
(617) 965-3552 (voice 9 to 11AM)
(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
Matt Mercaldo
Tim McDonough
Frank Sergeant
Clem Pepper
Richard Rodman
Jay Sage

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.

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

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 59 January/February 1993

EDITOR'S COMMENTS

Welcome to 1993 and the next issue of *The Computer Journal*. I have been very busy on the magazine as usual. I think the direction and minor changes are starting to produce good results for myself and you the reader. Our writers have been grinding out plenty to read and ponder. I fear that some have even gone overboard with their efforts.

Brad Rodriguez has part one of moving Forth to other CPUs. This has been a hot topic among readers for some time and I think Brad will lay to rest all your questions on the topic. An on going problem for many new users of Forth has been making turnkey programs. Frank Sergeant shows you how to build your own application in Forth and make it into a turnkey application.

TCJ's regular writers are here with their usual words of wisdom, or at least some good comments on their favorite topics. Rick Rodman fills in some gaps I had about the MINIX file system while showing you how he recovered from the dreaded curse.

Charles Stafford helps all us novice Kaypro users decide which machine we really have. I for one am looking forward to future articles of his, now that I know which version I want to buy at the next swap meet. Speaking of swap meets, Herb Johnson reviews a number of S-100 vendors and comments on the type and quality of cards they produced. With all this information your next trip swapping should make sure you buy good classic systems and leave the junk for the naive shopper.

Our ever popular and resident guru of the ZCPR world, Jay Sage explains some fancy macro-ing and gives us some insights to why he uses ZMATE for everything. Here we all just thought ZMATE was a text editor, little did we know what power it really has. Thanks Jay for the peek in the door of your favorite program.

TCJ CENTER FOLD?

Yes, *TCJ* has entered the big time with a center fold section. Well it may not be what first comes to your mind, but then if you are truly a hardware hacker at heart and a lover of old machines, our first center fold should warm you up some. It is a schematic of the original MPU-A CPU card from IMSAI.

This first item should help all you would be hardware people by bringing your collection of schematics up to date. That first date is 1975 however. Yup, this is an 18 year old CPU card. For those new people, life in the micro computing world started before PC Clones by many years. I have used these cards in the past and the schematic actually shows some corrections I made to resolve a problem.

Check out the front and back pages of the center fold, as they contain a description of the card. There is also some request for suggestions of schematics you would like to see, but then you can read all that yourself in our new section, the Center Fold.

I'M LATE, I'M LATE...

If you haven't notice this issue is a bit late. I took my first vacation in two years over Christmas and have been trying to catch up ever since. The problem was also compounded by many of my writers waiting till after Christmas to send their articles and not before as had been planned. Oh well, when things seem to go wrong, everything goes wrong.

Issue number 60 should be closer to on time, but then what is on time around here. I try and get to the printer by the first week of the month listed (by Jan 10 for Jan/Feb issue). That should put it in the mail by the 20 to 25 of January. With those dates now laid out you can see I am about 4 weeks behind. I should be able to gain one or two weeks each issue. In short, 62 or 63 may be on time (if the printing press doesn't break AGAIN).

No matter what delays happen, you should find the ever increasing pages of *TCJ* to your liking. If not, just write to the editor (oh, that's me...) and I will print what you think. Thanks and Enjoy, Bill Kibler.

READER to READER

Letters to the Editor

All Readers

MINI Articles

7 September 1992

Hello Bill,

When reading the 'new' TCJ issue #56 (I received it on 3 September since it came with surface mail), I found some things to criticize and some room for improvements. So I will just list them up now (don't matter which order), and hope to get your reply somehow.

1. Your article 'The Next Ten Years'

I agree that sometimes going 'back to the roots' is necessary and good. But in some details, I cannot agree to your method of how to do that. Of course the know-how presented in the articles should be as portable as possible, but reducing all hardware projects to 'serial' projects or all software ideas to forth listings won't be the right way (and additionally is far too extreme). If you would indeed publish only (or almost only) 'serial hardware' and forth software, the magazine would lose very much (contents and readers).

BTW, even if Forth as the language is very portable (and available for almost any computer), I don't think that it is ideal for demonstrating the how-to of programming. Instead, structured languages as Pascal or Modula-2 are far more dedicated to that purpose, since they nearly exactly follow flowchart diagrams, and explain themselves with their clear english commands. Additionally, there will be far more people being able to read Pascal or Modula sources of a strange author than there are with forth. (I also have some difficulties when trying to read a forth program.) So I think it is best to keep your eye on explaining algorithms (with flowcharts and/or Pascal/Modula sources), and not on strictly reducing every program to forth.

Another thing that should be discussed is your statement about hardware and the components to use. When proclaiming TTL-only circuits, you should not forget that these normally need much times the board space and the power of the comparably higher integrated circuits. Many of the newer designs would have been impossible without using modern VLSI components. With arguing even against PALs, you throw people back to the times where only TTL's were available. I think that one should freely use those components which are established and easily to get. Otherwise, you must also forbid using EPROMs (using diode matrices instead) or the good old Z80 (far too much integrated - replace it with TTL!). The EPROMs are a very good comparison to GALs. Since many people have EPROM programmers, you freely use those parts. Since more and more people are able to program GALs, we should use them too (but of course reduce it to few different types).

I think I don't have to write much more about 'serial' hardware. Of course, sometimes it makes much sense (and provides portability). But you should not forget the large and very interesting sector of peripherals which need some bus connection, i.e. my IDE-Interface. Although this one uses the ECB bus, the circuit idea is portable to any other 8-bit bus (but not to a serial port).

One last sentence about 'going back to the roots'. If you write everything only for the absolute ('bloody') beginner, be aware that the more experienced readers will cancel their subscription, since they won't find anything new. I think that TCJ articles should be written for different readers with different skills, just as it is until now. BTW, some of the beginners will understand more if they look in the 'old' issues later.

2. Missing information on how to reach someone

In former TCJ issues, there was a short paragraph at each article giving some information about the author and how to reach him. This information is missing in the last issue. So this is why I have to forward this message via Jay Sage, and ask you for forwarding some details to some other authors.

3. Something to forward to Tim McDonough

That was a very nice article about input expansion. Perhaps one should have noticed that output expansion is possible exactly the same way, just with other shift registers. Concerning the software, the READ8 routine could be much easier:

```
READ8:   MOV  R0,#8      ; counter
READ81:  SETB  CLOCK
         CLR   CLOCK     ; pulse clock
         MOV  C,DATA   ; move data into carry
         RLC    A         ; shift into ACC
         DJNZ  R0,READ8 ; loop for all 8 bits
         RET          ; return with data in ACC
```

When the 8031 has such a wonderful Boolean processor, why not use it? This speeds up the routine somewhat (and since serial input is quite slow compared to parallel ports, reduces this difference). I didn't look at his software in detail, I just saw that routine when moving on to the next page.

4. Your comment on my article

I read the complete TCJ very carefully but did not find anything that would give your comment some sense. What about the Novix system and your problems connecting an IDE hard disk to it?

5. New IDE-Interface version available

Meanwhile, I improved the IDE interface circuit and PCB layout due to some critical timing details. The new version is available now. (There are no boards of the old version ever come into the USA.)

6. Real Computing

When reading Rick's article this time, I hardly missed any statement about Coherent (actual version 4.0), which seems to be a very good OS for the UNIX beginner. The rest of the article is very informative and interesting. I hope that it will be continued (and will offer some information about Coherent).

7. Article Jumping

I am very surprised! Some time ago I wrote to Chris McEwen about the article-hopping within TCJ. I pointed out that the articles should be printed just one after another, to avoid that unnecessary harassment. He just answered, that's american typesetting and not european. But now I find that the 'new' TCJ has its articles just one after another. VERY GREAT ADVANTAGE!

8. Subscription Expiry

When I wrote my last article for TCJ (about the CPU280), Chris gave me some kind of free subscription for a limited time instead of money. I agreed to that, since it simplifies everything (we don't have to transfer money overseas two times). Unfortunately, the subscription expiry printed on the address label did not change with my new article. I think that is caused by the changes and you will surely correct it with the next issue.

9. Another article in TCJ ?

Meanwhile, the text and graphics terminal mentioned in my CPU280 article is finished and running. So this could be the stuff for another 'bus connected peripheral' article. It is also a EuroCard with its own Z80 processor and a parallel bus interface (for the host it appears just like a Z80-SIO with one data port and one status port).

For your answer (and further correspondence) you may reach me by e-mail (best) or by fax (2nd best) as written above.

Greetings, Tilmann

Thanks Tilmann for the long letter. Although I didn't get it until late November, your comments are always welcome. I have updated your subscription and you should receive them much faster than this response.

As to your comments, I had no idea what "BTW" means until Jay told me it stands for "by the way". I had looked in a few dictionaries and found nothing. So when it comes to "BTW" I guess I am a beginner.

I am passing on your comments to Tim McDonough. Thanks for the other ideas. As to the problems with the "Next Ten Years" article, that was mostly a deadline problem. That was in fact the first draft which never got updated as would be normal. I am trying to get more time to work on things and hopefully starting in January this will be the case.

On my going back to the roots, as far as TCJ is concerned, is not giving up on our past or NEW readers. For TCJ to continue we must add new readers every month. The only way we can do this is to make sure all readers can understand our articles. That means taking the time at the onset of an article to review all basic information our readers would need to know to be able to complete the project. The only option to NOT providing beginning information as has been the case, is to charge $50 a year subscription fees for the two or three hundred readers who could understand the advanced material. Not a choice I would like to make.

Supporting beginners is not teaching fundamentals, but just making sure we explain all our terms (like BTW and yes I apologize for not explaining SOG, means Semi Official Get-together and came from Micro Cornucopia's editor Dave Thompson) and tell them where to seek more information if they have not yet reached the needed level of knowledge or experience. Check out Jay's column this time and notice his treatment of explaining what he covered in back issues. Jay's review only took one or two paragraphs, but lets the reader know what they will find in those past issues, plus he reviews it a little now so they don't have to read the articles again before understanding this article.

Experience is one area that most beginners usually lack. A typical question I get is how do you get that experience. Hopefully our articles will provide projects that the reader can use to acquire their needed experience. Giving enough information just helps make sure this is a positive experience for them. Should they not have a positive experience it will not be just the project which doesn't get done, it will be TCJ that stops being read by them (as has been the case in the past).

My position on PALs is quite simple, I hate them. I spent too many years replacing them and throwing boards away because no replacements were available after the supplier went out of business. Now I agree that PALs are considerably better than they were, and most vendors are starting to supply some information so you could program them yourself, but that concept is still not the normal situation. I rather doubt that more than 25% of our readers have access to PAL programming equipment. With that in mind, projects that only give you the PAL code and not the equivalent circuit it replaces, makes it out of reach to 75% of our readers. Sure we can have them supplied by you when they buy your board, but that prevents the junk box builder from making his or her one of a kind project.

My other dislike of "only" providing PAL design information is the lack of education. We now use field programmable gate arrays (at my work) and the designer programs them using regular *discrete logic cells that are arranged or tied together (in a $3000 software program) much as if you used them separately. Thus the designer must know how to do regular logic design and not just some logic description for the PAL programmer. What our articles are suppose to do is teach the reader how to do this project and others like it. Showing them how and why you have this logic or that is what they want to know. Showing them next how that logic gets defined into several logic statements for the PAL programmer then is much more educational.*

What I have proposed is our articles be structured. Make sure we explain what the project is suppose to do. Review the fundamentals needed and where the reader may turn to gain the lacking knowledge or skills. Do the project much as we have always done them, but give them both a non-PAL schematic and the smaller PAL options. Follow up by providing how you can be reached, where parts are available, and any alternative projects to the one presented.

On software, I am not a Forth only programmer. My favorite language is assembly. My main concern is that "C" has grown way out of proportion to the claims laid on it. But both of those statements detract from the main idea which you so well expressed. We want the readers to see the steps needed in software as we do in hardware. Yes, I agree that Pascal might be better, but often it is too strict for hardware level programming. Pseudo style language flow charting is best, and lets the programmer do it in their favorite language. I like Forth only because it is truly the only platform independent language currently available. I have and can still write programs that can be used on many different CPU architectures without any changes. Simply load the Forth screens and run the program. That is all you need do, very simple, anyone can use them. My direction for TCJ is to try and make all our software projects that simple for the reader.

That brings me to the IDE article which many of our readers were glad to see but felt a bit short changed. I keep getting asked where the software part of the project is. Although you did a very good job of pointing out the hardware side of the interface, there wasn't any help in writing code to talk to the new device (a point I missed when I wrote the introduction). What is needed is information such as: what commands are needed and used, what are the responses you get back from the interface, what data structures are needed if any, are there any timing problems I must be aware of, and lastly is the data transferred byte at a time, word at a time, or must there be some form of DMA transfer? I must admit to not getting all publications, but I have not seen anything in print about IDE interfaces. So what I guess we need is a comprehensive review of IDE interface standards. We also need to know if the schematic is still correct or did you make changes to it to resolve the speed problem you mention.

On UNIX and Coherent, I have had several conversations with Rick Rodman and we are planning many articles that review the advantages and features that each implementation provides. Rick is always open to specific questions, so please pass any questions along that you might have to him. Which is another area you were concerned about. I think I did miss a few "biographies" of writers, but it has and always will be our main concern to provide the reader with a means of contacting the writer directly. Lastly your being on Internet and my not being there as well. GENIE now has a Internet service and I will be on it by the time you get this. It will take some time to get up to speed on using the service, but it should cut out the three month turnaround on your questions.

Thanks again, Bill Kibler.

Dear mr. Kibler:

In response to the article in TCJ issue no. 58 concerning moving the reset button on Kaypro computers, the article describes exactly what I have done many times on my own Kaypro computers and also for other members of my club, the Data Bytes User's Group of St. Louis, formerly The St. Louis Kaypro Users Group. The procedure is great but stops just short of one step I've taken.

On those Kaypro CP/M computers which have a cooling fan, the fan inhales air from the outside and pressurizes the case somewhat causing an outward flow of air through the many slots. I've found it desirable to close the hole from which the reset switch was removed so as not to unbalance the air flow, especially, in the older Kaypro 10s, with a full height hard drives, where there is a lot of heat generated.

Initially, I put a piece of duct tape over the hole. Later, I found metal hole plugs to close the hole. After my source of plugs ran out, I found a functional duplicate of the reset switch, a SPST normally off push button switch, at the local Radio Shack. This switch had the advantage of using a 5/16" diameter hole, which helped since I do not have a 17/32" to 9/16" diameter drill bit to make a hole like Kaypro used. I ran a pair of 24 gauge wires from the new switch to the original Kaypro reset switch and soldered them in place at both ends in parallel to the original switch, and insulated the soldered connections at the front end with clear silicon sealer. Makes a dandy strain relief, too.

I'm writing this on a modified Kaypro 10 with TurboROM, NZ-COM, and two 20 megabyte hard drives, which works very nicely with this switch arrangement. The original Kaypro switch makes a dandy hole plug, too!

Incidentally, related to heat, I've resoldered the main connector joints on the power supply of each Kaypro I've worked on to PREVENT future solder problems. I've seen some in which the solder was literally blasted out of the 110V pins on the power supply board (the bottom two).

Yours very truly,
Bob Rosenfeld
DBUG NEWS Editor
Data Bytes User's Group of St. Louis

Those are some good ideas and comments Bob. Charles Stafford is planning a power supply article soon, seems the *supplies are underrated and PC supplies can be used very cheaply. I take great joy in knowing that you are still using your Kaypro and wonder what was involved in putting two 20MB drives on it.*

I mentioned the taping the vents to Charles and he indicated that adding larger fans might be better all the way around. I think we will hear more about cooling those beasts at a later date. Lastly let me remind you that I would like to get your newsletter and can give discounts to your members if they subscribe as a group. These are a few of the NEW options available now that I have increased the rates. I cover those options elsewhere in the magazine.

Thanks for the letter Bob.

Mr. Kibler

Enclosed is my check for two year subscription.

I began building computers back in 1978 when a physics student of mine suggested that I might be interested in a EE extension course at the local university (Univ of MO.) As a result of the course I put together a microcomputer based on the National SC/MP microprocessor ....it had a total of 256 bytes of memory.

After that experience I worked as a programmer/system analyst for a local environmental engineering firm that used Cromemco computers in their data acquisition systems that took weather and pollution data....I stopped working for the firm about 10 years ago. I still have two complete working Z2D Cromemco systems in my basement with all sorts of software and accessory boards.

Much to my wifes chagrin, I have become a collector of old computers. Along with the Cromemco machines I also have a KIM computer, an old working Commodore 64 computer and about six IBM clones (use them as data collectors in my high school physics course) plus two IBM XT's.....I've been quite lucky at garage sales and auctions getting most of the old machines for very little...

Well.....that's my story.
Cordially yours
Lawrence LHote
Columbia, MO

You will be glad to know Lawrence, that my S-100 system, which still gets used occasionally, is mostly Cromemco. Herb Johnson our S-100 person, has a CROMIX system for sale (Cromemco's version of Unix) which I think about getting from him (used one at previous job). My spouse also likes picking on my "classic" collection of computers, but then collectors have always been treated that way until their collectible junk becomes valuable antiques. If TCJ goes the way I plan, it will not be long before these truly classic systems start being sought after and enjoyed by more than a few users like us.

Thanks for the subscription. Bill.

Gentlemen:

Thanks for sending me the sample issue of TCJ, No. 57. I'd like to subscribe for two years beginning with No. 58.

I'm using mostly CP/M computers, including Kaypro 2X (I think it is) and 10, plus the old North Star; but in addition I have a Texas Instruments Professional which does pretty well with some of the MS-DOS software. I am interested in installing ZCPR3, but don't want to do so until I have the source code for the utility programs that form an important part of the system. It seems to me that some of the bulletin boards had the whole package years ago. I have a lot of the material on the various components of the control program itself, but not on the utilities for directories, file manipulation, etc. I understand that a program needs to be tailored for ZCPR3 in order to make proper use of it and I'd like to see what's involved. Ultimately, as Humpty Dumpty said, it's a question of who is the master, and I want to be the master of my system.

If you know any BB's or libraries that have all the source code, please let me know. I am glad to see that you're offering support to us antiquarians who are interested in older machines. Keep up the good work!

Sincerely yours,
Fred Ordway

Thanks for the support Fred. Well the person who has all the answers about ZCPR is Jay Sage. Try his bulletin board listed on his ad (Sage Microsystems) inside the front cover. You might try GENIE's CPM section (or JAY.SAGE), it has plenty of source code to many of the utilities. We also have some of the original source in the Micro-C disk library (22 & 23), but I am not sure if it is all the utilities your interested in. You have however hit on one of the main selling points of using the old systems (we have all the source code for the programs, including the operating system). I am planning on setting up a new section for what and where you can find the source code for the many programs our user might need for using and upgrading their systems. I have always been after BIOS code, for without it you can not add new devices very well. The utility code is just as important as the BIOS but sadly more often overlooked. Please keep me posted as to what you found and where.

Thanks again Freed. BDK.

Dear Mr. Kibler

I have enjoyed your comments in TCJ ever since you have taken over editorship, especially in your replies in REARER to READER.

I would like to add my perspective on things:

I am looking for a replacement or the information that was presented in early BYTE/Creative Computing/Micro/Dr.Dobbs. I read your first Editor's Comments and it resonated. I would like to influence you to make only two changes over your comments : 1) open up hardware articles to programmable logical devices if the article author specifies how the prospective builder could obtain the programmed devices; 2) provide a quantitative definitia of what a rare or expensive component is.

On the programmable devices, I offer to help anyone program any memory device, and some PAL, GAL, PEEL and PLD. So any programmable devices should be allowed in hardware articles. If this becomes important, I will write another letter giving a device list and conditions.

On the rare and expensive components, I would like to suggest that a popular article use only components that can be found in Radio Shack, from Jameco, JDR, Digi-Key and Moser. A popular hardware article would be an article about hardware to solve a useful or fun function using an embedded computer. Occasionally a special hardware project is called for, the task can only be solved well with a special component, many readers would find the approach interesting, but few would actually build the unit. In this case I would like to suggest line drawing at the author locating at least 200 of the unique components and making the address of this location available in the article. I would further like to draw the line at $100 for a component and $250 for a hardware project.

I would like a hardware to software article ratio of 50/50.

I am seeking a magazine where people of interest similar to my own can meet. Right now I subscribe to Circuit Cellar, Midnight Engineering, TCJ and Atari User. If I dropped one of these other magazines I would have more money for a TCJ subscription. In the past I have found Circuit Cellar the magazine the most worth keeping, even though I don't like it.

I will try quickly to describe my interests:
Assembly language program: 6502, 6809, 68HC11, 65C816. HOL program: BASIC, FORTH, C (in that order of preference). I like to build hardware. My preference is a 65C816 based system using 65C51 and 65C22 for I/O. Found that the single board 68HC11 is the fastest and lowest cost solution to embedded control (also with plenty of free software tools).

Would like to read software articles based on the languages I use. Would like to read hardware articles based on the processors I like.

I would like to read or write an article on how to use (hardware and software) a XT clone keyboard to any of the processors I am interested in, as that is the best and cheapest input device. I would like to read lots of articles on how to connect output devices to the processors I am interested in.

Would like TCJ to set publication cost to subscription rate such that it was 100% subscription supported. It has broken my heart too many times to see a well loved magazine fold when the advertising withdrew. Tell us how many pages of text and schematic you can mail to us per year for how many dollars per year and let us make recommendations.

Suggest that you publish a "How to Write for TCJ" include style, format, word processors supported, CAD formats supported and desired type of articles. There is no reason that all submissions can not come "camera ready".

Joseph Ennis
Valparaiso Florida

Well Joseph that was a great letter that reinforces many of the ideas I have been talking about lately. On cost of publishing, I have adjusted the rates closer to my actual costs, and separated out the extra mailing costs our overseas readers pay. Mailing cost went up, but our rates had stayed the same, but now all that has changed.

TCJ has never had and may never be a advertiser supported magazine. ELEKTOR, just folded US publishing with 10,000 readers, because the profit margin was too small. Some day I hope TCJ has a profit margin to be concerned about. WE are and always will be oriented to users, and as a classic support magazine (no vendors provide support), that means no advertisers will be interested in us either. If you look at PC magazines (IBM clones that is), they only sell new items. New items are not what are readers are interested in.

The PALs and all are still not my favorite component. I appreciate your offer to burn PAL's, but that could be a bit tricky for someone in China, or India. TCJ's readership is far outside the US borders and as such keeps me from endorsing a PALs only design. I consider the best approach is one that uses regular TTL logic and then shows how that logic can be replaced with a PAL. We leave the choice and option up to the reader and not the writer. Remember, you had to figure the design out first, as if you could use those TTL devices before you programmed that PAL. Why not pass along that design and thinking to our readers as well as the other aspects of the project (much more educational).

Rare and expensive components to me, are ones not available in junk boxes. Now many of our readers have access to very fancy junk boxes, but most have to remove components from old S-100 boards or game machines. That means at least ten year old technology or simple TTL devices. Your dollar limits actually are a bit high for me, maybe half those values would be better. I will have to wait and see what our other readers have to say before I place judgment on that position. A reason for that is a ten dollar component may become a forty dollar one in South America.

I hate to keep repeating myself, but I feel Americans have lost sight of how much freedom of access we have to cheap components and especially the habit of using the latest high technology solution. I have received comments about using Forth (which I noted you use), but in Russia where big fast computers are a rarity, Forth is the number one language. Many countries still do not have the computer systems we take for granite.

Don't hesitate to write those articles about using PC Clone keyboards, I think many of the PC items are just the ticket to keeping alive classic systems at mini-

mum cost. Thanks again Joesph for your comments. Bill Kibler.

DEAR BILL,

I have been a subscriber of COMPUTER JOURNAL since its inception and enjoy reading most of the articles. Due to circumstances beyond my control the last few years I have been unable to do very much in the way of experimenting with the various things that can be done with computers. Consequently my equipment is rather ancient and I have not added to the systems that I use. Now that deterrent has been eliminated so perhaps I can get on with the playing.

The computers here consist of an H89A that I assembled in 1980 or 1981; two Z90s and a couple or three years ago I bought the Ampro Little Board 100 that Art Carlson had. All systems are operational and have 4 floppy drives and the Ampro has a 10M hard drive.

I have no idea how many you have on your subscription list but the printing of the labels could probably be done here and at least for awhile and it would cost you nothing. If that would interest you send me a list and I'll make you a sample. The printer here is a Diablo 630.

The occupation here during my working life was printing -- both newspaper and state printing plant. Therefore typographical errors and grammatical errors are quite noticeable. I do not wish to be overly critical but hope there is more time to proofread the copy in the future.

I hope publication of the COMPUTER JOURNAL continues for a long time.

Sincerely,
MARVIN F. ROBERTS
TOPEKA KS

Thanks Marvin for the offer of doing labels. The problem I have discovered is not printing the labels, but doing all the bookkeeping and paper work. I discovered problems with my mailing program when I received a Christmas gift subscription request. How was I to charge and keep track of the payer and the receiver of the gift. With the addition of Micro-C disks, the many different mailing cost, and some sales tax collection and reporting needs, it has become apparent that a full charge bookkeeping program (with mailing list option) is needed. About the only way someone else could help me with the labels would be if they took over the 1-800 number, credit card processing, back issues and disk coping and mailing, banking mail-in subscriptions, and just sent me the labels after telling me how many to print. Unfortunately it would also have to be done free, as we are not breaking even now and could not afford to pay anyone for those services. I also forgot to mention they should also be available for answering the phone during normal working hours, have knowledge of classic computers, and use computers.

As you can guess I don't really expect any help in this area, it is just another situation in which an improved program will make things easier and faster and over time become less of a problem. Finding time to set up the program however has been the real problem.

I hope your getting back into computing is not buying a PC clone! Your past knowledge with older machines would certainly make you an unhappy pc user. I know Art will also be glad to read that you are still using the system you got from him. Keep up the good work and let us know what that new system becomes. BDK.

Dear Bill.

Received issue #58. Thank you. I want TCJ to succeed. Please don't worry overly about what Joseph Mortensen said. I saw the typos, etc. and I felt the content of TCJ made up for them. If I send in articles. what format is readable by you? I prefer 3.5 inch MacIntosh or 1.4M 3.5 inch IBM.

In response to the letter in #58 from Christopher Browne: I use New Micro Computers of various sizes (the ones with embedded forth language) and I think they are great. I have written software modules for X-IO, i2c. stepper motors, cardreader emulators, cardreaders, ADB support, multi-tasking, etc. I will provide details on these items to interested parties. I hope to publish this info in TCJ if Bill is interested.

Gus Calabrese
Prsident WFT.

Thanks for the good words here and in Forth Dimensions. I saw your letter to the editor in which you mentioned us, we need all the good words we can get. Currently I use PC Clone based system for my TCJ work. I have plenty of other systems (most CP/M formats supported, including 8 inch) but have not broken down and bought a Macintosh yet (did buy an Magic Sac for my Atari, but alas it doesn't work). Please write an article on using the New Micros system, we get plenty of requests for just what you have done. Drop me a note before you start on the articles, so I can make sure you are not doing the same project one of our other writers is working on. Thanks again Gus. Bill.

Admissions to TCJ

To submit articles or letters to TCJ please send them to the below address. I can be reached alternately, by means of GEnie or CompuServe as listed below.

TCJ is always looking for new writers and we accept most manuscripts that help our readers update and improve their computing skills and systems. We especially look for low tech solutions to complex problems. TCJ also supports systems no longer supported by manufacturers or other magazines. Some of these systems are : Z80 based S-100, Kaypros, Xerox 820, and other systems running CP/M. We also support the 68xx(x) series of machines made by GIMIX, SWTP, and those systems that run Flex or OS9.

TCJ
P.O. Box 535
Lincoln, CA 95648-0535

GENIE B.Kibler
CompuServe 71563,2243

The Z-System Corner

By Jay Sage

Regular Feature

ZCPR Support

Advanced ZMATE

Advanced Applications of ZMATE

Autoexec Macro for Automating ZMATE

ZMATE is a very flexible, powerful, and configurable editor (and even word processor), but it can actually do far more than meets the eye, even the eye of an experienced user. Over the years I have continued to develop more and more productive uses for it. On the one hand, I have learned to automate the operation of ZMATE itself; on the other, I have learned how to use ZMATE as a tool to automate other operations. In this column and several subsequent columns I would like to share some of these techniques with you.

Relatively few people have taken advantage of the availability of ZMATE. The reason for this, I think, is that most people already had an editor or word processor that they were comfortable with, and they saw no reason to learn to use a new one. I hope to convince some of you here that ZMATE can add a whole new dimension to what you can accomplish on your computer, and that it is worth the effort to learn to use it.

The discussion here will also illustrate a more general principle, one that constantly motivates Z-System developments. When software supports a flexible, generalized interface, it can find uses beyond anything the author originally envisioned. I am sure that Michael Aronson (MATE stands for Michael Aronson's Text Editor) would be amazed to see how I use MATE -- but he would not be amazed to see that I am using it in new ways. He put in all the hooks and functions to make that possible. You may have other programs that provide powerful interfaces, and you may be able to invent radically new applications for them.

Before getting down to business, I would like to mention that Gene Pizzetta and Howard Goldstein teamed up to clean up the ZMATE code. Bridger Mitchell did a fabulous job of disassembling PMATE (Mike Aronson could no longer find the source code!) and then adding a number of nice features, such as two-window operation. However, a number of old bugs remained, and a fair number of new ones got added. In addition, Gene and Howard have made it possible to set most of ZMATE's configuration options using Al Hawley's ZCNFG tool.

If you purchased ZMATE from Sage Microsystems East, you may get an update by sending back your original diskette in a mailer that can be reused to return the diskette to you. Include a return-address mailing label and enough stamps or money to cover the postage. If you are out of the U.S. and have no way to supply stamps or money, SME will cover the cost.

In issues 46 and 47 of *TCJ* I described ZMATE in considerable detail. The first column covered its underlying principles of operation, such as the totally user-controlled binding of key sequences with characters and functions and the fact that ZMATE is really an interpreted programming language, like BASIC, but with command primitives designed for text processing. It pointed out that ZMATE programs -- or macros, as they are generally called -- can be run in several ways. They can be entered manually and executed from ZMATE's command line; they can be stored in what is called the Permanent Macro Area or PMA; and they can be stored and executed from any of the 10 numbered editing buffers (0 through 9).

In issue 47 I gave a broad overview of ZMATE's macro language. In this column you are going to see some substantial examples. If you have and use ZMATE, you might want to study them in detail; if not, by skimming the comments you will get an impression of how ZMATE operates.

The ZMATE Autoexec Macro

The ZMATE permanent macro area contains definitions for a number of macros that have single-letter names. These macros can be invoked in other macros entered on ZMATE's command line or executed from ZMATE's text editing buffers. You'll see some examples later. Each permanent macro definition begins with a control-X character followed by the name of the macro, and it ends either with the beginning of the next definition or the end of the PMA.

There can also be one special definition, a macro that executes automatically whenever ZMATE starts up. I call it the "autoexec" macro, since it is like the AUTOEXEC.BAT file that runs at bootup on a DOS machine. (I must have given it that name before NZCOM came along; otherwise I would have called it a "startup" macro.) This macro must be the first definition in the PMA, and it must start with a control-S instead of control-X followed by a name.

If no autoexec macro is defined, ZMATE interprets command-line arguments as the names of files to open. The first token is the name of the input file to edit. If a second file is named (and the first file already exists), then it is used as the name of the output file. Thus the command line

```
EDIT MYFILE
```

opens an existing file or creates a new file called MYFILE. The command line

```
EDIT THISFILE THATFILE
```

opens an existing file, THISFILE, (you get an error message if it does not exist), and writes the edited contents to a new file, THATFILE (you get an error message if it already exists).

Making a Dedicated Tool

The autoexec macro allows one to turn ZMATE into a dedicated text-processing tool. For example, let's suppose that we want a tool that will convert a text file entirely to upper case. We could load the macro shown in Listing 1 into an editing buffer; store it into the PMA using the command QMC (Q-Macro-Copy); and then clone a new version of ZMATE using the XD (file Duplicate) command. For example, XDupcase would create a file UPCASE.COM that would run the macro in Listing 1 as soon as it was invoked. We might use our new tool as in the following examples:

A>upcase thisfile	convert text in the file THISFILE to upper case
A>upcase lc uc	convert text in the file LC to upper case and store it in the file UC

Of course, this is not a terribly practical example. ZMATE macros, being interpreted, do not run very fast, especially when, as here, every single character has to be processed one-by-one. This simple example could be implemented rather easily in any programming language, such as BASIC or PASCAL. Using the SYSLIB assembly-language library routines, we could even do it quite easily in assembly language. However, more complex editing tasks might be so hard to write in other languages, that we would not even attempt to do so. Operations that make use of ZMATE's more powerful macro commands, such as searching for text, can run quite fast.

A Generalized Autoexec Macro

One day I suddenly realized that it was a waste of disk space to keep cloning ZMATE every time I wanted a tool to perform a function with a particular macro, and I conceived an approach that, in a sense, allows the autoexec macro to be specified on the command line. This turned out to be enormously useful.

Before I describe how this macro works, I would like to show you an example of how it is used. First, this will give you some idea of what the macro has to accomplish. Second, it may give you the motivation to slog your way through the listing for that macro!

I often want to make modifications to my ALIAS.CMD file. Frequently, I even know the name of the alias that I want to change. To expedite this operation, I wrote an ARUNZ alias that can be invoked with a word as an argument. It brings the ALIAS.CMD file into ZMATE with the cursor on the first occurrence of the specified word. If that's not the right occurrence, invoking my "ALT-comma" instant command will quickly search for the next occurrence. Here is the listing for the ALED (ALias EDit) alias.

```
                ALED
1       a0:;
2       b0:edit $$xialias.cmd$$b8eies$1$$
                b9ei.oalias.cmd$$btea.8;
4       if in U%>pdate permanently? ;
5          mcopy b15:=alias.cmd /e;
6       fi;
7       $hb:
```

The first line changes to the A0: directory. This is where I keep the ALIAS.CMD file. It is the root of my path and is on a RAM disk. The next command (line 2 with overflow to 3) invokes ZMATE (which I rename to EDIT). More on this command in a moment. After the editing has been completed, the command on line 4 asks if I want to update the permanent copy of ALIAS.CMD on the hard disk. If I answer affirmatively, then the command on line 5 copies the file to B15:, overwriting the original file. Line 6 terminates the flow control state initiated in line 4, and line 7 takes me back to the directory from which I started all this. I have a similar alias, by the way, called ZFED (ZFiler EDit) that works on the ZFILER.CMD file containing the macro key definitions for ZFILER.

Now let's look at that long editing command. Once we allow for the fact that a dollar sign must be represented in any alias by a pair of dollar signs, we figure out that the real command tail is

```
$xialias.cmd$b8eies<param1>$b9ei.oalias.cmd$btea.8
```

Here "<param1>" is the first parameter token on the command line by which the alias was invoked, namely the word I want to search for.

The autoexec macro treats everything before the first dollar sign as the specification for the names of the file(s) to open for editing and everything after it as a macro command to execute after it is open. In this case, there is no file name; the whole command tail is a macro. Since ESC characters cannot be entered on a command line, dollar signs are used instead. This macro thus becomes the following, to which I have added comments and line numbers (and where, as in all ZMATE macro listings, a dollar sign is used to represent the ESC character).

```
1   xialias.cmd$      ; read in ALIAS.CMD file
2   b8e               ; switch to buffer 8
3   ies<param1>$      ; insert text "es<param1>"
4   b9e               ; switch to buffer 9
5   i.oalias.cmd$     ; insert text ".oalias.cmd"
6   bte               ; switch to buffer T
7   a             ; go to beginning of ALIAS.CMD
8   .8                ; execute command in buffer
8
```

What's happening here is that the ALIAS.CMD file is being read into the T buffer (note that it is not being opened for editing, only read in). Then buffers 8 and 9 are being filled with text to be used as macro commands. The text in buffer 8 is a command to try searching for the next occurrence of the word named on the ALED command line, if any.

The text in buffer 9 is meant to be used after any changes have been made to the alias definitions. It invokes my permanent macro "O", which is a version of the ZMATE XO (file Output) command that prompts for the deletion of any existing file of the given name. This allows me to write out the new ALIAS.CMD file right over (on top of) the old one. If we have optimized the speed of ARUNZ by keeping the ALIAS.CMD file early in the directory and early on the disk, this would generally keep it there.

The most important thing that I want you to take away from the discussion so far is that I could have created a special COM file, say ALED.COM, to perform these tasks. However, this would have used up an additional 25K or so of disk space, just for this one task. Because of the autoexec macro, I was able to accomplish the same function using my standard version of ZMATE (EDIT.COM) and a short alias definition in ALIAS.CMD. In addition, because the ZMATE macro specification is in ALIAS.CMD and not embedded in the permanent macro area of a special version of ZMATE, changes are much easier to make.

Details of the Autoexec Macro

A commented, line-numbered version of the autoexec macro is shown in Listing 2. Probably some of you will not be interested in the details of this macro, and you might want to stop reading now. However, anyone who uses ZMATE macros will pick up a number of interesting MATE programming tidbits by following along.

My autoexec macro uses buffer 0, which is generally considered a scratch buffer, and buffer 5. I chose a middle buffer so that command-line macros generated by the autoexec macro could use both high-numbered and low-numbered buffers.

In line 2, we move to buffer 5 and (line 3) insert whatever text was in the command line tail. ZMATE uses a number of special text-source expressions that begin with a control-A, here represented by caret-A. The control-A can be followed by the number of one of the numbered buffers or by a colon, as here. We will see some more examples of this later.

Line 4 begins the processing of the command tail. We start at the beginning and place the tag there. Then we strip away any leading white space. Spaces actually would do no harm, but tabs do cause a problem. We do this by moving the cursor until we encounter the first character that has an ASCII value higher than that of a space character. Then, in line 12, we delete from the tag to the present location.

We now make sure that something is left (line 13). If not, we just return now to the T buffer and quit (line 15). If there is something left to process, we proceed to the first major step: spitting the text into the part representing the file names and the part representing a macro command to perform. The boundary between these two parts is the first dollar sign, if any.

In line 17 we look to see if the string starts with a dollar sign, indicating that no file names are specified. If so, we delete the leading dollar sign and store a 0 ('false' value) onto the numerical stack for testing later. If the first character is not a dollar sign, then we store a 1 ('true' value) onto the stack and search for the first dollar sign later in the string. Since there might not be one, we run the E command (line 22) first, so that failure of the search will not halt processing.

Line 24 tests the command error flag to see if a dollar sign was found. If not, we move the cursor to the end of the text (line 25). Otherwise, we delete the dollar sign (line 27), leaving the cursor on the character that followed the dollar sign. At this point, the tagged block comprises the characters from the beginning of the string up to the cursor location. This block contains the name or names of files to open. The command in line 29 moves this text to buffer 0, leaving in buffer 5 only the text that represents the initial macro command to run.

Now we begin a block of code that does some special format conversions on the macro command. There are several characters that cannot be entered directly on the Z-System or CP/M command line. These include lowercase letters (the command processor always converts your command line to upper case) and control characters, including the ESC character. The caret character is used as a prefix to indicate that the following character is to be converted to the equivalent control character, while the double quote character (") is used as a special escape character to indicate that the character following it is to be taken as is, without special translations.

This scheme has one minor complication. Since we usually type commands in lower case and the command processor converts them (against our will), the autoexec macro normally assumes that letters are intended to be lower case, and so it converts them back (unless the double quote prevents this processing). Let's look at how the macro does this work.

At line 32 we begin a repeat block that continues until the end of the buffer is reached (line 51). In line 33 we check to see if the cursor is on an upper-case letter. This test is a little tricky. Because ZMATE does not support tests of less-than-or-equal-to or greater-than-or-equal-to, we have to take a roundabout route. We test to see if we are either below "A" or above "Z" (i.e., do not have a capital letter), but then we negate the test with the prime (apostrophe). Thus we test for a capital letter.

If we have a capital letter, we convert it to lower case by replacing it by a character greater in value by 32 decimal (20 hex). By representing this in the form quote-space (i.e., the ASCII value of the space character), the macro does not depend on the radix of the number system currently in use. The replacement operation moves the cursor to the next character, so we go back to the beginning of the repeat loop (line 35) to process the next character.

At line 37 we check for the quote character (''). If found, we delete it, skip over the character that follows it (so that it is left as is), and loop back to continue scanning. This part of the macro code allows us to leave a character in uppercase by putting a quote character before it. We can also enter the three special characters -- quote, caret, and dollar sign -- by preceding them with a quote character.

Line 42 of the macro tests for the caret character. If it is found, it is deleted (line 43) and the next character is forced into the ASCII control-code range by replacing it with the value obtained by logically ANDing it with 31 decimal (1F hex, 00011111 binary).

At line 46 we come to the test for the last of the three special characters, the dollar sign. If we find one, we replace it with the ESC character and loop back. If we do not detect any special character, we reach line 50 and simply move on to the next character. When we reach the end of a buffer, the character under the cursor is the null character with the value zero. Line 51 tests for that condition and continues the repeat loop if it is not true.

Once we have processed the entire command tail text, we can operate using it. We go back to the text buffer (line 52). Then we pop that value we pushed on the stack earlier to remind us whether or not we found a file specification earlier. If so, we invoke the XF (open File) command using as the file name(s) the string contained in buffer 0. ZMATE's ability to get macro command input indirectly from its buffers is one of its most powerful features!

After we have opened any files specified, we clear out the text in buffer 0 (we don't want to be sloppy and leave old text lying around!) and then execute (line 57) the macro we composed in buffer 5. After that is complete, we again clean up after ourselves by clearing buffer 5. That's it!

Plans for Next Time

I still have a lot more to cover on the subject we started this time, and I expect to have two more installments. However, the next issue of *TCJ* is number 60, *TCJ's* tenth anniversary. Bill Kibler asked me to do a reprise on NZCOM and Z3PLUS from a beginner's point of view. This won't be easy for me, since I have not been a beginner at this for almost a decade now! But I'm going to try. I plan to take one or two of my spare computers and start from scratch bringing up NZCOM and/or Z3PLUS. In the process I will probably discover a number of things that don't go quite as easily as they should, and some changes to the RELEASE.NOT file will undoubtedly result.

In the next installment on ZMATE, in issue 61, I will show a much more elaborate suite of automatic macros initiated from an ARUNZ alias. This is my GEnie mail-replying environment. A MEX script logs me onto GEnie and captures all my new mail into a file. The ZMATE macros then automate the process of creating reply messages for later upload to GEnie.

A third installment I have in mind will show how ZMATE-based text processing can be used to automate other computing tasks. Recently I have been using ZMATE (actually PMATE) extensively this way on my DOS computer. For example, I have been running numerous circuit simulations in which I want to vary some parameters. For each simulation I used it invoked PMATE, found the lines in the circuit definition file that needed to be changed, entered the new values, saved the file, and invoked the simulator. Now I just write an alias (batch file) that does the whole thing automatically. It generates a PMATE command with a macro in the command tail to insert the new parameters. In some cases, I even have another command in the batch file that instructs PMATE to examine the simulator's output file and determine whether the circuit performed properly. The batch file can then automatically adjust the circuit parameters and start the cycle again. The most dramatic example of this was when I went away on vacation for almost a month. The computer worked the whole time unattended, and when I got back, I had a nice text file with all the results neatly summarized.

To contact Jay use, JAY.SAGE on GEnie, or sage@ll.mit.edu on internet. Alternately try one of his phones or BBS numbers listed in the SAGE Microsystems East advertisement on the inside cover of this issue.

Listing 1.

```
; Macro to convert text in an entire file to upper case
; and then write out file and exit. There is code to
; force disk scrolling.

^S                      ; autoexec macro
b1e                     ; go to buffer 1
i^A:$                   ; insert command tail
bte                     ; return to buffer T
xf^A@1$                 ; open files named in buffer 1

[                       ; REPEAT

(@t<"a)!(@t>"z){        ;  IF char outside range a..z
;  (i.e., not lower case letter)
m                       ;    just move to next character
}{                      ;  ELSE
@t-" r                  ;    replace with upper case char
;    (by subtracting 32)
}                       ;  ENDIF

@t=0{                   ;  IF end of buffer or file
m                       ;    try moving to next character
@t=0{_}                 ;    IF still end, exit repeat loop
-m                      ;    otherwise, move back and
continue
}                       ;  ENDIF

]                       ; END REPEAT

xe$                     ; end editing
xh                      ; exit MATE
```

Listing 2. The full definition of my autoexec macro with comments and line numbers.

```
01 ^S                        ;Autoexec Macro

02 B5E                  ; work in buffer 5
03 I^A:$     ; insert command line argument string
04 A                    ; start at beginning
05 T                    ; tag it

; Skip over white space

06 [                    ; REPEAT
07   @T>" {             ;  IF char higher than space char
08     _                ;    exit repeat loop
09   }                  ;  ENDIF
10   M                  ;  move to next character
11 ]                    ; END REPEAT

12 #D                   ; delete the block of white space

13 @T=0{                ; IF nothing left
14   BTE                ;  go immediately to T buffer
15   %                  ;  exit this macro
16 }                    ; ENDIF

17 @T="${               ; IF first character is dollar sign
18   D                  ;  delete it
19   0,                 ;  push 0 (false) onto stack
20 }{                   ; ELSE (first char not dollar sign)
-1,                  ; push 1 (true) onto stack
E                    ; turn off error trapping
S$$                  ; search for first $, if any
@E{                  ; IF none found
  Z                  ;   go to end of text
}{                   ; ELSE (dollar sign was found)
  -D                 ;   delete the dollar sign
}                    ; ENDIF
#BM                  ; move preceding text to buffer 0
}         ; Reformat the macro passed on command line

[                    ; REPEAT

@T<"A!(@T>"Z)'{ ;    IF upper case character
  @T+" R             ;   replace it with lower case char
  ^                  ;   start next loop
}                    ; ENDIF

@T=""{               ; IF character is double quote
  D                  ;   delete it
  M                  ;   move past following character
  ^                  ;   start next loop
}                    ; ENDIF

@T="^{               ; IF character is caret
  D                  ;   delete it
  @T&31R             ;   replace char with control char
  ^                  ;   start next loop
}                    ; ENDIF

@T="${               ; IF character is dollar sign
  "$R                ;   replace with ESC character
  ^                  ;   start next loop
}                    ; ENDIF

M                    ; move to next character

@T=0]                ; UNTIL we reach end of buffer

BTE                  ; go to buffer T
@S{                  ; IF stack contents true
  XF^A@0$            ;   open file spec in buffer 0
}                    ; ENDIF

B0K                  ; clear out buffer 0
.5         ; execute macro passed on cmd line
B5K                  ; clear out buffer 5
                     ; end of macro
```

SUBSCRIPTION RATES for *The Computer Journal*

These rates are effective ***January 1, 1993.*** Foreign rates now reflect the actual charge of additional mailing fees and have been set by adding those charges to current subscription fees.Calculated fees for Canada and Mexico are $8.00 for surface and $10.00 for air. European and other countries have been averaged to the rates of $10.00 for surface and $20.00 for air. **Sales tax is no longer needed for California residents on mail order subscriptions,** taxes are collected however on items purchased by mail, such as back issues and floppy disk programs.

Education rates are available to all institutions and public libraries. User groups and organizations who subscribe as a group (five or more subscriptions with each order) can use the education rates.

Sample issues are available at $5.00 each and will be mailed immediately when pre-paid. Trial (and samples not pre-paid) will be shipped with next general bulk mailing (Bi-monthly).

Subscription	U.S.A.	Canada/Mexico	Europe/Other Countries
1 Year Surface	$24.00	$32.00	$34.00
education	$22.00	$30.00	$32.00
1 year Air	$34.00	$34.00	$44.00
education	$32.00	$32.00	$42.00
2 year Surface	$44.00	$60.00	$64.00
education	$40.00	$56.00	$60.00
2 year Air	$64.00	$64.00	$84.00
education	$60.00	$60.00	$80.00

All U.S.A. shipping is third class bulk mail, except First class or air which cost an additional $10.00. Foreign surface and Foreign Air Mail is currently shipped as printed matter and packaged in appropriate mailing envelope. Current rates are calculated on average shipping weight of 6 ounces.

32-Bit Systems

All Readers

MINIX File System

Real Computing

By Rick Rodman

Each crisis we face seems much less daunting when it has passed. But there, in the thick of it all, is the true test of our Insight, our Ingenuity, and our downright Determination, in the true TCJ Ethic, to Do The Job with the Tools At Hand. No words can truly convey the Challenge, the Struggle, and the Exhilaration of Triumph.

The Superblock is Corrupt!

It was a day like any other, when these chilling words that strike fear into the heart of any Minixer appeared from my PC-532, "The Superblock is Corrupt!". The initial despair gave way to a resolute determination to fix the problem. This problem can occur on *any* Minix system. Minix-PC and the PC-532 use identical file systems, and the Atari ST and Amiga use the same file system but with bytes reversed.

The tools required are: First, some means of reading and writing sectors in the Minix partition or volume *outside* of Minix. Second, a bootable Minix diskette, or set of diskettes, with *fsck* somewhere available, with /dev/hd2 accessible but not the system disk.

In the Minix-PC environment, these tools take the following form. For the first requirement, you need a DOS partition or a bootable DOS diskette and a read-write tool similar to READABS. The second tool is your Minix-PC diskette set.

In the PC532 environment, the reading and writing can be done with the monitor. The other tool needs to be made - you need one disk with a bootable image (the hard disk is fine), and another disk with a complete file system and *fsck*.

One would think that *fsck* alone could fix a corrupt file system. That's what it's supposed to do, after all. Unfortunately, *fsck* can't fix the file system - it won't even try - if the superblock has been damaged.

Before we get to work fixing the blown superblock, here's a brief explanation of the Minix file system. By these words "file system" I mean the overall structure placed on a floppy diskette or hard disk partition, within which files are created by the operating system.

A Minix file system consists of a 1K (1024-byte) "boot block", the "superblock"; some number of "inodes"; and the data in "zones".

The "boot block", on a floppy, could hold the BPB (Boot Parameter Block explained in my "Mysteries of PC Floppy Disks Revealed" article on page 16 of issue #44) or possibly a short boot program. On the hard disk, however, the partition table is in the first sector, so there's no need for anything to be in the boot block. Because PC hardware almost always uses 512 byte sectors (due to a logic bug in DOS 1.0), two sectors are required for each 1K Minix block.

The "superblock" comes next. It's an 18-byte data structure at the second Minix block of the file system. It consists of the values listed in figure 1.

The definition for this structure is in fs/super.h on page 540 of the Minix 1.5 source listing. A whole block, 1K bytes or two sectors, is allocated to the superblock.

There is some number of "inodes," which correspond to directory entries, or the FAT under PC-DOS, but are a little different. Each inode is a 32-byte data structure very similar to a CP/M directory entry, but with no name in it. As in CP/M, there must be a number of inodes such that they take up an even number of 1K blocks.

The inode has date and time in it, attributes, and a short list of block numbers. These block numbers point to the first few blocks of a file. Once the file goes beyond that many block numbers, though, instead of allocating another directory entry like CP/M does, a data zone is allocated, and more block numbers written there. That block is called

1.	2 bytes - number of inodes on the file system
2.	2 bytes - number of zones on the file system
3.	2 bytes - inode bit map size in 1K (1024-byte) blocks
4.	2 bytes - zone bit map size in 1K blocks
5.	2 bytes - first data zone
6.	2 bytes - log 2 of blocks per zone - always 00 00 (see below)
7.	4 bytes - max. size - always 00 1C 08 10 hex (This is 262,663 decimal - see text)
8.	2 bytes - magic number (7F 13 hex)

Figure 1 Superblock Structure

a "single indirect block". If that block fills up, then another data block is allocated, and the block numbers of the indirect blocks are written there - this is a "double indirect block". If the block numbers are two bytes each and the indirect blocks are 1K bytes, you can have seven direct blocks, one single-indirect block from the inode, one double-indirect block, and 512 single-indirect blocks from the double-indirect block, for a maximum file size of 7 + 512 + (512 * 512) blocks, which works out to be 262,663 blocks or 268,996,912 bytes. Since a Minix partition can't exceed 64 megabytes, that's plenty big. (In Unix, you can have triple-indirect blocks.) The definition of the inode structure is in fs/inode.h on page 535 of the listing.

But wait, you say, how do I find a file if there's no filename? The filename isn't in the inode because it's in a directory somewhere. A directory is a regular file with a special attribute, with all of its space out in the data area. Each entry in a directory has a two-byte "inode index", from 0 to the number of inodes minus one, plus a 14-character filename. Using the inode index, it's quite easy to go to the inode and thus to the file data. But, if you ever need to go *from* the inode *to* the file name, out in some directory, that's quite a difficult thing to do.

Also note that there's nothing preventing two directory entries from pointing to the same inode. This could happen quite easily by accident, or we might do it deliberately. In Unix parlance, this is called a "hard link", because Unix has another kind of link called a "symbolic link", which goes through the filename. There's a link count field in the inode, *i_nlinks*, which keeps track of how many files point to the same inode, so that if you should erase one of the linked files, the other one doesn't get lost.

After the inodes comes the actual user data, in "zones". The "zone" is like an MS-DOS "cluster" or "allocation unit" - it is the granularity in which data space is allocated by the operating system. Actually, for all intents and purposes, a zone is the same as a block - 1K bytes.

You might pause for a moment and make sure you understand all of the foregoing before proceeding. To read more about this, the place to go is Andy Tanenbaum's book. The Minix manuals don't waste any time describing these details. This complex inode and indirect-block structure, with the filenames separated, is traditional in Unix and Unix-like systems - and that tradition, not its efficiency or speed, is the reason it is used. In actuality, compared to most other designs, it is inefficient and slow, and, in my opinion, the "link" feature is really just calling a bug a feature.

There are some "Unix believers" who will sincerely argue that this is the most efficient system possible. These people are wrong, of course, but they will defend to the death the Standard Unix way of doing things no matter how bad it is, just Because It Is Unix. When really pressed, they'll try to dismiss a topic as a "Religious Issue", which is a face-saving way of saying "I see I'm wrong, but I'll never admit it".

Now that we've got a basic understanding of the Minix file system, let's get started fixing the blown superblock. Obviously, a lot of trouble could be saved if you already have a hex dump of what your superblock is supposed to contain. If you're a Minix user, I *urge* you to go and dump the second 1K block of each of your Minix partitions. Print them out and save them - because only you can rescue your superblock.

But now, we're going to rebuild the superblock by recomputing all of the values in it. We have to start with the number of zones of the file system. Now, in theory, you could have multiple blocks per zone; in practice, nobody would, because Minix can't handle more than 64K blocks anyway. So, #2 of the superblock is just the number of kbytes of the partition, low-byte-first.

The zone bit map is an array of bits, one per zone, saying whether each zone has been used or not. We need one bit per zone for the zone bit map. How many blocks is that? Well, it's #2, divided by eight, plus 1023, divided by 1024. Convert the result to hex, low-byte-first, and that's #4. Did I mention you need a hex calculator?

The number of inodes in a given file system could be anything. This is the problem. I had to go looking through the disk with the monitor to see where they ended, but there appears to be a general rule: the number of zones divided by 3. Oddly, *mkfs* doesn't always bother padding this value to some multiple of 32, so some space is wasted. I think it should be rounded up: Take #2, divide it by 3, and round up to next multiple of 32. Express as hex, low-byte-first, and that's superblock item #1. It wouldn't have to be this value, and it might not - but in my case, it was.

Like the zone bit map, there's an inode bit map. If you have 8,192 inodes or less, you need one block; otherwise, you need 2. Put 01 00 or 02 00 as #3.

The only thing left is the starting data zone. This is 1 (boot block) plus 1 (superblock) plus #1 times 32 divided by 1024, plus the two bit map sizes. Calculate hex and store low-byte-first as #5.

Fill in #7 and #8 with the fixed values indicated, and you've got your superblock. Use the monitor or absolute writer to put this block at the beginning of superblock's block, and you should be able to *fsck* the partition. In my case, the partition was 17336 blocks (hex B8 43), had 5792 inodes (hex A0 16), and the starting data zone came to BB hex.

Once you write out the corrected superblock, you're almost there. Boot on the bootable floppy, then run *fsck /dev/hd2* - because hd0 is the whole drive, and hd1 is the little boot partition - and you should end up with a clean file system. I did!

Another Tale of Woe

For some folks, a bridge is something you walk or drive over, and a router is a tool used to make scroll work in wood. But we in the computer world have our own crazy vocabulary. I think I'm the only person in my neighborhood with a LAN in the basement. In fact, it's sort of two LANs, one thinnet Ethernet and one Token Ring.

At any rate, I strove to bring my OS/2 2.0 system up on the Netware LAN under Token Ring. It was refused admission! This situation persisted for days. Yet the machine was accessible via NetBIOS, so it couldn't be a hardware conflict ... right? Of course the only error was "Unable to get connection ID", one of those "Something is wrong" type of messages. (On another topic, the only error message in MSCDEX is "Incorrect DOS version" - which it says no matter what is wrong.)

Finally, however, I decided to dust off the little test utility that comes with the Token Ring boards. Nobody but a paranoid person like me even *keeps* these diskettes, because they do almost nothing. Well, this utility came up with the interesting fact that, while the board worked fine, it had to move the RAM buffer to CC000, because there was a conflict at DC000 where most software, including the LAN Support Program, expected it to be!

The manual on this board makes no mention of the RAM buffer address needing to be at DC000 or anywhere else. In fact, it never mentions the I/O address either, which is 220-227 hex *and* A20-A27 hex. So, I had a conflict with the SCSI board, which I had moved because of a conflict with the graphics board, which I had moved because of a conflict with the Token Ring board... Just exactly where are these "holes in the memory map" I keep hearing about?

Back in the Z-80 and CP/M days, we'd have "banked out" all of these video and network boards, and they'd only be "switched in" when they were actually being accessed. There'd never be any conflict. Keep reminding me how we've made progress since then.

Well, to make a long story short (which Ed. will appreciate), the graphics board is out and all the conflicts are resolved, and the OS/2 2.0 machine is now a proud constituent of the LAN.

Next time

Next time we go Back to Basics to explore the fundamentals of Minix, Coherent, and other Unix look-alikes. Plus, we discover the Mother Load - the source of much code in seedy Rome. In the meantime, may you never *long* for an *int* that's not *short.*

Where to call or write

BBS or Fax: 1-703-330-9049

Sending Articles to *TCJ*

Send your articles and letters of inquiry to:

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

The editorial policy is to seek articles that can enhance and educate our readers. Letters of interest will be printed in our Reader To Reader section on a space and topic consideration. Material is typically printed "as is", however *TCJ* does reserve the right to reject or modify (by omitting) portions of letters or articles deemed unfit for publication. Any letters received by *TCJ* or it's technical editors may be printed or included within an article unless **YOU** indicate otherwise. Your name and city/state only will be used unless **YOU** indicate that you desire to have your full name and address included in references or letters printed.

Major letters and minor articles are accepted on floppy disk or by network services and will aid in getting your letter published "as is." *TCJ* does NOT return disks and material unless suitable and appropriate return mailers and postage is provided.

Floppy disk and word processing formats support by *TCJ,* are 3.5, 5.25, and 8 inch disk formats. Several CP/M to PCDOS conversion programs are used to transfer data for editing under WordStar with final output under PageMaker 4. Please do not use embedded punctuations in file names which can prevent reading by transfer programs. WordStar 7 can read and convert most other word processing programs output, but providing at least one ASCII file is recommended. Use of GENIE (as B.Kibler) and CompuServe (ID: 71563,2243) is the preferred method of sending information and articles to *TCJ.* Please ZIP files with a READ.ME, your article and an ASCII text version included.

TCJ is currently looking for articles that show our readers how you are still using older systems. Those systems can be anything except PC clone machines (machines like MBC 550 which are NOT 100% compatible are OK!). Your article should be written as if you are talking among friends and recounting your experiences. Please make it clear that "YOU" did this, and "I" had these problems, which "I" was able to resolve using these steps and techniques. The readers level of knowledge ranges from beginner to advanced. All references should provide a brief review of information to assist readers in determining the importance of the reference in relation to their own needs.

TURNKEY APPLICATIONS DEVELOPMENT IN FORTH

Special Feature

Intermediate Users

Forth Operations

by Frank Sergeant

How do you develop a turnkey application in Forth? That is, after you have developed your application, how do you make it run as a stand-alone program? After all, forth is strange. If you are used to other languages you may keep looking for the compiler and linker and never find them. How do you wrap up the application so you can ship it to a client or use it yourself from the DOS command line?

Let's divide the universe of possible applications into those that run on general computers and those that run on dedicated computers (''embedded'' systems). The two need to be handled somewhat differently. This article concentrates on the first class.

General Computers

Suppose you write a calculus tutor in Forth that you hope to distribute on a disk. You'd like the customer to be able to do a directory listing of the disk, see your executable file, type in its name at the operating system prompt, and have the calculus tutor come up, but _not_ have the Forth system come up. Right? I'll describe how to do this in Pygmy Forth.

You must do five things:

1. Write the application and test it.
2. Create a special startup word that initializes and runs the application.
3. Create a special error handling word that prints error messages and restarts the application.
4. Install the new boot and abort words.
5. SAVE the .COM file.

Step 1. Write the application.

Hopefully you write the application incrementally, testing each piece exhaustively from the keyboard, thus making a very sturdy system. It culminates in a single high-level word, perhaps TUTOR. All this tested code resides in block files (ok, you could use text files instead if you insist). So, load the application by typing 4001 LOAD or " TUTOR.TXT" FLOAD or whatever. See the listing at the end of this article. Blocks 4002, 4003, and 4004 show an extremely simple calculus tutor program.

Step 2. Define the startup word.

Next write your own startup word which handles any special initialization, such as opening files, loading in a table from disk, or setting screen colors. It must finally execute the main word of your application. Suppose you name your startup word TUTOR-BOOT (defined on block 4005). We want to arrange things so that Pygmy will execute TUTOR-BOOT when it starts up, rather than displaying a greeting and waiting for you to type Forth commands. The word BOOT is to Pygmy more or less what AUTOEXEC.BAT is to MS-DOS. BOOT is a DEFER'd word which points to another word which is actually executed whenever Pygmy begins. We won't do it yet, but later we make BOOT point to TUTOR-BOOT by typing

```
' TUTOR-BOOT  IS  BOOT
```

This will ''vector'' BOOT to TUTOR-BOOT, which initializes things and then executes TUTOR. The default in Pygmy is for BOOT to execute the word (BOOT. Use (BOOT as a model, if you like, when designing your own startup word.

Step 3. Define the custom error handler.

What happens, though, if an error occurs? The word ABORT will print an error message, clean up the stacks, and execute the Forth main loop QUIT. This is perfect during development but it is not what you want the final application to do. You probably want to print an error message and then either restart the application or exit gracefully. Fortunately, ABORT is also a DEFER'd word. It normally executes the word (ABORT, but you can write your own routine instead, as described above for BOOT. Let's call the custom error handler TUTOR-ABORT (defined on block 4005). Typing

```
' TUTOR-ABORT  IS  ABORT
```

would install the new error handler in place of the default (ABORT, but don't do it yet.

In order to exercise our error handler, we need an error. So, in block 4004, the definition of TUTOR contains a test for the ''error condition'' of the user pressing the Enter key instead of a digit. If only this were the most serious error we needed to guard against.

Step 4. Install boot and abort words.

OK, you can do it now. That is to say, during the development process you will leave BOOT and ABORT alone. Then, when you have finished testing, the last thing you do before saving the .COM file is re-vector them to your custom words by typing

```
' TUTOR-BOOT  IS  BOOT
' TUTOR-ABORT  IS  ABORT
```

as shown on block 4001.

Step 5. Save the .COM file.

Finally, as shown on block 4001, save the system by typing

```
SAVE TUTOR.COM
```

Type BYE to get back to DOS and run your new turnkey application by typing TUTOR at the DOS prompt. The saved file (i.e. TUTOR.COM) is your turnkey application. When you or your customers run it the underlying Forth is never seen because BOOT executes TUTOR-BOOT. Then TUTOR-BOOT executes TUTOR. As TUTOR-BOOT's final step it execute BYE to return the user to DOS.

Snapshots

As you are developing in Pygmy, but before you are ready to produce the shippable application, you can take a "snapshot" of your system at any time with the word SAVE. When doing this, there is no need to re-vector BOOT or ABORT. Just type

```
SAVE TST1.COM
```

Then, when you later execute TST1.COM from DOS, the extensions to the dictionary will be there without needing to be reloaded. The files open at the time of doing the SAVE will be opened automatically.

Getting Fancier

The approach described above for making a turnkey version of your application is very easy to do and works great. The .COM file is guaranteed to be less than 64K bytes long. In this day of bloated applications, that's not considered very large. Of course, depending on the size of your application, the file might be much smaller than 64K. However, there are some additional steps you can take to make the .COM file smaller. You can make many of the words headerless. You do this a word at a time by preceding each word with | (a vertical bar), or a whole group of words at a time by surrounding the group with HEADERS-OFF and HEADERS-ON.

Another trick is to use just the kernel of Pygmy (around 8K). Don't load the editor. Don't even load the assembler unless your application defines new CODE words. Do this as the last step, because you want the editor present during development but not in the final application.

If you need the assembler, but do not want it taking up space in the final application, you can load it "high" by using curly braces around it, ie { 112 132 THRU } then load your CODE words (they need the assembler), then unlink the assembler with the word PRUNE. For example, you might start with the Pygmy kernel (about 8K bytes) and use the following load block, assuming your application uses several code words on blocks 6002 through 6008:

```
{ 112 132 THRU }      ( temporarily load the assembler)
HEADERS-OFF            ( optional)
6002 6008 THRU        ( define the CODE words)
6010 6020 THRU        ( load rest of application)
' NEW-BOOT IS BOOT    ( startup word)
' NEW-ABORT IS ABORT  ( error handler)
HEADERS-ON            (optional)
PRUNE                 ( throw away the assembler and headers)
SAVE NEWAPP.COM       ( make turnkey .COM file)
```

Note, we vectored BOOT and ABORT _before_ we threw away their headers!

Dedicated Systems

This section will be a little sketchier. I'll just touch on a few possibilities. I expect to have a lot more to say about dedicated systems in a future article.

Suppose you are building a stand-alone microprocessor gizmo (the target system). The final code will probably need to be burned into an EPROM. Let's assume the host is a PC. If you object to this, please consider my arguments in the last section of this article. During development the PC is connected to the target system by a serial line. Here are some possibilities:

1. The target system is a PC (80x86 processor)
2. The target system uses any small microprocessor
a. with Forth
b. with a monitor ROM
c. with nothing (just RAM you download to, or an EPROM you program)

This simplest situation is if the target system is also a PC, and you are willing to run under DOS, possibly booting from a floppy. Just develop the system as described in the first part of this article and set up an AUTOEXEC.BAT file that loads your turnkey application .COM file. In this case, naturally, you do not need the serial line or the second computer.

However, if the target system is a PC but you do _not_ want to run under DOS, things get a little more complicated. Perhaps you'd like to stick the PC in a cabinet, perhaps without a keyboard, monitor, or even disk drives, to let it control some machinery. You'll have to avoid using any DOS calls (since DOS will not be present). You'll have to write some initialization code so the POST (power on self test) of the BIOS ROM will recognize your ROM and turn control over to it. Even though you plan to burn the code into a ROM, you must decide whether to actually _run_ it out of ROM or out of RAM. I suggest running it out of RAM for simplicity. If you choose RAM, the initialization routine must copy the code from the ROM to RAM, then jump to the RAM to execute it. If you choose ROM, you must alter how Pygmy handles VARIABLEs so they get directed to RAM instead of to memory which cannot _actually_ vary. See the source code for Pygmy's system variables for a hint on how to do this.

If the target system is not a PC, but perhaps some simpler micro running a Forth, you can use Forth on the PC as a smart terminal to talk to the target system's Forth and supply the editor and disk services (and keyboard and video display, of course).

If the target system does not have Forth, but does have a monitor ROM, you can still control it from Forth on the PC. You can either use Forth as a terminal program allowing you to type monitor commands, or you can define various macros to talk to the monitor for you, thus automating some of your typing.

The last choice, a target system without even a monitor ROM, presents an interesting problem. The Motorola 68HC11 microcontroller is excellent for this because it has a special bootstrap mode. In that mode, it starts up in a loop waiting for you to download a program to its RAM. So, although _you_ may not have put a ROM on the target system, the microprocessor itself has a tiny boot loader ROM built-in. There is already an 'HC11 assembler that runs on Pygmy on a PC that generates code for the 'HC11 (it's on the Pygmy Bonus Disk). So, assembly language development is easy enough with this combination. Also, I described a simple monitor program for the 'HC11 in the 1991 FORML Conference Proceedings (available from the Forth Interest Group). For other microprocessors, the trick will be to get _something_ running so you can get some feedback so you can start to figure out what's going on. More on this later.

Frank uses Forth for everything from writing business software to controlling hardware. He can be reached at F.SERGEANT on GEnie or fs07675@swtexas.bitnet, or c/o Famous Math-

Why a PC Makes A Good Host

PCs are cheap but sturdy. You can buy surplus original IBM PCs for less than dumb terminals for CP/M machines used to cost (just to try to put things in perspective). PCs are plentiful. Just like the industrial revolution, they use interchangeable parts, and there is lots of competition by makers of those parts, which keeps prices down. Monochrome graphic video boards are easily available for under $20, for example. I don't say PCs make the only possible host (I certainly would *_not_* say that in this magazine!). I think I've seen surplus original PCs available for under $150, with monitor; XT motherboards for around $30. It's something to keep in mind. Frank Sergeant.

This is getting to be a hard topic to talk about these days. Yes, the cost of PC Clones is dropping and there can be little price difference between an OLD CP/M system (typically $50 with all the software) and PC/XT clones. Where I differ is how you use it. Is it a development system using purchased software or your own developed utilities. If you only buy software, then PC Clone machines are probably good deals (watch out however, most new packages will not run on older PC/XT machines).

If you generate your own support, then CP/M machines may be better. The main difference is whether you use the machine as an appliance or play with it as well. One also needs to check on how complex your application is. I find the tools available with Pygmy and FPC currently superior to F83 for CP/M or PCDOS. Those superior tools will work on most PC/XT machines (FPC normally needs a hard disk, but can be made to work from floppy) and make great cross-development platforms. Programs written for F83 however can run on either platform (CP/M, CP/M68K, or PCDOS) without modification or hard disks if properly done.

The direction at *TCJ* is to provide code that can be used on as many different platforms as possible, and thus leave the choice up to you and your pocketbook. We are also considering supporting older PC based machines (such as Sanyo MBC 550) as these are not true clones and currently are not supported by any magazines. These non-clone machines all typically have 128K or 256K of memory and can run only the smallest of programs (great for Pygmy programs). Bill Kibler.

Your comments on this subject are always welcome! Please send us your thoughts and ideas for a special article on HOST Systems "TO PC or NOT to PC?"

ematicians Academy (hey, it works for artists). Pygmy v1.4 is available from FIG, ftp at asterix.inescn.pt (evenings), etc.

```
Block 4000
Example code for turning a simple application into a
turnkey .COM file.

Steps:
1. Write the application and test it.
2. Create a special startup word that initializes and
runs the application (see TUTOR-BOOT).
3. Create a special error handling word that prints an
error message and restarts the application
(see TUTOR-ABORT).
4. Install the new boot and abort words.
5. SAVE the .COM file.


Block 4001
( Load block for a simple application, a Calculus Tutor)

4002 4005 THRU

' TUTOR-BOOT  IS  BOOT
' TUTOR-ABORT  IS  ABORT

SAVE TUTOR.COM


Block 4002
( Rudimentary Calculus Tutor)

: QUESTION ( -)  ." What is 2 + 2 ? "  ;

: ANSWER ( - key)  KEY  ;

: RIGHT? ( key - f)  '4 =  ;

: ?CONTINUE  ( -)
CR ." press a key to continue"  KEY DROP  ;


Block 4003
( Rudimentary Calculus Tutor)

: REWARD ( -)
CR ."      Congratulations, you have been accepted by the"
CR ." Famous Mathematicians Academy to study for a career"
CR ." in mathematics.  Please send $1200.00 cash to"
CR ."          Famous Mathematicians Academy"
CR ."          809 W. San Antonio Street"
CR ."          San Marcos, Texas  78666"     ;

: CONSOLATION ( -)
CR ."      Ur, uh, not quite, but you are showing potential."
CR ." Please try again soon."  ;


Block 4004
( Rudimentary Calculus Tutor)

: TUTOR ( -)
QUESTION  ANSWER ( key)
DUP 13 = ABORT" You entered a carriage return."
CR ( key)
RIGHT? ( f) IF  REWARD  ELSE  CONSOLATION THEN
CR CR ?CONTINUE   ;


Block 4005
( Custom BOOT & ABORT routines)

: TUTOR-BOOT ( -)
$1F ATTR ! ( set colors to white on blue for CGA or VGA)
TUTOR      ( ie run application)
BYE        ( ie return to DOS)  ;

: TUTOR-ABORT  ( -)
>SCR ( just in case)
BEEP CR ." Utoh, utoh, a serious error has occurred: "
CR POP POP TYPE$ ( ie print the error message)
SP!  RP!  CR     ( ie reset the stacks)
?CONTINUE
TUTOR-BOOT ( ie restart the application)  ;
```

Names used in the article that are likely to be trademarked: Motorola, MS-DOS, IBM, CP/M

Help TCJ BY RENEWING EARLY

Please check your mailing label and renew early, this saves on mailing you a bill and possible interruptions in your subscription!

Don't forget to mail those change of address cards early (at least six weeks).

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

TCJ Center Fold

Special Feature

All Users

IMSAI MPU-A

This issue's center fold is the IMSAI MPU-A S-100 8080A CPU board. The description and schematic are from the original manuals produced in 1975. Except for my one reference to see a note, this is the original text supplied with the board. The note and changes to the schematic represent solutions to problems I personally encountered some years back. If you encounter problems bringing one of these units up, pay close attention to the note and check your other boards for compatibility with all S-100 lines used and unused. Bill Kibler.

MPU Revision 1

FUNCTIONAL DESCRIPTION

The MPU-A board is the processor board for the IMSAI 8080 Microcomputer System. It is designed using the Intel 8080 micro-processor chip. The bus arrangement and board connector has been chosen to be 100% compatible with the MITS Altair M8800 Microcomputer system so that all boards are 100% interchangeable between the Altair system and the IMSAI 8080 system.

Every effort has been made to keep the desiqn simple and straight-forward to maximize reliability and ease of maintenance. MSI and LSI are used where appropriate, and discrete components are held to a minimum for greater circuit reliability and ease of assembly.

The 8224 clock driver chip and an 18 Megahertz crystal are used to generate the 2-phase, 2 Megahertz non-overlapping clock for the 8080A. An 8212 is used as a latch for the status signals and two 8216 tri-state bi-directional bus drivers are used to interface the 8080A with the IMSAI 8080 input and output data buses. All other address, status, and control lines are driven by tri-state bus drivers.

Unregulated +16, -16, +8 volts, and ground must be supplied to the bus. On-board regulation is used to arrive at the power, supply levels needed to run the chips. Integrated circuit power regulators with overload protection are used. The board is supplied with ample bypass filtering using both disc ceramic and tantalum capacitors.

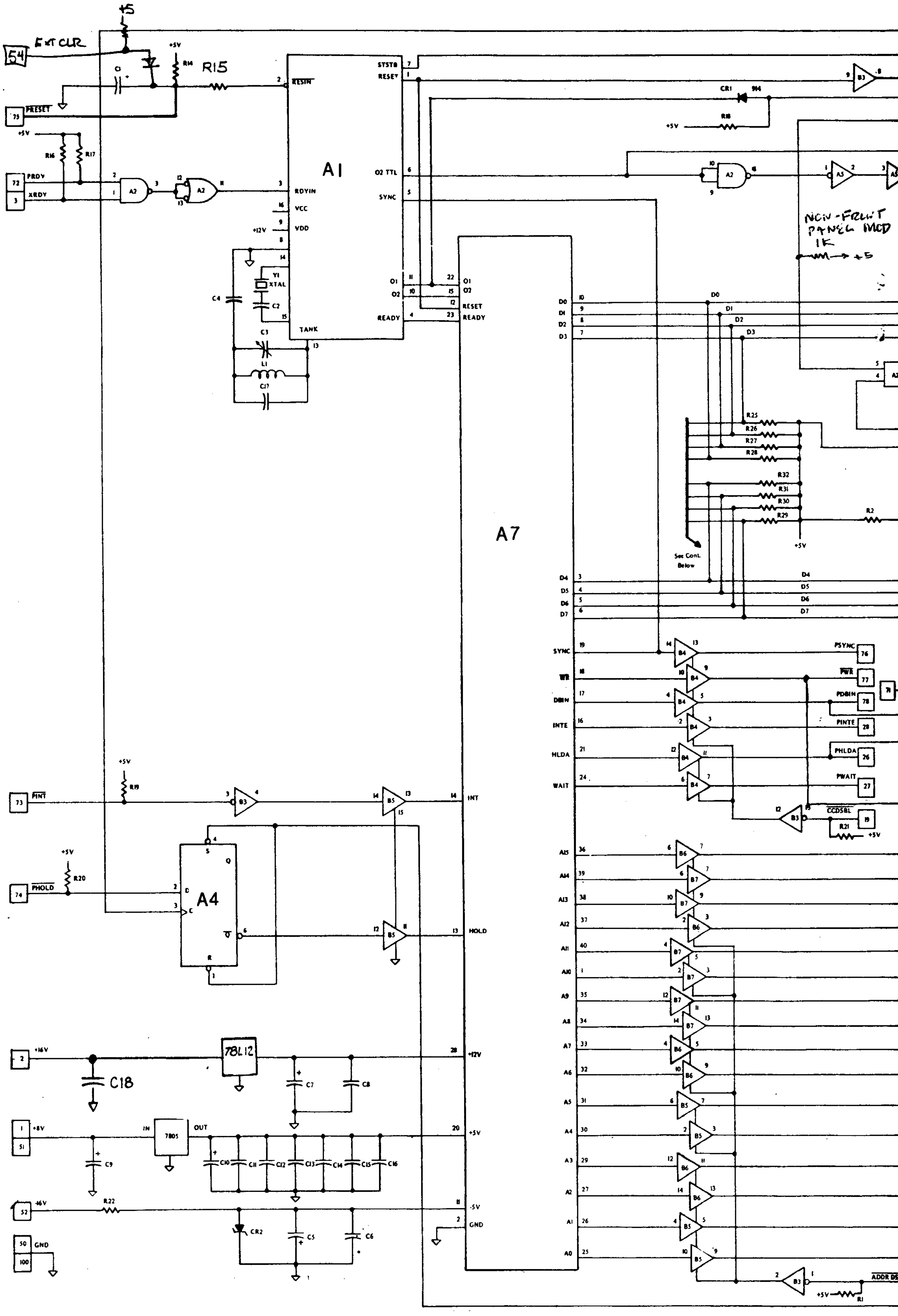
The board connector is a 100 pin edge connector on .125 inch centers 50 pins on each side. Dimensions are 5 inches by 10 inches, using 2 sided glass reinforced epoxy laminate, with plated feed through-holes to eliminate the need for any circuit jumpers. The contact fingers are gold-plated over nickel for reliable contact and long life. All other circuitry is tin-lead plated for better appearance and more reliable solder connections.

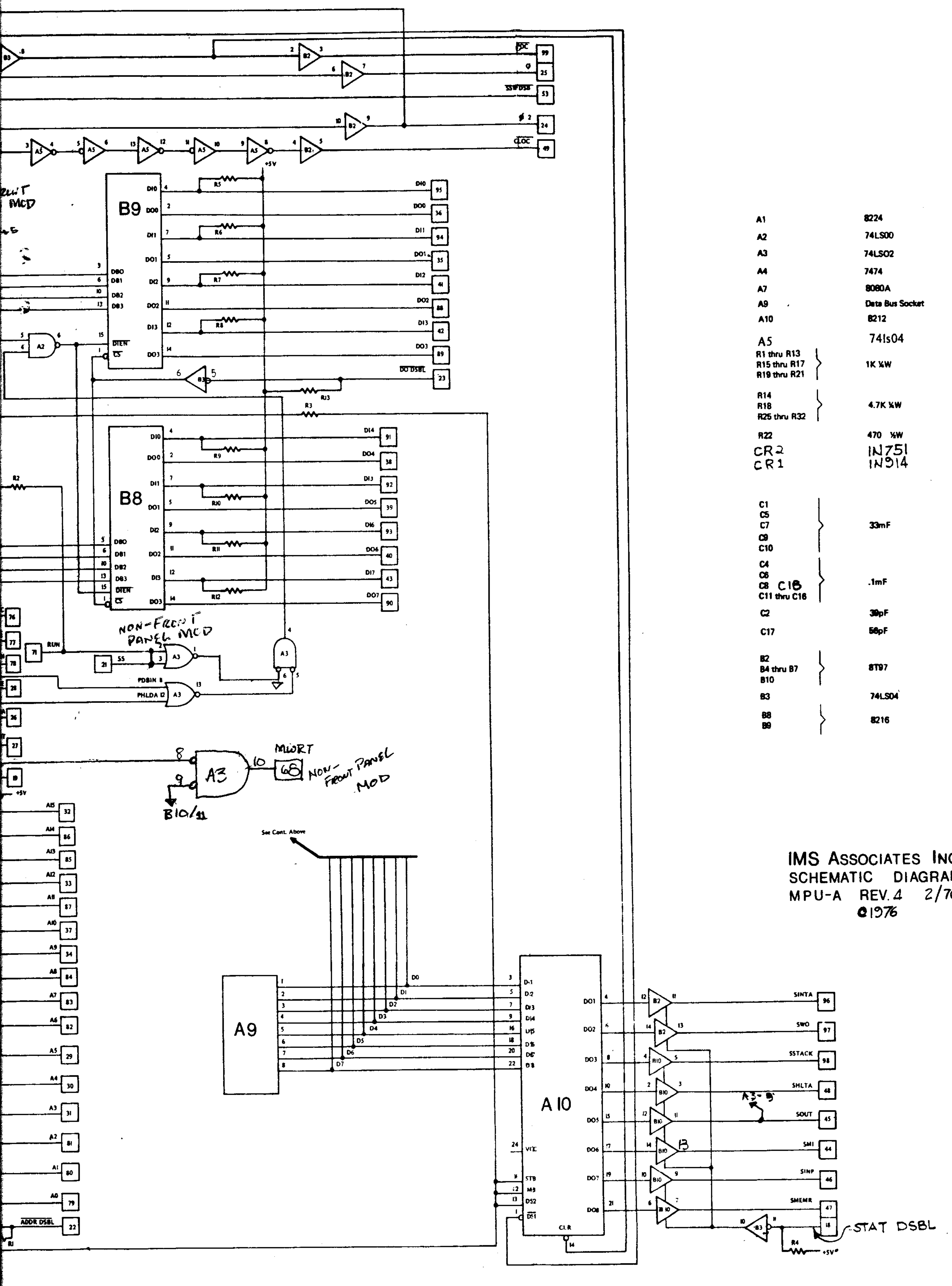
Power-on reset is included on this board along with pull up resistors for all inputs required so that with the front panel removed from the IMSAI 8080 machine, the power-on reset will start the program at position O out of a ROM. All other necessary conditions are met so that the system will run without the front panel attached *(see notes BDK)*, for use in dedicated controller applications where no operator-processor interaction is desired.

THEORY OF OPERATION

The IMSAI MPU-A board is structured around the Intel 8080A microprocessor chip, and much of the MPU-A board is wired to support the 8080A device. The MPU-A board provides interfacing between the 8080A chip and the data and address busses, clock and synchronization signals, and the voltage regulation necessary for the 8080A and other chips. The internal functioning of the 8080A is thoroughly described in the Intel 8080 Microcomputer System User's Manual. Reference should be made to this manual for Information concerning the operation and use of the 8080A.

The address lines from the 8080A drive the address bus on the back plane through 8T97 tri-state buffer drivers. These drivers may be disabled through the ADDRESS DISABLE line on pin 22 of the back plane. Intel 8216 bi-directional bus drivers connect the 8080's bi-directional data bus to the back plane's dual uni-directional DATA IN and DATA OUT busses. The


+5
EXT CLR
54
+5V
R14
R15
C1
PRESET
75
+5V
R16
R17
PRDY
72
XRDY
3
A2
A1
RESIN
RDYIN
VCC
VDD
+12V
XTAL
Y1
C4
C2
C3
L1
C17
TANK
STSTB
RESET
02 TTL
SYNC
01
02
READY
CR1
914
R18
A7
NON-FRONT PANEL MOD
1K
+5
D0
D1
D2
D3
D4
D5
D6
D7
R25
R26
R27
R28
R32
R31
R30
R29
R2
See Cont. Below
SYNC
WR
DBIN
INTE
HLDA
WAIT
PSYNC
PWR
PDBIN
PINTE
PHLDA
PWAIT
CCDSBL
R21
A15
A14
A13
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
A1
A0
ADDR DSB
R1
PINT
73
R19
B3
B5
INT
HOLD
PHOLD
74
R20
A4
+16V
2
C18
78L12
C7
C8
+12V
+8V
1
51
IN
7805
OUT
C9
C10
C11
C12
C13
C14
C15
C16
-16V
52
R22
CR2
C5
C6
-5V
GND
50
100



A1	8224
A2	74LS00
A3	74LS02
A4	7474
A7	8080A
A9	Data Bus Socket
A10	8212
A5	74ls04
R1 thru R13, R15 thru R17, R19 thru R21	1K ¼W
R14, R18, R25 thru R32	4.7K ¼W
R22	470 ½W
CR2	IN751
CR1	IN914
C1, C5, C7, C9, C10	33mF
C4, C6, C8 C18, C11 thru C16	.1mF
C2	39pF
C17	56pF
B2, B4 thru B7, B10	8T97
B3	74LS04
B8, B9	8216

IMS Associates Inc.
SCHEMATIC DIAGRAM
MPU-A REV. 4 2/76
©1976

direction of data transmission is determined by the DIRECTION ENABLE line. The DIRECTION ENABLE line is in turn controlled by the front panel and the processor status signals DATA BUS IN and HALT ACKNOWLEDGE. The 8216 can be disabled by the DATA OUT DISABLE line on pin 23 of the back plane.

The 8080A's bi-directional data bus is also connected to the data bus socket and the 8212 status byte latch. The data bus socket is used to connect the front panel to the bi-directional bus, while the 8212 latch transfers the status byte to the back plane via 8T97 drivers. These drivers are disabled by the STATUS DISABLE line on pin 18 of he back plane. The 8212 is latched up by the STATUS STROBE signal of the 8224 clock chip to store the status information for each instruction cycle.

One K pullup resistors to +5 volts are connected to all the bi-directional bus lines to ensure that during the time the bus is not driven, the 8080A reads all 1's.

The 8224 clock chip and crystal oscillator, provide the two-phase non-overlapping 2 megacycle system clock for the 8080A. These clocks are also driven onto the back plane through 8T97 tri-state buffered drivers.

The CLOCK line on the back plane is driven from the TTL Phase II clock line through a delay so that the phase relation of the clock signal to the Phase II and Phase I back plane signals, is nearly identical to that produced by the MITS Altair 8800 system. Six sections of a 7404 are used for this delay to provide greater simplicity and higher reliability than a one-shot. The 8224 chip also provides the power-on reset function through use of a 4.7K resistor and 33 uf capacitor connected to the reset input of the 8224. The power-on reset is applied to the 8080A and is applied to the POWER ON CLEAR line, pin 99 on the back plane.

The two BACK PLANE READY signals are ANDed and connected to the 8224 for synchronization with the Phase II clock before being connected to the 8080A chip. The INTERRUPT line is connected directly to the 8080A, while the HOLD REQUEST line is synchronized with the Phase II clock and then connected to the 8080A.

The six processor status signals (SYNC WRITE, STROBE DATA BIT IN, READ STROBE, INTERRUPT ENABLED, HOLD ACKNOWLEDGED, and WAIT ACKNOWLEDGE) are all driven onto the back plane through 8T97 tri-state buffered drivers. These drivers may be disabled by the CONTROL DISABLE line, pin 19 on the back plane.

The +5 volts is regulated from the +8 volts by a 7805 integrated circuit regulator, while the -5 volts is regulated by a 5 volt zener and a 470 ohm resistor from the 16 volt bus. The +12 volts is regulated by a 12 volt Zener and connected to the +16 volt line by two 82 ohm 1/2 watt resistors in parallel. All voltages are filtered with .33 microfarad tantalum and disc ceramic capacitors.

Note: These boards were made for front panel use even though the description says otherwise. The problem noted in the schematic is the lack of MWRT, a signal indicating a memory write is in operation. Not all boards need or use this signal. I encountered this problem when attempting to interface to later versions of S-100 boards that relied on this signal. The signal was generated however by a front panel circuit. The front panel contained switches to single step the CPU as well as providing other control signals and pull ups for lines not supported by the CPU card. An extra pull up resistor and jumper were also added for similar reasons, but the exact number and type of changes needed will depend solely on the other type of cards used. In bringing up these older boards, you must check out ALL control signal lines on all cards used. Not all vendors used all lines as intended or as specified later in the IEEE 696 standard. BDK.

TCJ Center Fold

The Computer Journal solicits reprintable schematics and documents for use in the center fold section. We prefer schematics that have been redrawn and reproduced using high quality systems (CAD programs and laser printers). Many older schematics will not reproduce due to too small of print and folds that make pin numbers unreadable. Minor changes are acceptable if followed with a description of the why the schematic changes were needed and what problems were resolved. Descriptions on disk of what and how the circuit works is desired but not always required. Send all enquiries and drawings to *TCJ*, PO Box 535, Lincoln, CA 95648.

Planned Center Folds are:

- Big Board Z80 (Xerox and Kaypro early design).
- IMSAI PIO and SIO S-100 boards
- CCS 2422 disk controller
- SD systems SBC300 single card Z80 S-100 controller
- GIMIX 6809 CPU and Floppy DISK controller card for SS-50 bus

Dr. S-100

By Herb R. Johnson

Regular Feature

Intermediate

Vendor Review

Greetings

A good new year to you all! I just returned from Washington DC, where I wandered over to the Smithsonian collections of surplus computers, spacecraft, and so on. As the catalogs say, there was "too much to list here!"; I'll cover some of S-100 equipment (!) they have on display next time.

This month, after answering some mail about S-100 and about the **Z180-PC** proposal, I'll discuss some of the major vendors of the S-100 equipment you are likely to find in your (or someone else's) basement. Again, there are too many vendors to list, so I'll wrap up with some "generic comments" on evaluating an S-100 system you might come across.

Letters

John Haugh of Shorewood WI, a collector of **Cromemco** equipment, offers his services to help other users of this line of S-100 computers. "After sending computers to my kids to help them in college, I expect to build several Cromemco's for my grandchildren!". His systems range from simple CDOS boxes to full Unix systems, all pure Cromemco. Give him a call if you need a little help.

Paul Herman of **Paul F Herman Inc.** wrote to congratulate me on this column. He publishes **Z-100 Lifeline** for the Heath/Zenith Z-100 crowd, and sells a variety of hardware and software for this dual-processor IEEE-696 system (which I'll describe later in the column). His newsletter plug hopes we cover Z-100 equipment: I'd encourage him and his readers to send me whatever they'd like to see: this column is not a sole-source effort!

Paul offers a SCSI controller for $210, with software for the Z-100 (only) for $39.00 and its source code for $39.00 more. If the TCJ readership is interested, I could ask Paul to loan me a card and I'll see if it can be used for non-Z100 systems. Paul has been in this business for many years, and the Heath crowd is a very loyal group, so keep Paul (and Heath/Zenith) in mind. We must note, in passing, that Heath Co. is finally dead: after several acquisitions, the back stock of all replacement parts were auctioned off in December and no Heath products are for sale.

Got a long letter about S-100 systems from **David Drew** of Newark DE. He was kind enough to write on his personal history of building, debugging, and trading. Starting with ExpandoRAM (by Morrow, I think?) and SD Systems cards in 1979, today he has a N* (an abbreviation for NorthStar) and other unnamed systems with "monstrous" 14-inch hard drives. Thanks for the letter, David: mind if I publish it as a "testimonial"?

The Z180-PC: What about MY Computer?

TCJ editor **Bill Kibler** asked last issue if anyone was interested in our idea for a Z180-based IBM-PC bus compatible card, that would "talk to" other IBM-PC cards and thus replace the 8088-based motherboard. I've only had a few responses so far. One exchange came from TCJ advertiser **Bill Roch** of **Elliam Associates**. We discussed a Z180 design upgrade for the British **Amstrad PCW**. While it's not an S-100 system, it is a Z80-based machine that is popular in Europe and has a following here in the USA. We kind of decided it would not be cost-effective to compete with the upgrades that already exist (a common consideration!) but it gave me a chance to see the "guts" of a CP/M machine with ASIC's (i.e. big custom IC's) instead of TTL (i.e. a single custom IC instead of lots of little stock logic chips.)

I'm hesitant to pursue a "Z180-PC" without more reader feedback. I don't want to compete with other Z180 vendors (who probably need the business, particularly if they service us hobbyists), nor try to build another kind of PC, which will scare away people who already find the IBM PC and MS-DOS "too complicated". Finally, how do I deal with even trying to support many of the cards out there and keep my day job? As users, you'll have the same problem (or challenge, depending on your priorities).

A simpler and cheaper solution may be a simple processor card with a few devices supported, with the ability to daughter-card support hardware for other devices. (A "daughter card" attaches to a circuit board via a short jumper cable or a simple connector. Often it is designed well after the original product is in production as an afterthought.) Actually, I've had more correspondence for **hard drive controllers** adaptors than any other development project. A simple card could do this for several machines. No feedback, no projects: readers, it's up to you!

S-100 Manufacturer's Survey: The Way It Was

Surprisingly, there are still a few vendors building IEEE-696 cards. I believe Compupro (sometimes called Viasyn) and Cromemco are still selling systems; maybe they build a few cards when the stockroom is empty. However, these are 80386 systems for **thousands** of dollars, no schematics provided, etc. In short, **not for hackers!** They still charge hundreds of $$ for old cards too, so there is no reason for you to call them. Keep in mind that some users of these systems are still in business, doing business things with these computers! Consequently, the system vendors are still able to **command original price** since system users must either **pay or upgrade**! So, you can try to call the original vendors, but don't expect more than (maybe) free documentation.

"For the rest of us," we must get our systems and cards used. You may find yourself in a position to even **choose a system** amongst several at a flea market, or at a surplus outlet. So many systems, so little money! So this month, let's walk down memory lane with the Ghost of Computers Past for a tour.

IMSAI and **Altair** systems are among the oldest S-100 systems built. Early versions had the front panel for "convenient" access to address and data lines for debugging and programming. In the old days, peripherals were **unavailable** at any sane price so blinking lights and toggle switches were themselves a miracle! Later systems became more "personal", even "PC" like. **Positives:** potential antique value, "classic" systems. Front panels seem popular and useful to some. **Negatives:** 8080/8085 class only, minimal functions per card, parts **very** obsolete.

Compupro systems vary from simple 8085's and Z-80's to 68000's and 80386's. Their static RAM cards vary likewise in size, and they had a variety of peripheral controllers for floppy and hard disk of all ages. Note: their later systems used cards without voltage regulators, relying on bus power for +/-5 volts and for +/-12 volts. **Positives:** their cards are well-marked, professionally built and laid out. You can look at any card and, if you know chip functions by name, figure it out. **Negatives:** some of their cards went through a lot of revisions, and early revisions were flaky (e.g. disk controllers).

Cromemco systems also cover a range of technology. They produced a number of IMSAI-class S-100 cards and systems, including a copycat (relabeled?) IMSAI. Later they also went IEEE-696 and built some fancier systems that competed with IBM-PC's for quite a while (as did Compupro). **Positives:** well-manufactured cards. Docs were complete and readable. **Negatives:** incredibly heavy mainframes!

The **Heath/Zenith Z-100** was "strongly" derived from the Compupro 8/16 design, which also uses an 8085/8088 processor pair. The motherboard has 6 IEEE-696 slots, the dual processors as mentioned, 172K of RAM (3 X 64K), serial and parallel ports, and a daughter board color/mono video card. The slots held a disk controller (5" and 8" drives supported), and the case had two 5" floppies or one floppy and one hard drive which required a pair of controller cards, one for the slot and one on the drive. One model included a mono monitor as part of the unit.

Positives: Docs, docs, docs! and current support from a number of third parties. Also, a lot of these were sold to the military, and are now appearing in surplus! **Negatives:** no Z-80 capability. The units with internal monitors are a **bear** to ship! I broke two CRT's at the tube socket (the glass nipples pop off). Write to me for details.

Northstar systems were popular as business systems, especially with their optional wooden cover. I don't think they built to IEEE-696, and the Northstar Horizons I've seen were all based on **hard sectored floppy disk controllers!** Still, they also have had a loyal following. By the way, they also built a non-S-100 system in a terminal-like package called the Advantage. **Positives:** seem reliable. **Negatives:** hard-sectored floppies.

Ithaca Audio or **Ithaca Intersystems** was another long-standing vendor. They built the **prettiest front-panel** system of all! Their docs are OK, and their cards all seem reasonably functional. I especially like their 8K static RAM cards, covered with 64 2102's. They often work at 3 or 4 MHz, and were a design copied by many others in the old days. **Positives:** availability. **Negatives:** none other than the ravages of age and old designs.

Generic comments

About identifying cards in general.... Know your chips! If you can identify floppy controller chips, serial chips, parallel chips, memory chips, and so on by name, you are **halfway to a full understanding of any computer card!** Also know S-100 cards by **shape and size**, and know how to **count edge connector pins**. Make a full-scale card from cardboard and mark off the pin functions. Use these valuable and vital clues to determining function and worth of any card you find. You can learn about old chips by reading books and docs on older systems and noting the common chips: check your local library, your friend's basements, etc. (If there is interest, I'll create a list: call or write for details!)

Specific examples: Avoid dynamic RAM cards if you can, especially cards built of 16K or 4K DRAMS. The older the floppy disk controller chip, the less reliable and flexible the card. Look at bus pins 20 and 70: if they are shorted to ground, the card is not designed to work with a front-panel (IMSAI/ALTAIR) system (but it might if you cover those pins). Check pins 61 through 64: (address lines A20-A23) if they are in use, the card is probably near IEEE-696.

The **Ultimate Hint** about IC's: learn to read **date codes.** They are usually a number for year and week of the year. Examples are **8412** (for the 12th week of 1984) or 732 (for the 32nd week of 1987 or 1977: you decide). Most boards have chips dated within a year or two. The boards themselves will have a copyright date too: this should help.

Finally, for a detailed tour of vendors and cards, you might write or call for my **catalog**, which lists cards by vendor name. Send $1 plus SASE for the latest on these late systems!

Next Time

What is **your** favorite vendor? Do you have something good or bad to say about S-100 manufacturers or their products? Send your comments to me directly, and if the volume warrants I'll discuss it next time. Alternatively, I'd like to answer the simplest question about S-100 systems: **what is the bus**? Why have a bus, how does it work, and so on. The general discussion will apply to all bus-based systems, even those white MS-DOS bricks I've heard about recently, so attendance is open to all.

References

Heath/Zenith Z100 systems: Paul F Herman Inc, 9317 Amazon Drive, New Port Richey, FL 34655. orders: 800-346-2152 other business 813-376-5457.

Cromemco support: John J Haugh, MD, 4205 North Newhall St, Shorewood WI 53211.

Amstrad PCW support, sales: Bill Roch, Elliam Associates, PO Box 2664, Atascadero CA 93423. (805) 466-8440.

To get the full list of all references to date, send $1 plus SASE to me, Herb Johnson, for the ever-updated list of S-100 sources!Love letters are also acceptable. Bill Kibler prefers that correspondence to this column go directly to me, to expedite my response. That address again, **CN 5256 #105, Princeton NJ 08543.** Or call **(609) 588-5316** and ask for "Dr. S-100."

CLASSIFIED, FOR SALE and WANTED

Amstrad (c) PCW SIG: $9 for 6 bi-monthly newsletters dealing with the most popular CP/M machines still in production. Learn where to buy 3" discs, how to add 3.5 and 5.25 drives and where to buy the 8 MHz Sprinter board with room for 4 Meg of RAM. Make checks out to Al Warsh, 2751 Reche Cyn Rd #93, Colton, CA 92324.

For Sale: GIMIX 6809 SS-50 floppy disk controllers. Have six to sell at $25 each plus shipping ($5). Bill at TCJ (800) 424-8825.

WANTED: Modula-2 (Borland Product) for CP/M, originally marketed thru Echelon, Inc. Manual and Disks are needed. Apple II/CPM disk format preferred but not essential. Call Norman Leet at (513) 864-2261 or leave message at Z-node #3 or Compuserve Mail (ID: 70200,144). Snail Mail address: 840 Hunter Rd. Apt L, Enon, OH, 45323.

Wanted Circuits of ADM3A and ADM5 dumb terminals (and with board layouts if possible). I am tempted to modify these to be a CP/M computer by adding to the circuitry. Answers to J. S. Butler, 16 Uphill Drive, London NW9 0BU, England.

Wanted: CP/M Astrology program, contact: Leon Brown, 144 Brewester Rd., Jewett City, CT 06351.

Wanted: Information, manuals, etc. for DATAVUE DV-80 373M2 Multiuser system with 15 Mbyte Hard drive. Unit currently down, needs reformatting of hard drive, maybe boot disk for floppy? Any help getting back running would be appreciated. Alwyn Stockey, G3EKE/W7, P.O. Box 1764, Sisters, OR 97759.

Wanted: Will pay for Cromemco SCC Z80 cards. Need for on going commercial use. Also looking for RS 488 S-100 cards?? Contact M. Schmidt (408) 432-1150 or Mark Tech Lazer Inc. 2211D Fortune Dr. San Jose, CA 95131-1806.

Wanted: Help/Information on making "MAGIC SAC" and "Translator ONE" work with Atari ST. These convert the Atari ST into an MacIntosh. Have version 4.52 and 5.91 of software. Gets to Mac screen then dies and have trouble with disk formats. Also interested in turning "Translator" into CP/M system (has Z180)??? Contact Bill at TCJ (800) 424-8825.

Wanted: Information or where abouts of source code and internal information to "POOR MAN'S NETWORK" by Anderson Techno Products of Ottawa, Ontario, Canada. This is network software for CP/M systems. Have used it with Xerox, S-100, and Superbrains over serial lines. Allows remote and background operation of second computer. I have an official version, are they still in business?? Contact Bill at TCJ (800) 424-8825.

Wanted: Looking for CP/M68K or BIOS code for SAGE/STRIDE II. May be a UNIX version available as well. Bill at TCJ (800) 424-8825

The Computer Journal classified section is for items FOR SALE. The price is based on Nuts & Volts rates. If you currently have a Nuts & Volts ad just send us a copy of the invoice and we will print the ad for the same price.

Classified ads are on a pre-paid basis only. The rate is $.30 per word for subscribers, and $.60 per word for others. There is a minimum $4.50 charge per insertion.

Support wanted is a free service to our readers who need to find old or missing documentation or software. No For Sale items allowed, however exchanges or like kind swapping is permitted. Please limit your requests to one type of system. Call TCJ at (800) 424-8825 or drop a card to TCJ, P.O. Box 535, Lincoln, CA 95648.

Regular Feature

Kaypro Support

Versions of Kaypros

Mr. Kaypro

By Charles B. Stafford

In the last issue, we talked about moving the reset button from the back, to the front panel, where it might be more useful. A similar project with the video brightness control was hinted at. Neither of these modifications are ''model'' sensitive, but some in the future may be, and it might be nice to be able to identify your particular machine for other purposes as well. The following discussion may shed some light on the vintage of your treasured toy.

About the time that Adam Osbourne decided that Apples were too expensive and elitist, and embarked on the enterprise that produced the Osbourne I, Andrew Kay, owner, President and Chief Potentate of Non Linear Systems, Inc. decided that it would be really neat, if the technicians who used his test equipment in the field, had a portable computer they could interface with almost any kind of system. This would make trouble-shooting much easier. Of course, he couldn't predict what systems those technicians might want to work on, so this computer would have to be easily modifiable, have a fairly complete programming language, be easy to transport out into the field, AND come with all the software a technician could ever want !!

Thus was born the KayComp computer. It was a single board design, using readily available components (for repair purposes), had an integral 9 inch green monitor, and was housed in an aluminum case with a suitcase handle that would fit under a standard airline seat. In fact, with minor differences, that original design was used throughout the entire production of CP/M based computers and for the first MS-DOS machines. (For you aficionados, the sole exception is the Robie.)

The name had to be changed early on, because there turned out to be another computer (much larger) that had an earlier claim to the name, and Andrew's child became the Kaypro. The first model had vertical drives, and was only produced for a short time, then came the Kaypro II, which stored 200 kilobytes of data on each of the TWO Diskette Drives! It also had 64 kilobytes of random access memory, (you had to do an expensive upgrade to get that much on an Apple). AND it cost less than $2000, and had more software with it than you could ever use.

Diskettes cost about $2.00 each and the initial decision was whether to buy more than 2 diskettes. How long would it take to fill them up ?

Over the next few years came a succession of improvements, double density diskette drives, a hard drive model, rudimentary graphics, a universal board design, a co-processor (courtesy of SWP), a real time clock, and finally a built-in modem, all housed of course, in a case that looked like Darth Vader's lunch box.

Since they all looked similar, and had similar names, (Kaypro IV vs Kaypro 4) the differences although significant became blurred. The only reliable way to determine what you really have is to remove the hood and look at the ''assembly'' numbers silk screened on the main board and the labels on the monitor roms. The table on the following page lists the ''stock'' configurations as they left the factory.

Fortunately for Andrew's technicians, and incidentally us, all of the Kaypros, except the Robie, have the same aluminum case, with different paint and in the case of the K-10s only one half-height floppy drive space. We are fortunate, because aluminum lends itself to relatively easy, accurate, modifications, such as fans and mounting holes for non-standard power supplies, and doesn't deteriorate significantly with age and ultra-violet radiation like plastic.

Now that your treasure can be accurately identified, repairs and modifications should be much less traumatic and projects much more easily undertaken. May the FORCE be with you!

BITES:

Note the spelling, in this case, the word stands for Benefits, Ideas, Techniques, Experiences, and Serendippities. It is my fervent hope that this area will become a repository for some of the shortcuts I devise, but more importantly, for your contributions. I'm sure that most of you readers have developed sneaky ways to do things that I wouldn't think of in a million years, and I as well as the rest of our readers would love to BENEFIT from your labors. Here's a few I've run into.

PERFECT WRITER and the TurboRom
If you install the TurboRom exactly as the instructions say, you'll never run into this, but if you get excited and can't

wait to try the new operating system, it'll bite you.

Symptoms;
Perfect Writer will run just fine in the "edit" mode, windows and all. BUT, when it's time to print the fruit of your efforts, and you format the file, this message will appear "NOT ENOUGH MEMORY."
I remember that sinking feeling well, thinking that I'd never get a document out of that damned machine. The size of the file didn't make any difference either.

Problem;
The formatter for Perfect Writer is loaded, and then loads the file specified at a certain address above itself, and 'way above that stakes out a "scratch" area. The top of the scratch area is about 57k up, and the initial operating system that is built for the TurboRom is a 56k system, so the program is right, there is apparently not enough memory, at least not that it knows about.

The Cure;
Somewhere around the last two steps in the installation procedure, the directions call for running a program called "PEEK". That small program will tell you what the maximum system size is based on your preferred disk buffer. Then the directions ask you to run MOVTURBO to create the max system. NOTE several of us have found that subtracting .25k from the figure that "PEEK" gives you, saves a lot of headaches later if you make other hardware modifications.

NON-STANDARD LABEL PRINTING

Sometime ago, I found myself with the task of printing 3/4 inch by 1/2 inch labels. I don't remember what the occasion was, a garage sale, or just a wifely desire to label a whole bunch of small boxes, but it was non-computer related, which didn't make me any more inclined to do it by hand. Fortunately these particular labels came in an array 5 wide by 7 deep (down), but even that is a real pain to put in the printer, and you lose the last three rows (BEEP, paper out). It took a while, but I finally

Versions of Kaypro computers

Model	CP/M version	Mainboard	ROM version
K 1	2.2U1	81-294	81-478-A
K 2X/MTC	2.2U1	81-580	81-478-A
K 2X	2.2H	81-294	81-292-A
K 2X	2.2G	81-294	81-292-A
K 2/84	2.2G	81-294	81-292-A
K II/83	2.2F	81-240	81-232
K II/83	2.2F	81-184	81-242
K II/83	2.2F	81-110 (obsolete)	81-149-C
NEW 2	2.2U1	81-294	81-478-A
K 4X	2.2H	81-297	81-326-E
K 4X	2.2G	81-297	81-326-E
K 4/84	2.2H	81-184	81-292-A
K 4/84+88	2.2H	81-184	81-292-A SWP
K 4/84	2.2G	81-184	81-292-A
K 4/84+88	2.2G	81-184	81-292-A SWP
K 4/83+88	2.2F	81-240	81-232-A SWP
K 4/83	2.2F	81-240	81-232-A
K 4/83	2.2F	81-184	81-232
K 10/MTC	2.2U1	81-582	81-478-A
K 10	2.2H	81-181	81-302-C
K 10	2.2G	81-181	81-302-C
K 10	2.2F	81-181	81-302
K 10/83	2.2D	81-180	81-188-N
Robie	2.2G	81-296	81-326-E

note 1: The Robie and the K 4X use Drivetec high density diskette drives storing 2.6mb each. They will also read DSDD.

note 2: The only way to conveniently group storage capabilities of these beast, is by MAINBOARD number and model. The 81-110 board (the original K II/83) was only capable (in it's virgin state) of single sided double density drives. All of the rest are capable (with some effort) of supporting double-sided double density drives. All of these machines except the K-10 series had two floppy drives. The K-10 series had one floppy drive and a 10 megabyte hard drive. A conversion was engineered by the Micro Cornucopia staff to convert the original K II to a K 4 (double sided drives) and ADVENT had a decoder/personality board that allowed two floppies on a K 10. (these will be the subject of future "HOW-TO" articles.)

note 3: "/83" is a model year designation; "/MTC" indicates an onboard modem (300 baud) and a real time clock; "+88" indicates an 8088 coprocessor board installed, strictly generic MD-DOS that used the real Kaypro as a terminal. SWP was the manufacturer of the coprocessor board.

note 4: The K 10 series and the ROBIE, as well as some "late model" dual drive machines had 85 watt power supplies. The rest had 65 watt power supplies, both of which were/are marginal. The major symptom of inadequacy is "wavering" of the display when a drive is accessed.

figured it out. I used the word processor to create an array of text 5 across and 7 down. Then I set the printer for condensed (17cpi) and printed it. When I held the printout up to the light with the array of labels superimposed, it was obvious that the text needed some adjustment. Four iterations later it was perfect, now came the sneaky part. I set the top edge of the paper right at the top of the ribbon guide on the print head. (The actual reference doesn't matter as long as it's repeatable) Then using the "list" command, so that there weren't any strange unforeseen margin or formatting instructions, I printed the pattern copy of the text. After removing the paper from the printer, I used "double stick" tape to fasten an array of blank labels directly over and in "register" with the text. Then I put the paper back in the printer, set the top of the paper right at the top of the ribbon guide on the print head, set the printer for condensed, and "list"ed the file. VIOLA, perfect labels

NZCOM & LAZINESS

When I first installed NZCOM, I maintained named directories absolutely scrupulously. Every time I needed a new area, I'd name it, build a new names.ndr file, load it and use NZBLITZ to build a new system image. That way I could find everything, right? Well yes, it worked, but after the "NEW" wore off, it became a real pain, or I became lazy, the result was the same, I just didn't do it. SO, I devised a new easier system. I just install the new software in a discreet area and use "SALIAS" to create a "batch" file that will take me to the appropriate area, invoke the program and return me to the root directory when I'm finished. By doing this, moving all the "system" files into a different area, and using "SDZ" as an initial command line, only the alias files appear on the screen at boot-up as a menu of sorts. There are more elegant ways to provide menus, but this is just for me, and it's quick and simple, so I'm likely to keep up with it. An example follows:

Letters.com

```
a6:     * this is the area where my word
        * processor is
nw      * invokes NewWord, my word
        * processor
a0:     * takes me back to the root area
sdz     * puts all the alias filenames on the
        * screen again
```

Like I said, not elegant, but it works!!

REGULAR LABELS

While rummaging around the other day, in search of heaven knows what, I came across my original printer, an Epson RX-80 that I paid a king's ransom for 'way back when. It was retired in favor of a Star NX-1000, a year or so ago, dual paper feed, near letter quality, etc. you know the routine, but it occurred to me that it would be ideal for labels. Up to now, I've been removing the paper in the Star and loading tractor feed labels when I needed them, but it was such a nuisance that I avoided it whenever I could. Sound familiar? I now have the Epson on its own shelf, with a box of tractor feed labels on their own shelf beneath it, and an "A-B" switch box between the printers and the computer. I'm still using my Kaypro IV, by the way. The convenience is absolutely wonderful. I use a public domain program called LABELGEN. It was written in basic and compiled, and it's crude but effective. It asks for five lines of input, one at a time, and an "offset" and then prints as many labels as you asked for. The "offset" is really the left margin, and counting characters is up to you. I find myself using it not only for diskette labels, but for return address labels, addresses, box labels, and sometimes just notes I want to stick on something.

Enough of my blunders and techniques, I'd like to hear about some of yours. I can be reached care of TCJ, or at 4000 Norris Avenue, Sacramento, CA 95821.

Advertising Rates For *The Computer Journal*

Size	1 Insertion	2-3 Insertions	4+Insertions
Full	**$400**	**$360**	**$320**
1/2 Page	**$240**	**$215**	**$195**
1/3 Page	**$195**	**$160**	**$145**
1/4 Page	**$160**	**$120**	**$100**
Market Place	**$50**	**$35**	**$35**

Production Schedule

Issue Date	Space Reservation	Copy/Articles 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

MOVING FORTH

by Brad Rodriguez

Special Feature

Intermediate Users

Forth Kernel Design

Part 1: Design Decisions in the Forth Kernel

INTRODUCTION

Everyone in the Forth community talks about how easy it is to port Forth to a new CPU. But like many "easy" and "obvious" tasks, not much is written on how to do it! So, when Bill Kibler suggested this topic for an article, I decided to break with the great oral tradition of Forthwrights, and document the process in black and white.

Over the course of these articles I will develop Forths for the 6809, 8051, and Z80. I'm doing the 6809 to illustrate an easy and conventional Forth model; plus, I've already published a 6809 assembler [ROD91,ROD92], and I'll be needing a 6809 Forth for future *TCJ* projects. I'm doing the 8051 Forth for a University project, but it also illustrates some rather different design decisions. The Z80 Forth is for all the CP/M readers of *TCJ,* and for some friends with TRS-80s gathering dust.

THE ESSENTIAL HARDWARE

You must choose a CPU. I will not delve into the merits of one CPU over another for Forth, since a CPU choice is usually forced upon you by other considerations. Besides, the object of this article is to show how to move Forth to any CPU.

You can expect the usual 16-bit Forth kernel (see below) to occupy about 8K bytes of program space. For a full kernel that can compile Forth definitions, you should allow a minimum of 1K byte of RAM. To use Forth's block-management system for disk storage, you should add 3 Kbytes or more for buffers. For a 32-bit Forth model, double these numbers.

These are the minimums to get a Forth kernel up and running. To run an application on your hardware, you should increase PROM and RAM sizes to suit.

16 OR 32 BIT?

The word size used by Forth is not necessarily the same as that of the CPU. The smallest practical Forth is a 16-bit model; i.e., one which uses 16-bit integers and 16-bit addresses. The Forth community calls this the "cell" size, since "word" refers to a Forth definition.

8-bit CPUs almost invariably support 16-bit Forths. This usually requires explicit coding of double-byte arithmetic, although some 8-bit CPUs do have a few 16-bit operations.

16-bit CPUs commonly run 16-bit Forths, although the same double-precision techniques can be used to write a 32-bit Forth on a 16-bit CPU. At least one 32-bit Forth has been written for the 8086/8088.

32-bit CPUs normally run 32-bit Forths. A smaller Forth model rarely saves code length or processor time. However, I know of at least one 16-bit Forth written for the 68000. This does shrink application code size by a factor of two, since high-level Forth definitions become a string of 16-bit addresses rather than a string of 32-bit addresses. (This will become evident shortly.) Most 68000s, though, have plenty of RAM.

All of the examples described in this article are 16-bit Forths running on 8-bit CPUs.

THE THREADING TECHNIQUE

"Threaded code" is the hallmark of Forth. A Forth "thread" is just a list of addresses of routines to be executed. You can think of this as a list of subroutine calls, with the CALL instructions removed. Over the years many threading variations have been devised, and which one is best depends upon the CPU and the application. To make a decision, you need to understand how they work, and their tradeoffs.

Indirect Threaded Code (ITC)

This is the classical Forth threading technique, used in fig-Forth and F83, and described in most books on Forth. All the other threading schemes are "improvements" on this, so you need to understand ITC to appreciate the others.

Let's look at the definition of a Forth word SQUARE:

```
: SQUARE  DUP * ;
```

In a typical ITC Forth this would appear in memory as shown in Figure 1. (The header will be discussed in a future article; it holds housekeeping information used for compilation, and isn't involved in threading.)

Assume SQUARE is encountered while executing some other Forth word. Forth's Interpreter Pointer (IP) will be pointing to a cell in memory -- contained within that "other" word -- which contains the address of the word SQUARE. (To be precise, that cell contains the address of SQUARE's Code Field.) The interpreter fetches that address, and then uses it to fetch the contents of SQUARE's Code Field. These contents are yet another address -- the address of a machine language subroutine which performs the word SQUARE. In pseudo-code, this is:

```
(IP) -> W     fetch memory pointed by IP into "W" register
              ...W now holds address of the Code Field
IP+2 -> IP    advance IP, just like a program counter
              (assuming 2-byte addresses in the thread)
(W) -> X      fetch memory pointed by W into "X" register
              ...X now holds address of the machine code
JP (X)        jump to the address in the X register
```

This illustrates an important but rarely-elucidated principle: the address of the Forth word just entered is kept in W. CODE words don't need this information, but all other kinds of Forth words do.

If SQUARE were written in machine code, this would be the end of the story: that bit of machine code would be executed, and then jump back to the Forth interpreter -- which, since IP was incremented, is pointing to the next word to be executed. This is why the Forth interpreter is usually called NEXT.

But, SQUARE is a high-level "colon" definition -- it holds a "thread", a list of addresses. In order to perform this definition, the Forth interpreter must be re-started at a new location: the Parameter Field of SQUARE. Of course, the interpreter's old location must be saved, to resume the "other" Forth word once SQUARE is finished. This is just like a subroutine call! The machine language action of SQUARE is simply to push the old IP, set IP to a new location, run the interpreter, and when SQUARE is done pop the IP. (As you can see, the IP is the "program counter" of high-level Forth.) This is called DOCOLON or ENTER in various Forths:

```
PUSH IP       onto the "return address stack"
W+2 -> IP     W still points to the Code Field, so W+2 is
              the address of the Body! (Assuming a 2-byte
              address -- other Forths may be different.)
JUMP to interpreter ("NEXT")
```

This identical code fragment is used by all high-level (i.e., threaded) Forth definitions! That's why a pointer to this code fragment, not the fragment itself, is included in the Forth definition. Over hundreds of definitions, the savings add up! And this is why it's called Indirect threading.

The "return from subroutine" is the word EXIT, which gets compiled when Forth sees ';'. (Some Forths call it ;S instead of EXIT.) EXIT just executes a machine language routine which does the following:

```
POP IP        from the "return address stack"
JUMP to interpreter
```

Walk through a couple of nested Forth definitions, just to assure yourself that this works.

Note the characteristics of ITC: every Forth word has a one-cell Code Field. Colon definitions compile one cell for each word used in the definition. And the Forth interpreter must actually perform a double indirection to get the address of the next machine code to run (first through IP, then through W).

ITC is neither the smallest nor the fastest threading technique. It may be the simplest; although DTC (described next) is really no more complex. So why are so many Forths indirect-threaded? Mainly because previous Forths, used as models, were indirect-threaded. These days, DTC is becoming more popular.

So when should ITC be used? Of the various techniques, ITC produces the cleanest and most elegant definitions -- nothing but addresses. If you're attuned to such considerations, ITC may appeal to you. If your code fiddles around with the insides of definitions, the simplicity and uniformity of the ITC representation may enhance portability. ITC is the classical Forth model, so it may be preferred for education. Finally, on CPUs lacking a subroutine call instruction -- such as the 1802 -- ITC is often more efficient than DTC.

Direct Threaded Code (DTC)

Direct Threaded Code differs from ITC in only one respect: instead of the Code Field containing the address of some machine code, the Code Field contains actual machine code itself.

I'm not saying that the complete code for ENTER is contained in each and every colon definition! In "high-level" Forth words, the Code Field will contain a subroutine call, as shown in Figure 2. Colon definitions, for instance, will contain a call to the ENTER routine.

The NEXT pseudo-code for direct threading is simply:

```
(IP) -> W     fetch memory pointed by IP into "W" register
IP+2 -> IP    advance IP (assuming 2-byte addresses)
JP (W)        jump to the address in the W register
```

This gains speed: the interpreter now performs only a single indirection. On the Z80 this reduces the NEXT routine -- the most-used code fragment in the Forth kernel -- from eleven instructions to seven!

This costs space: every high-level definition in a Z80 Forth (for example) is now one byte longer, since a 2-byte address has been replaced by a 3-byte call. But this is not universally true. A 32-bit 68000 Forth may replace a 4-byte address with a 4-byte BSR instruction, for no net loss. And on the Zilog Super8, which has machine instructions for DTC Forth, the 2-byte address is replaced by a 1-byte ENTER instruction, making a DTC Forth smaller on the Super8!

Of course, DTC CODE definitions are two bytes shorter, since they no longer need a pointer at all!

I used to think that high-level definitions in DTC Forths required the use of a subroutine call in the Code Field. Frank Sergeant's Pygmy Forth [SER90] demonstrates that a simple jump can be used just as easily, and will usually be faster.

Guy Kelly has compiled a superb review of Forth implementations for the IBM PC [KEL92], which I strongly recommend to all Forth kernel writers. Of the 19 Forths he studied, 10 used DTC, 7 used ITC, and 2 used subroutine threading (discussed next). I recommend the use of Direct-Threaded Code over Indirect-Threaded Code for all new Forth kernels.

Jump to NEXT, or code it in-line?

The Forth inner interpreter, NEXT, is a common routine to all CODE definitions. You might keep just one copy of this common routine, and have all CODE words jump to it. (Note that you Jump to NEXT; a subroutine Call is not necessary.)

However, the speed of NEXT is crucial to the speed of the entire Forth system. Also, on many CPUs, the NEXT routine is quite short; often only two or three instructions. So it may be preferable to code NEXT in-line, wherever it is used. This is frequently done by making NEXT an assembler macro.

This is a simple speed vs. space decision: in-line NEXT is always faster, but almost always larger. The total size increase is the number of extra bytes required for in-line expansion, times the number of CODE words in the system. Sometimes there's no tradeoff at all: in a 6809 DTC Forth, an in-line NEXT is shorter than a Jump instruction!

Subroutine Threaded Code (STC)

A high-level Forth definition is nothing but a list of subroutines to be executed. You don't need interpreters to accomplish this; you can get the same effect by simply stringing a list of subroutine calls together:

```
SQUARE:     CALL DUP
            CALL *  ; or a suitable alphanumeric name
            RET
```

See Figure 3. This representation of Forth words has been used as a starting point to explain Forth threading techniques to assembly language programmers [KOG82].

STC is an elegant representation; colon definitions and CODE words are now identical. "Defined words" (VARIABLEs, CONSTANTs, and the like) are handled the same as in DTC -- the Code Field begins with a jump or call to some machine code elsewhere.

The major disadvantage is that subroutine calls are usually larger than simple addresses. On the Z80, for example, the size of colon definitions increases by 50% -- and most of your application is colon definitions! Contrariwise, on a 32-bit 68000 there may be no size increase at all, when 4-byte addresses are replaced with 4-byte BSRs. (But if your code size exceeds 64K, some of those addresses must be replaced with 6-byte JSRs.)

Subroutine threading may be faster than direct threading. You save time by not having an interpreter, but you lose time because every reference to a Forth word involves a push and pop of a return address. In a DTC Forth, only high-level words cause activity on the return stack. On the 6809 or Zilog Super8, DTC is faster than STC.

There is another advantage to STC: it dispenses with the IP register. Some processors -- like the 8051 -- are desperately short of addressing registers. Eliminating the IP can really simplify and speed up the kernel!

The only way to know for sure is to write sample code. This is intimately involved with register selection, discussed in the next section.

STC with in-line expansion; optimization; direct compilation

On older and 8-bit CPUs, almost every Forth primitive involves several machine instructions. But on more powerful CPUs, many Forth primitives are written in a single instruction. For example, on the 32-bit 68000, DROP is simply

```
ADDQ #4,An      where An is Forth's PSP register
```

In a subroutine-threaded Forth, using DROP in a colon definition would result in the sequence

```
BSR ...
BSR DROP  ------>  DROP: ADDQ #4,An
BSR ...  <------         RTS
```

ADDQ is a two-byte instruction. Why write a four-byte subroutine call to a two-byte instruction? No matter how many times DROP is used, there's no savings! The code is smaller and faster if the ADDQ is coded directly into the stream of BSRs. Some Forth compilers do this "in-line expansion" of CODE words [CUR93a].

The disadvantage of in-line expansion is that decompiling back to the original source code becomes very difficult. As long as subroutine calls are used, you still have pointers (the subroutine addresses) to the Forth words comprising the thread. With pointers to the words, you can obtain their names. But once a word is expanded into in-line code, all knowledge of where that code came from is lost.

The advantage of in-line expansion -- aside from speed and size -- is the potential for code optimization. For example, the Forth sequence

```
        3 +
```

would be compiled in 68000 STC as

```
        BSR LIT
        .DW  3
        BSR PLUS
```

but could be expanded in-line as a single machine instruction! Optimizing Forth compilers is too broad a topic for this article. This is an active area of Forth language research; see, for instance, [SCO89] and [CUR93b]. The final culmination of optimized STC is a Forth which compiles to "pure" machine code, just like a C or Fortran compiler.

Token Threaded Code (TTC)

DTC and STC aim to improve the speed of Forth programs, at some cost in memory. Now let's move the other direction from ITC, toward something slower but smaller.

The purpose of a Forth thread is to specify a list of Forth words (subroutines) to be performed. Suppose a 16-bit Forth system only had a maximum of 256 different words. Then each word could be uniquely identified by an 8-bit number. Instead of a list of 16-bit addresses, you would have a list of 8-bit identifiers or "tokens," and the size of the colon definitions would be halved!

A token-threaded Forth keeps a table of addresses of all Forth words, as shown in Figure 4. The token value is then used to index into this table, to find the Forth word corresponding to a given token. This adds one level of indirection to the Forth interpreter, so it is slower than an "address-threaded" Forth.

The principal advantage of token-threaded Forths is small size. TTC is most commonly seen in handheld computers and other severely size-constrained applications. Also, the table of "entry points" into all the Forth words can simplify linkage of separately-compiled modules.

The disadvantage of TTC is speed: TTC makes the slowest Forths. Also, the TTC compiler is slightly more complex. If you need more than 256 Forth words, it's necessary to have some open-ended encoding scheme to mix 8-bit and larger tokens.

I can envision a 32-bit Forth using 16-bit tokens, but how many 32-bit systems are size-constrained?

Segment Threaded Code

Since there are so many 8086 derivatives in the world, segment threading deserves a brief mention. Instead of using "normal" byte addresses within a 64K segment, paragraph addresses are used. (A "paragraph" is 16 bytes in the 8086.) Then, the interpreter can load these addresses into segment registers, instead of into the usual address registers. This allows a 16-bit Forth model to efficiently access the full megabyte of 8086 memory.

The principal disadvantage of segment threading is the 16-byte "granularity" of the memory space. Every Forth word must be aligned to a 16-byte boundary. If Forth words have random lengths, an average of 8 bytes will be wasted per Forth word.

REGISTER ALLOCATION

Next to the threading technique, the usage of the CPU's registers is the most crucial design decision. It's probably the most difficult. The availability of CPU registers can determine what threading technique can be used, and even what the memory map will be!

The Classical Forth Registers

The classical Forth model has five "virtual registers." These are abstract entities which are used in the primitive operations of Forth. NEXT, ENTER, and EXIT were defined earlier in terms of these abstract registers.

Each of these is one cell wide -- i.e., in a 16-bit Forth, these are 16-bit registers. (There are exceptions to this rule, as you will see later.) These may not all be CPU registers. If your CPU doesn't have enough registers, some of these can be kept in memory. I'll describe them in the order of their importance; i.e., the bottom of this list are the best candidates to be stored in memory.

W is the Working register. It is used for many things, including memory reference, so it should be an address register; i.e., you must be able to fetch and store memory using the contents of W as the address. You also need to be able to do arithmetic on W. (In DTC Forths, you must also be able to jump indirect using W.) W is used by the interpreter in every Forth word. In a CPU having only one register, you would use it for W and keep everything else in memory (and the system would be incredibly slow).

IP is the Interpreter Pointer. This is used by every Forth word (through NEXT, ENTER, or EXIT). IP must be an address register. You also need to be able to increment IP. Subroutine threaded Forths don't need this register.

PSP is the Parameter Stack (or "data stack") Pointer, sometimes called simply SP. I prefer PSP because SP is frequently the name of a CPU register, and they shouldn't be confused. Most CODE words use this. PSP must be a stack pointer, or an address register which can be incremented and decremented. It's also a plus if you can do indexed addressing from PSP.

RSP is the Return Stack Pointer, sometimes called simply RP. This is used by colon definitions in ITC and DTC Forths, and by all words in STC Forths. RSP must be a stack pointer, or an address register which can be incremented and decremented.

If at all possible, put W, IP, PSP, and RSP in registers. The virtual registers that follow can be kept in memory, but there is usually a speed advantage to keeping them in CPU registers.

X is a working register, not considered one of the "classical" Forth registers, even though the classical ITC Forths need it for the second indirection. In ITC you must be able to jump indirect using X. X may also be used by a few CODE words to do arithmetic and such. This is particularly important on processors that cannot use memory as an operand. For example, ADD on a Z80 might be (in pseudo-code

```
POP W   POP X   X+W -> W   PUSH W
```

Sometimes another working register, Y, is also defined.

UP is the User Pointer, holding the base address of the task's user area. UP is usually added to an offset, and used by high-level Forth code, so it can be just stored somewhere. But if the CPU can do indexed addressing from the UP register, CODE words can more easily and quickly access user variables. If you have a surplus of address registers, use one for UP. Single-task Forths don't need UP.

X -- if needed -- is more important to keep in register than UP. UP is the easiest of the Forth virtual registers to move into memory.

Use of the Hardware Stack

Most CPUs have a stack pointer as part of their hardware, used by interrupts and subroutine calls. How does this map into the Forth registers? Should it be the PSP or the RSP?

The short answer is, it depends. It is said that the PSP is used more than the RSP in ITC and DTC Forths. If your CPU has few address registers, and PUSH and POP are faster than explicit reference, use the hardware stack as the Parameter Stack.

On the other hand, if your CPU is rich in addressing modes -- and allows indexed addressing -- there's a plus in having the PSP as a general-purpose address register. In this case, use the hardware stack as the Return Stack.

Sometimes you do neither! The TMS320C25's hardware stack is only eight cells deep -- all but useless for Forth. So its hardware stack is used only for interrupts, and both PSP and RSP are general-purpose address registers. (ANS Forth specifies a minimum of 32 cells of Parameter Stack and 24 cells of Return Stack; I prefer 64 cells of each.)

You will occasionally encounter the dogma that the hardware stack "must be" the Parameter Stack, or "must be" the Return Stack. Instead, code some sample Forth primitives, such as

```
SWAP  OVER  @  !  +  0=
```

and see which approach is smaller or faster. (DUP and DROP, by the way, are no test -- they're usually trivial.)

Occasionally you reach strange conclusions! Gary Bergstrom has pointed out that a 6809 DTC Forth can be made a few cycles faster by using the 6809 user stack pointer as the IP; NEXT becomes a POP. He uses an index register for one of Forth's stacks.

Top-Of-Stack in Register

Forth's performance can be improved considerably by keeping the top element of the Parameter Stack in a register! Many Forth words (such as 0=) then don't need to use the stack. Other words still do the same number of pushes and pops, only in a different place in the code. Only a few Forth words (DROP and 2DROP) become more complicated, since you can no longer simply adjust the stack pointer -- you have to update the TOS register as well.

There are a few rules when writing CODE words:

> A word which removes items from the stack must pop the "new" TOS into its register.
>
> A word which adds items to the stack must push the "old" TOS onto the stack (unless, of course, it's consumed by the word).

If you have at least six cell-size CPU registers, I recommend keeping the TOS in a register. I consider TOS more important than UP to have in register, but less important than W, IP, PSP, and RSP. (TOS in register performs many of the functions of the X register.) It's useful if this register can perform memory addressing. PDP-11s, Z8s, and 68000s are good candidates.

Nine of the 19 IBM PC Forths studied by Guy Kelly [KEL92] keep TOS in register.

I think this innovation has been resisted because of the false beliefs that a) it adds instructions, and b) the top stack element must be accessible as memory. It turns out that even such words as PICK, ROLL, and DEPTH are trivially modified for TOS-in-register.

What about buffering two stack elements in registers? When you keep the top of stack in a register, the total number of operations performed remains essentially the same. A push remains a push, regardless of whether it is before or after the operation you're performing. On the other hand, buffering two stack elements in registers adds a large number of instructions -- a push becomes a push followed by a move. Only dedicated Forth processors like the RTX2000 and fantastically clever optimizing compilers can benefit from buffering two stack elements in registers.

Some examples

Figure 5 has the register assignments made by Forths for a number of different CPUs. Try to deduce the design decisions of the authors from this list.

Narrow Registers

Notice anything odd in the figure 5 list? The 6502 Forth -- a 16-bit model -- uses 8-bit stack pointers!

It is possible to make PSP, RSP, and UP smaller than the cell size of the Forth. This is because the stacks and user area are both relatively small areas of memory. Each stack may be as small as 64 cells in length, and the user area rarely exceeds 128 cells. You simply need to ensure that either a) these data areas are confined to a small area of memory, so a short address can be used, or b) the high address bits are provided in some other way, e.g., a memory page select.

In the 6502, the hardware stack is confined to page one of RAM (addresses 01xxh) by the design of the CPU. The 8-bit stack pointer can be used for the Return Stack. The Parameter Stack is kept in page zero of RAM, which can be indirectly accessed by the 8-bit index register X. (Question for the advanced student: why use the 6502's X, and not Y? Hint: look at the addressing modes available.)

In the 8051, you can use the 8-bit registers R0 and R1 to address external RAM, provided that you explicitly output the high 8 bits of address to port 2. This allows a "page select" for two stacks.

UP is different from PSP and RSP: it simply provides a base address; it is never incremented or decremented. So it's practical to supply only the high bits of this virtual register. The low bits must then be provided by whatever indexed addressing technique is used. For example, on the 6809, you can use the DP register to hold the high 8 bits of UP, and then use Direct Page addressing to access any of the 256 locations in this page. This forces all user areas to begin on an address xx00h, which is no great hardship, and limits the user area to 128 cells in length.

On the 8086 you could conceivably use a segment register to specify the base address of the user area.

PART II: BENCHMARKS

By now it must seem that the answer to every design question is "code it and see." .... *and at this point we leave Brad till next issue when he continues our exploration into the insides of different Forth kernels. Brad uses code examples of kernel implementations on 6809, Z80, 8086, and 8051.*

In subsequent articles, Brad will look at:
- design tradeoffs in the Forth header and dictionary search.
- the logic of CONSTANTS, VARIABLEs, and other data structures.

FIGURE 5. REGISTER ASSIGNMENTS

	W	IP	PSP	RSP	UP	TOS	
8086[1]	BX	SI	SP	BP	memory	memory	[LAX84]
8086[2]	AX	SI	SP	BP	none	BX	[SER90]
68000	A5	A4	A3	A7=SP	A6	memory	[CUR86]
PDP-11	R2	R4	R5	R6=SP	R3	memory	[JAM80]
6809	X	Y	U	S	memory	memory	[TAL80]
6502	Zpage	Zpage	X	SP	Zpage	memory	[KUN81]
Z80	DE	BC	SP	IX	none	memory	[LOE81]
Z8	RR6	RR12	RR14	SP	RR10	RR8	[MPE92]
8051	R0,1	R2,3	R4,5	R6,7	fixed	memory	[PAY90]

[1] F83. [2] Pygmy Forth.

"SP" refers to the hardware stack pointer. "Zpage" refers to values kept in the 6502's memory page zero, which are almost as useful as -- sometimes more useful than -- values kept in registers; e.g., they can be used for memory addressing. "Fixed" means that Payne's 8051 Forth has a single, immovable user area, and UP is a hard-coded constant.

- The defining word mechanisms, CREATE.....;CODE and CREATE.....DOES>.
- The assembler vs. metacompiler question.
- The assembler and high-level code that comprises a Forth kernel.
- multitasking modifications to the kernel.

So stay tuned for the next exciting episode of "MOVING FORTH". (editor)

REFERENCES

[CUR93a] Curley, Charles, "Optimization Considerations," Forth Dimensions XIV:5 (Jan/Feb 1993). Description of a 68000 subroutine-threaded Forth.

[CUR93b] Curley, Charles, "Optimizing in a BSR/JSR Threaded Forth," awaiting publication in Forth Dimensions. Single-pass code optimization for FastForth, in only five screens of code! Includes listing.

[KEL92] Kelly, Guy M., "Forth Systems Comparisons," Forth Dimensions XIII:6 (Mar/Apr 1992). Also published in the 1991 FORML Conference Proceedings. Both available from the Forth Interest Group, P.O. Box 2154, Oakland, CA 94621. Illustrates design tradeoffs of many 8086 Forths with code fragments and benchmarks -- highly recommended!

[KOG82] Kogge, Peter M., "An Architectural Trail to Threaded-Code Systems," IEEE Computer, vol. 15 no. 3 (Mar 1982). Remains the definitive description of various threading techniques.

[MOT83] Motorola Inc., 8-Bit Microprocessor and Peripheral Data, Motorola data book (1983).

[ROD91] Rodriguez, B.J., "B.Y.O. Assembler," Part 1, The Computer Journal #52 (Sep/Oct 1991). General principles of writing Forth assemblers.

[ROD92] Rodriguez, B.J., "B.Y.O. Assembler," Part 2, The Computer Journal #54 (Jan/Feb 1992). A 6809 assembler in Forth.

[SCO89] Scott, Andrew, "An Extensible Optimizer for Compiling Forth," 1989 FORML Conference Proceedings, Forth Interest Group, P.O. Box 2154, Oakland, CA 94621. Good description of a 68000 optimizer; no code provided.

[SIG92] Signetics Inc., 80C51-Based 8-Bit Microcontrollers, Signetics data book (1992).

Forth Implementations

[CUR86] Curley, Charles, real-Forth for the 68000, privately distributed (1986).

[JAM80] James, John S., fig-Forth for the PDP-11, Forth Interest Group (1980).

[KUN81] Kuntze, Robert E., MVP-Forth for the Apple II, Mountain View Press (1981).

[LAX84] Laxen, H. and Perry, M., F83 for the IBM PC, version 2.1.0 (1984). Distributed by the authors, available from the Forth Interest Group or GEnie.

[LOE81] Loeliger, R. G., Threaded Interpretive Languages, BYTE Publications (1981), ISBN 0-07-038360-X. May be the only book ever written on the subject of creating a Forth-like kernel (the example used is the Z80). Worth it if you can find a copy.

[MPE92] MicroProcessor Engineering Ltd., MPE Z8/Super8 PowerForth Target, MPE Ltd., 133 Hill Lane, Shirley, Southampton, S01 5AF, U.K. (June 1992). A commercial product.

[PAY90] Payne, William H., Embedded Controller FORTH for the 8051 Family, Academic Press (1990), ISBN 0-12-547570-5. This is a complete "kit" for a 8051 Forth, including a metacompiler for the IBM PC. Hardcopy only; files can be downloaded from GEnie. Not for the novice!

[SER90] Sergeant, Frank, Pygmy Forth for the IBM PC, version 1.3 (1990). Distributed by the author, available from the Forth Interest Group. Version 1.4 is now available on GEnie, and worth the extra effort to obtain.

[SEY89] Seywerd, H., Elehew, W. R., and Caven, P., LOVE-83Forth for the IBM PC, version 1.20 (1989). A shareware Forth using a five-segment model. Contact Seywerd Associates, 265 Scarboro Cres., Scarborough, Ontario M1M 2J7 Canada.

[TAL80] Talbot, R. J., fig-Forth for the 6809, Forth Interest Group (1980).

AUTHOR'S BIOGRAPHY

After twelve years of designing and programming embedded systems, Brad Rodriguez decided he didn't know everything, and went back to school. He is now working full time toward a Ph.D. in Computer Engineering, focusing on real-time applications of artificial intelligence. He still does a little work "on the side" as T-Recursive Technology, and can be contacted as bradford@maccs.dcss.mcmaster.ca on the Internet, or more promptly as B.RODRIGUEZ2 on GEnie. The telecommunicationally disadvantaged can write to him at Box 77, McMaster University, 1280 Main St. West, Hamilton, Ontario L8S 1C0 Canada.

FIGURE 1. INDIRECT THREADED CODE

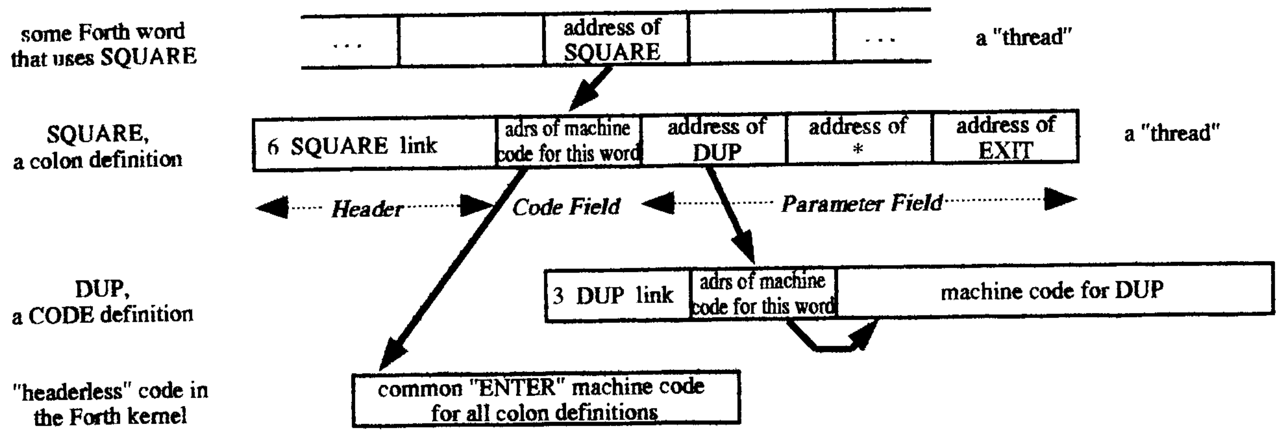


FIGURE 2. DIRECT THREADED CODE

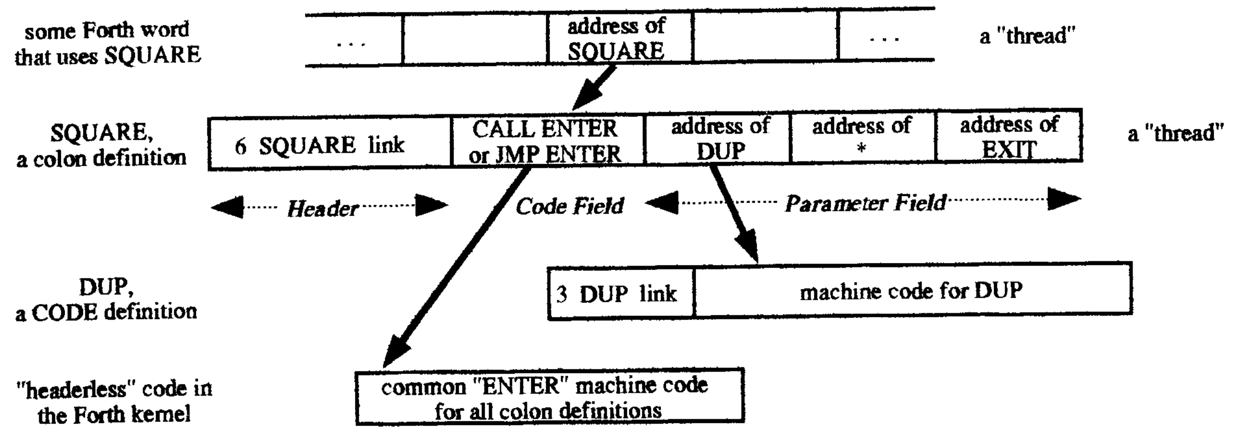


FIGURE 3. SUBROUTINE THREADED CODE

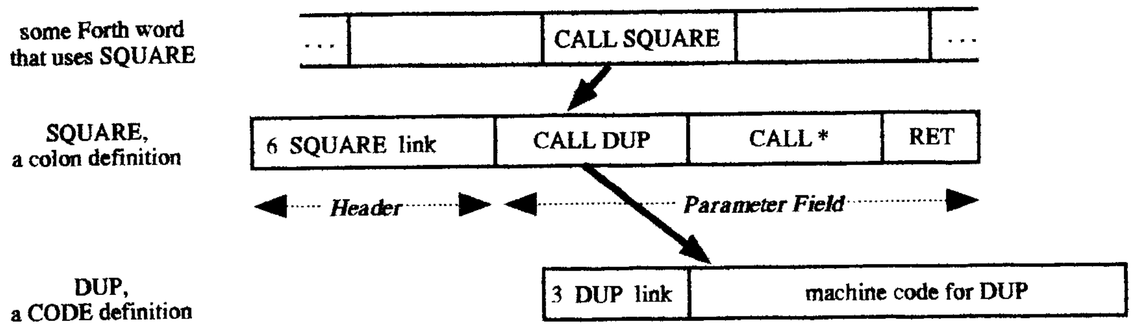
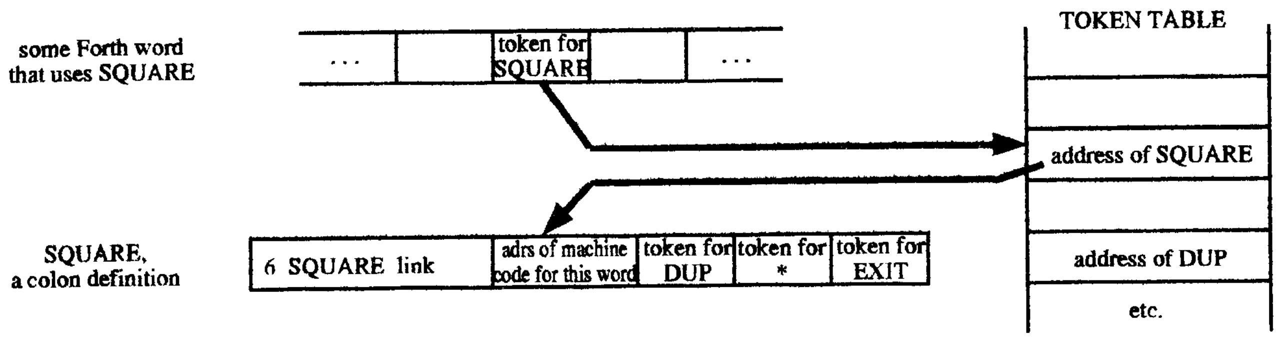


FIGURE 4. TOKEN THREADED CODE



The Computer Journal

Micro Cornucopia Kaypro Disks

K-22
ZCPR

-22-DISK	DOC	10k	EX14	COM	3k	ZCPR10S	HEX	6k
10GINSTL	SUB	1k	EX14	DOC	6k	ZCPR2	HEX	6k
10INSTAL	SUB	1k	GINSTALL	SUB	1k	ZCPR2S	HEX	6k
2INSTALL	SUB	1k	GXINSTAL	SUB	1k	ZCPR4	HEX	6k
4INSTALL	SUB	1k	ZCPR	ASM	53k	ZCPR4S	HEX	6k
CRC	COM	3k	ZCPR	DOC	5k	ZCPRGX	HEX	6k
CRC	DOC	1k	ZCPR	MAN	45k	ZCPRGXS	HEX	6k
CRCKLIST	CRC	1k	ZCPR10	HEX	6k			

Contains ZCPR for all portable Kaypros (II, 4, 10, and 84). This version of ZCPR differs from the version on disk K-9 in a couple of ways. Mainly it fixes bugs in the way the old version handled control characters. The other improvement is that you may use a semi-colon as well as a colon to log onto a drive. In other words, A; has the same effect as A:.

K-23
FAST TERMINAL & RCPM UTILITIES

-23-DISK	DOC	3k	BYE	COM	3k	INTTERM	COM	4k
10GINSTL	SUB	1k	BYEZCPR	DOC	9k	INTTERM	MAC	18k
10INSTAL	SUB	1k	CRC	COM	3k	XLATETBL	ASM	4k
10ZCPR	HEX	5k	CRC	DOC	1k	XMODEM	ASM	43k
2INSTALL	SUB	1k	CRCKLIST	CRC	1k	XMODEM	COM	3k
2ZCPR	HEX	5k	EX14	COM	3k	XMODEM	DOC	3k
484ZCPR	HEX	5k	EX14	DOC	6k			
4INSTALL	SUB	1k	GINSTALL	SUB	1k			
BYE	ASM	63k	INTTERM	$$$	0k			

BYE answers and hangs up the phone. It allows password access to specified drives and user areas.
XMODEM allows the remote system to send and receive files.
ZCPR (and Related Files) contain a protected version of ZCPR that prevents remote users from doing nasty things to your system (like erase all your files).
INTTERM is a replacement for TERM.COM that lets your Kaypro act as a terminal for another system at high baud rates without dropping characters while scrolling.

K-24
KEYBOARD TRANSLATOR & MBASIC GAMES

-24DISK	DOC	3k	KSTR484G	COM	4k	SLOTS	BAS	3k
BACARRAT	BAS	4k	KSTRO484	COM	4k	SPACEWAR	BAS	10k
CASTLE	BAS	26k	KSTROK10	COM	4k	SWORDS	BAS	12k
CASTLE	DOC	15k	KSTROKE2	COM	4k	UN2	COM	5k
CRC	COM	3k	KSTROKES	ASM	20k	UN2	DOC	1k
CRC	DOC	1k	KSTROKES	DOC	17k	USOPEN	BAS	14k
CRCKLIST	CRC	2k	MATHDICE	BAS	2k	XREF&UN2	DOC	3k
DSPACE	BAS	8k	NIM	BAS	4k	XREFPRN2	COM	3k
DUCK	BAS	7k	RUSROU	BAS	1k			
HURKLE	BAS	2k	SKUNK	BAS	8k			

KSTROKES Bill Forbes did an excellent job creating this keyboard translator similar to Smartkey. You can define 8 keystrokes up to 63 characters each.
MBASIC GAMES This disk also contains 13 MBASIC games.
USOPEN illustrates the fairway on the screen. Fore!
DUCK An offshoot of aliens (pardon the pun). Hunter tries to shoot down ducks while ducks try to bomb the hunter. Much fairer than real life.
CASTLE An adventure game.
UN2 allows you to UNprotect MBASIC (version 5.21) files.
XREF produces a cross reference by line number for the output of ASM and LASM files.

K-25
Z80 MACRO ASSEMBLER

25-DISK	DOC•	3k	README	ZQ0•	8k	Z80D	ZQ0•	14k
AZM-COM	DOC•	4k	UNSQ	COM•	12k	Z80E	ZQ0•	10k
CRC	COM•	3k	Z80	COM•	8k	Z80F	ZQ0•	9k
CRC	DOC•	1k	Z80	ZQ0•	5k	Z80MR	COM•	14k
CRCKLIST	CRC•	1k	Z80A	ZQ0•	6k	Z80MR	DOC•	26k
MAC-AZM	DOC•	5k	Z80B	ZQ0•	20k			
PHASE	DOC•	9k	Z80C1	ZQ0•	12k			
PHASE1	AZM•	2k	Z80C2	ZQ0•	17k			

This disk contains a Z80 macro assembler that is the nicest I've seen in the public domain.
Z80MR.COM The Z80 macro assembler.
Z80MR.DOC is our documentation file crammed with examples and as much information as I could come up with on this assembler.
AZM-COM.DOC How to get from an assembly language source file to a COM file.
PHASE.DOC How to get around the phase and dephase operators found in M80 files.
MAC-AZM.DOC What you need to know to change .MAC (MicroSoft's M80 assembler source files) to .AZM files.
PHASE1.AZM Sample program described in PHASE.DOC
Z80.COM The Crowe assembler extensivly modified to do conditionals, as well as a number of other new tricks.

K-26
EPROM PROGRAMMER & TOOLS

26-DISK	DOC•	6k	DDTTOMAC	COM	10k	I/O-CAP	DOC	8k
CHARK	DAT	2k	DDTTOMAC	DOC	2k	KDEBUG	AZM	8k
CHARS	DAT	2k	EPROM	AZM	23k	KDEBUG	HEX	3k
CHREDIT	COM	16k	EPROM	COM	4k	LOOK	ASM	7k
CHREDIT	DOC	3k	EPROM	DOC	1k	LOOK	COM	1k
CHRLIST	COM	14k	EPROM	TXT	9k	PROGTEST	AZM	6k
COMPARE	COM•	2k	GETMON	AZM	2k	PROGTEST	COM	1k
CRC	COM•	3k	GETMON2K	COM	1k	TELL	ASM	6k
CRC	DOC•	1k	GETMON4K	COM	1k	TELL	COM	2k
CRCKLIST	CRC•	2k	I/O-CAP	ASM	19k	UNLOAD	COM	1k
DDTTOMAC	C	7k	I/O-CAP	COM	1k			

EPROM runs the EPROM programmer published in *Micro Cornucopia* Issue #18.
GETMON These programs get the code in your monitor ROM and load it at 100h so that you can use the SAVE command to save the file for later disassembly or debugging.
I/O-CAP captures console output and/or input to a disk file.
DDTTOMAC works on the output of DDT.COM when using the "l" command to disassemble a .COM file. The output of DDT is captured using the program above. Then DDTTOMAC reworks the file so that it can be reassembled.
KDEBUG One of the biggest drawbacks of the Kaypro is its lack of RAM resident monitor. This is a kludgey way of providing one, but anything is better than nothing. With this, "break points" can be included in the program under test to point to this monitor, and then memory can be examined or modified.
COMPARE The best binary file compare I could find in the public domain.
LOOK looks for a 1-9 byte sequence in memory.
TELL shows you the CP/M landmarks.
UNLOAD creates a .HEX file from a .COM file.
CHREDIT A character set editor for 2716 character generator EPROMs. With this you can design your own character set and then burn it with your new EPROM programmer.
CHRLIST.COM dumps your character set to the printer.

The Computer Journal

Micro Cornucopia Kaypro Disks

K-27
TYPING TUTORIAL

```
27DISK    DOC•  5k    TDRILL   BAS•  5k    TTYPEXA  DAT•  1k
CRC       COM•  3k    TDRILL   COM• 14k    TTYPEXB  DAT•  2k
CRCKLIST  CRC•  2k    TTHELP0  DAT•  1k    TTYPEXC  DAT•  3k
CRT       DEF•  1k    TTHELP1  DAT•  2k    TTYPEXD  DAT•  2k
EATYPE    BAS• 23k    TTHELP2  DAT•  1k    TTYPEXE  DAT•  2k
EATYPE    COM• 23k    TTHELP3  DAT•  1k    TTYPEXF  DAT•  2k
ENVELOPE  BAS•  4k    TTHELP4  DAT•  1k    TTYPEXG  DAT•  2k
ENVELOPE  COM•  7k    TTHELP5  DAT•  1k    TTYPEXH  DAT•  3k
ENVELOPE  DOC•  1k    TTKEYBD  DAT•  1k    TTYPEXI  DAT•  3k
FOGINDEX  COM• 11k    TTYPE    BAS• 20k    TTYPEXJ  DAT•  3k
FOGINDEX  DOC•  2k    TTYPE    DOC• 19k
```

TTYPE.DOC Documentation file for EATYPE typing tutorial. Read this file first!
EATYPE EATYPE.BAS and EATYPE.COM are modified versions of the TTYPE program. They use the same data files and instructions but are much easier to list and install.
TTYPE TTYPE.BAS may drive your printer nuts because of the carriage returns without line feeds. Most of these have been removed in EATYPE.
TDRILL More typing drills.
CRT TTYPE has to be set up for your CRT and for the keyboard STATUS and INPUT ports.
ENVELOPE allows you to use your newly acquired tying skills to type addresses on envelopes. Happy Hunting and Pecking!
FOGINDEX calculates the Gunning-Mueller Clear Writing Institute Fog Index of an ASCII file based on sentence and word length.

K-28
MODEM 730 SOURCE

```
28-DISK   DOC   4k    M7FNK    COM   3k    MDM730   DOC  39k
CRC       COM   3k    M7FNK    DOC   2k    MDM730   MSG   4k
CRCKLIST  CRC   1k    M7FNK    NOT   3k    MDM730   NOT   3k
KM1200    COM  19k    M7LIB    COM   2k    MKP4-10  ASM  18k
KM300     COM  19k    M7LIB    DOC   3k    MLOAD21  COM   3k
KPOV      ASM  18k    M7NM-6   ASM   6k    PIP      COM   8k
KPOV      HEX   2k    M7RUB    MSG   2k
M7BELL    MSG   1k    MDM730   COM  19k
```

MDM730.COM is the main program developed by Irv Hoff and is one of the most versatile modem programs available bar none. This file has not been set up for Kaypro and is included for archival purposes only.
KM1200.COM A ready to run version of MDM730.COM set up for the Kaypros. As presently set up, it has a phone directory, autodial, redial, touchtone, 1200 baud, 8 bit, 1 stop bit, and no parity check.
KM300.COM Same as KM1200.COM except defaults to 300 baud rate.
MDM730.DOC A very comprehensive explanation of the features of MDM730 and KM1200.
MKP4-10.ASM Overlay file containing assembly instructions to set up MDM730.COM to operate properly on Kaypro II,4 & 10.
M7NM-6 Overlay file containing assembly instructions to set up MDM730.COM to include the autodial telephone directory.
M7FNK Program and documentation to allow you to change the 10 function keys in MDM730.COM or KM1200.COM to whatever you like.
M7LIB Program and documentation to allow you to change the phone numbers in the MDM730.COM or KM1200.COM autodial telephone directory.
MLOAD21 A public domain version of CP/M's LOAD.

K-29
TURBO PASCAL GAMES I

```
29-DISK   DOC•  5k    CHUCKLCK PAS•  4k    HANGMAN  CHN•  2k
BACCARAT  CHN•  3k    CRC      COM•  3k    HANGMAN  PAS•  6k
BACCARAT  PAS•  7k    CRCKLIST CRC•  2k    HORSERAC CHN•  3k
BJGAME    CHN•  5k    D        COM•  3k    HORSERAC PAS•  5k
BJGAME    PAS• 10k    DODGE    CHN•  6k    KENO     CHN•  2k
BLACKBOX  CHN•  4k    DODGE    PAS• 12k    KENO     PAS•  5k
BLACKBOX  PAS•  8k    GAMEMENU CHN•  1k    MAKEFILE CHN•  1k
BLOCK     CHN•  3k    GAMEMENU PAS•  2k    MAKEFILE PAS•  2k
BLOCK     PAS• 10k    GAMES    COM•  8k    READNUM  PAS•  1k
BOGGLE    CHN•  2k    GAMES    PAS•  1k    WORDS    DAT•  2k
BOGGLE    PAS•  4k    GUESSIT  CHN•  1k
CHUCKLCK  CHN•  2k    GUESSIT  PAS•  3k
```

K-30
TURBO PASCAL GAMES II

```
30-DISK   DOC•  5k    KISMET   PAS• 12k    READNUM  PAS•  1k
ARTILLRY  CHN•  2k    LANDER   CHN•  4k    SNAKE    CHN•  4k
ARTILLRY  PAS•  4k    LANDER   PAS• 10k    SNAKE    PAS•  8k
CRC       COM•  3k    LANDINST DAT•  2k    STARS    CHN•  4k
CRCKLIST  CRC•  2k    NUMCNVRT CHN•  2k    STARS    PAS• 13k
D         COM   3k    NUMCNVRT PAS•  5k    TELEPHON CHN•  2k
GAMEMENU  CHN•  1k    OVERUNDR CHN•  2k    TELEPHON PAS•  5k
GAMEMENU  PAS•  2k    OVERUNDR PAS•  4k    TWINKLE  CHN•  1k
GAMES     CHN•  1k    PASCAL   CHN•  1k    TWINKLE  PAS•  2k
GAMES     COM•  8k    PASCAL   PAS•  3k    WUMPUS   CHN•  4k
GAMES     PAS•  1k    PLIFE    CHN•  4k    WUMPUS   PAS•  9k
KISMET    CHN•  6k    PLIFE    PAS•  9k
```

More chained games with source in Turbo Pascal.

K-31
TURBO BULLETIN BOARD

```
-31-DISK  DOC   2k    D        COM   3k    TBBS     PAS  11k
10GINSTL  SUB   1k    EX       COM   3k    TBBSCOM  INC   8k
10INSTAL  SUB   1k    GINSTALL SUB   1k    TBBSHDR  INC   4k
2INSTALL  SUB   1k    GXINSTAL SUB   1k    TBBSMSG  INC  10k
31-DISK   DOC   1k    RZCPR10S HEX   5k    TUTL     COM  27k
4INSTALL  SUB   1k    RZCPR2S  HEX   5k    TUTL     PAS  13k
BYE       COM   3k    RZCPR84S HEX   5k    USQ      COM   2k
BYE       DOC   1k    RZCPRGXS HEX   5k    XMODEM   COM   3k
BYEZCPR   DOC   9k    SYSMSG   BB#  11k    XMODEM   DOC   3k
CRC       COM   3k    TBBS     COM  27k    ZCPRBLOC COM   1k
CRCKLIST  CRC   2k    TBBS     DOC  14k
```

TBBS is the bulletin board program. Turbo Pascal source code included.
TUTL maintains the bulletin board files (messages, users, and the log).
BYE To set up your system for remote use, type BYE. When someone calls, BYE answers the phone and runs TBBS.
XMODEM A program for sending or receiving files with TBBS.

K-32
FORTH-83

```
32-DISK   DOC   2k    D        COM   3k    META80   BQK  72k
CPU8080   BQK   9k    EXTEND80 BQK   9k    USQ      COM   2k
CRC       COM   3k    F83      COM  25k    UTILITY  BQK  36k
CRCKLIST  CRC   1k    F83      DOC  10k
CUSTOM    BQK   4k    KERNEL   COM  12k
```

Here we go—FORTH Heaven again! **F83** is the Laxen and Perry FORTH-83 with some extensions.

K-33
UTILITIES

```
33-DISK   DOC•  2k    NSWP2    HLP• 11k    NULUTERM ASM•  3k
CRC       COM•  3k    NSWP2    WS • 28k    SUPERZAP COM•  6k
CRCKLIST  CRC•  1k    NSWP207  COM• 12k    SUPERZAP DOC• 11k
D         COM   3k    NSWP207  DOC•  3k    VDO-KP   COM•  5k
EDIT      COM•  2k    NULU     DOC• 40k    VDO23-KP DOC• 13k
EDIT      DOC• 13k    NULU11   COM• 15k
HELP      COM•  2k    NULU11   NOT•  2k
```

NSWP207 The latest version of SWEEP, written by Dave Rand. NSWP allows you to tag files for copying, erasing, squeezing, unsqueezing, finding, and printing.
NULU11 Creates, manipulates, and extracts libraries.
SUPERZAP A full screen disk editor at only 6K.
VDO-KP A fast, mini-editor—small and easy to use.
EDIT A utility to copy and edit files (text and binary).

K-34
GAMES

```
34-DISK   DOC•  7k    DBLICK-V PQS•  8k    GERMS    COM  12k
BOARD     MQC   7k    DOCTOR   ELZ•  5k    GERMS    PQS   6k
CRC       COM•  3k    EL1      PQS•  6k    OTHELLO  COM  17k
CRCKLIST  CRC•  1k    EL2      PQS•  3k    OTHELLO  DOC   1k
CRIBBAGE  COM• 20k    ELIZA    COM• 13k    OTHELLO  FQR   7k
CRIBBAGE  PQ2•  3k    ELIZA    DOC•  8k    PIP      COM•  8k
CRIBBAGE  PQ3•  6k    ELIZA    PQS•  3k    USQ      COM• 10k
CRIBBAGE  PQS• 11k    FAKECPM  ELZ•  2k
DBLICK-V  COM• 17k    FAKEVAX  ELZ•  2k
```

Disk 34 contains five games along with source code. Turbo Pascal is the language of choice with the exception of OTHELLO, which is written in FORTRAN. OTHELLO will run only on 84 model Kaypros.
CRIBBAGE is, as you may have guessed, a computer version of the card game cribbage.
DBLICK-V Nicely done version of the venerable video game BREAKOUT.
ELIZA allows conversation with your computer.
GERMS is variation on LIFE.
OTHELLO takes advantage of the video capabilities of the 84 model Kaypros.

K-35
SMALL C (Ver 2.1) COMPILER & SOURCE

```
35-DISK   DOC•  1k    CC21     C  •  6k    CCCM     SUB•  1k
ARGS      C  •  1k    CC22     C  •  9k    CCCR     SUB•  1k
CAT       C  •  1k    CC3      C  •  2k    CCM      SUB•  1k
CC        COM• 29k    CC31     C  •  8k    CCRTL    MAC• 25k
CC        DEF•  8k    CC32     C  •  6k    CCRTL    REL•  3k
CC        DOC• 10k    CC33     C  •  5k    CRC      COM•  3k
CC1       C  •  6k    CC4      C  •  1k    CRCKLIST CRC•  2k
CC11      C  •  7k    CC41     C  •  8k    D        COM•  3k
CC12      C  •  8k    CC42     C  •  9k    UW       C  •  2k
CC13      C  •  8k    CCC      SUB•  1k    WC       C  •  2k
CC2       C  •  2k    CCCC     SUB•  1k
```

Disk 35 is Fred Scacchitti's upgrade of the Small C compiler (Version 2.1) and includes source.
This version requires Microsoft's M80/L80.

K-36
SMALL C LIBRARY

```
36-DISK   DOC•  1k    D        COM   3k    SCLIB2   LBR• 78k
CRC       COM•  3k    DELBR    COM• 13k
CRCKLIST  CRC•  1k    SCLIB1   LBR• 39k
```

Disk 36 contains a library of 105 C functions. Most of these are described in Jim Hendrix's *Small C Handbook.*

K-37
UTILITIES PRIMER

```
37-DISK   DOC•  2k    DEBUG    COM•  4k    VFILER   COM•  8k
CRC       COM•  3k    DEBUG    Z80• 52k    VFILER   DOC•  3k
CRCKLIST  CRC•  1k    DU-V86   ASM• 56k    VFILERSC ASM•  3k
DBUGABST  DOC• 28k    DU-V86   COM•  8k
DBUGINST  DOC• 10k    DU-V86   DOC• 13k
```

Disk 37 contains three utilities. We've included assembler source code for DEBUG and DU-V86.
DEBUG In Issue #25 of *Micro Cornucopia,* Richard Amyx wrote an article titled "Why I Wrote a Debugger." DEBUG is a beta test copy of his Z80 debugger discussed in the article.
DU-V86 The latest version of the disk utility DU allows you to view and edit any byte on a disk.
VFILER Similar to SWEEP, VFILER performs the same basic file manipulations, but adds screen oriented displays.

K-38
PASCAL RUNOFF WINNERS FIRST—THIRD

```
38-DISK   DOC•  3k    PLANTER1 IQC   7k    PROBE3   PQS  13k
CRC       COM•  3k    PROBE    COM  27k    RESCUE   DOC•  8k
CRCKLIST  CRC•  1k    PROBE    DQC  23k    RESCUE   PQS•  9k
PLANSTRU  DOC   5k    PROBE    PQS   1k    RESCUE60 COM• 14k
PLANTER   COM  17k    PROBE1   PQS  13k    RESCUE63 COM• 14k
PLANTER   DOC   4k    PROBE2   DQC   6k    USQ      COM•  2k
PLANTER   PQS   9k    PROBE2   PQS   7k
```

PROBE Rick Ryall's effort was the unanimous winner. His disk editor will view and edit sectors, find bad sectors, copy blocks to a new location, search for text strings, and feed the dog.
RESCUE Steve Mitton's second place entry provides a painless search of memory for text lost due to whatever made your Kaypro crash.
PLANTER Third place goes to Dennis Sprague. After the user describes the dimensions and number of sides of a wooden planter box, PLANTER draws (on 84 Kaypros) the required pieces and gives all dimensions necessary to build the box.

K-39
PASCAL RUNOFF WINNERS FOURTH–FIFTH

```
39-DISK   DOC•  2k    PAMPHLET PQS•  9k    VB-BLK2  IQC•  4k
BUILDING  FRE•  2k    SAMPLE   IQ •  7k    VB-BLK3  IQC• 10k
CHAPTER4  FRE•  2k    SAMPLE   OQT•  7k    VB30     DQC• 24k
CHAPTER5  FRE•  2k    TEST     FRE•  1k    VB30I    COM• 23k
CLASROOM  FRE•  1k    USQ      COM•  2k    VB30I    DTA•  5k
CRC       COM•  3k    VB       000• 10k    VB30I    MSG•  3k
CRCKLIST  CRC•  1k    VB       COM• 17k    VBDIRCPM IQC•  3k
HELP      VB •  7k    VB       PQS•  2k    VBDIRDOS IQC•  3k
PAMPHLET  COM• 11k    VB-BLK1  IQC•  6k
```

VB Written by Norman Saunders and Frances Coniglio, VB (Vocabulary Builder) teaches foreign language vocabulary.
PAMPHLET Steve Wilcox's fifth place finisher takes a Wordstar file and rearranges the pages in the proper order for printing a folded pamphlet.

K-40
PASCAL RUNOFF WINNERS SIXTH PLACE

```
40-DISK   DOC•  2k    BOTTIBLD STR•  2k    CRCKLIST CRC•  1k
BOTTI     COM• 25k    BOTTIDEC INC•  8k    IOERROR  INC•  2k
BOTTI     ERR•  1k    BOTTIFRE IQC•  6k    PRESIDNT BOT• 15k
BOTTI     INS• 24k    BOTTIINI IQC•  5k    PRESIDNT DSC•  1k
BOTTI     PAS•  6k    BOTTIINP INC•  8k    PRESIDNT FRE•  2k
BOTTI     UNK•  1k    BOTTIOUT INC•  3k    PRESIDNT IFR•  1k
BOTTIBLD  COM• 16k    BOTTIPLY IQC•  5k    PRESIDNT IPL• 10k
BOTTIBLD  DQC• 14k    BOTTIPTR INC•  3k    README   DOC•  2k
BOTTIBLD  DSC•  5k    BOTTISTR INC•  3k    USQ      COM•  2k
BOTTIBLD  PAS• 11k    CRC      COM•  3k
```

Sixth place in the Pascal Runoff goes to Ernest Adams for **BOTTICELLI**, a game in which you think of a person's name and the computer tries to guess it. Be fair now, don't use your Aunt Agnes.

K-41
EXPRESS 1.01 TEXT EDITOR

```
41-DISK   DOC•  1k    ECONFIG1 COM• 12k    TERM     DAT• 14k
CRC       COM•  3k    EXPRESS  OVL•  5k    USQ      COM•  2k
CRCKLIST  CRC•  1k    EXPRESS1 DQC• 57k
D         COM   3k    ROFF4    COM• 31k
E         COM• 19k    ROFF4    DOC• 34k
```

EXPRESS This is the public domain version of the Stump Brothers' full screen text editor.
ROFF is the classic text formatter.

U.S. Canada/Mexico Europe/Other
Software Disks (CA tax) add these shipping costs for groups of 3 disks ordered
MicroC Disks are $6.00ea +$1.00 +$1.00 +$1.25 +$1.50 +$2.50

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

The Computer Journal

Back Issues

Sales limited to supplies in stock.

Issues 1 to 19 are currently OUT of print. To assist those who want a full collection of TCJ issues we are preparing photo-copied sets. The sets will be issue 1 to 9 and 10 to 19. Each set will be bound with a plastic protective cover.

The price for bound volumes is $20 plus shipping . Expect TWO to THREE weeks for delivery after payment received at TCJ. Some single copies available, contact TCJ before ordering.

Issue Number 20:
· Designing an 8035 SBC
· Using Apple Graphics from CP/M: Turbo Pascal Controls Apple Graphics
· Soldering & Other Strange Tales
· Build an S-100 Floppy Disk Controller: WD2797 Controller for CP/M 68K

Issue Number 21:
· Extending Turbo Pascal: Customize with Procedures & Functions
· Unsoldering: The Arcane Art
· Analog Data Acquisition & Control: Connecting Your Computer to the Real World
· Programming the 8035 SBC

Issue Number 22:
· NEW-DOS: Write Your Own Operating System
·Variability in the BDS C Standard Library
· The SCSI Interface: Introductory Column
· Using Turbo Pascal ISAM Files
· The Ampro Little Board Column

Issue Number 23:
· C Column: Flow Control & Program Structure
· The Z Column: Getting Started with Directories & User Areas
· The SCSI Interface: Introduction to SCSI
· NEW-DOS: The Console Command Processor
· Editing the CP/M Operating System
· INDEXER: Turbo Pascal Program to Create an Index
· The Ampro Little Board Column

Issue Number 24:
· Selecting & Building a System
· The SCSI Interface: SCSI Command Protocol
· Introduction to Assemble Code for CP/M
· The C Column: Software Text Filters
· Ampro 186 Column: Installing MS-DOS Software
· The Z-Column
· NEW-DOS: The CCP Internal Commands
· 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.

The Computer Journal Back Issues

- 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
- 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

#25
ISSUE NOW AVAILABLE!

	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 groups of 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. 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

Regular Feature

Editorial Comment

Parting Words

Computer Corner

By Bill Kibler

Well it seems that after all the other work in putting *TCJ* together, I am also suppose to write my regular column. Strange as it may seem I in fact even have information for it as well.

GIVE ME A BREAK....

The break I am looking for in this column is not related to doing *TCJ*, but to some mail I received the other day. Seems this group gave out a support award to MicroSoft. The group is "Software Support Professional Association" and the award was for "High Call Volume". Seems MicroSoft handles 35,000 (yup thousand) calls every DAY!

Now give me a break here, you can get an award for having software so bad that 35 thousand people a day have to call you. Hasn't Microsoft ever hear of making manuals so people can use them instead of calling. Actually having used many of their manuals, the problem most likely is the fact that the software doesn't work like the book said it would. The flip side is that the user proably found another bug and is calling to find out what they MicroSoft intends to do about it (or not do).

Personally if I had a product that required that much support every day, it wouldn't take me long to figure I had more problems than I was solving. Certainly makes you think there are a lot of unhappy users out there, that given half a change to use something else would jump at it. This fact also reinforces my belief that if you really want to learn about software and hardware, don't do it on a PC, unless you want to be one of the 35,000 callers.

LANs

My data communications course is over till next fall and that gives me a chance to check out new information. My students were evenly split between needing MODEM and LAN technology. The MODEM part went very well, althought many students found their understanding of the hardware and software relationships involved a bit lacking. That was very clear when we tackled the LAN information.

It seems that LAN operations are still a mysterious box and work by some sort of black magic. My course did better than most, because I laid it out with t he idea that all fundamentals would be covered using MODEMs as the instructional medium. Lets face the facts, LANs are but an outgrowth of MODEM technology. LANs have control software that talks to device drivers that talk to medium which carries the signals to the opposite end. The layers of software and hardware are really not that different between LANs and modems, yet most students are lost once you say LAN.

After the course work I started checking out cheap LAN systems again, especially for the older CP/M systems. I did talk to the people at $25 network to see if their serial LAN software would be portable to CP/M. The answer was a "most likely NOT." That means we will still need to contact other vendors for crossover products.

I understand that Artisoft had an early product using Z80 controllers for the LAN interface cards. It might be possilbe that their earlier software could be had and ported to our machines. Maybe this is going the wrong direction as well and an older lower technology approach is needed.

XEROX X.25

On one of the older disks in SIG/M is an X.25 software driver for the Xerox 820 computers. I have it around here and have tried it. This is on SIG/M 238 and you can get it from Elliam Assoc. or your regular SIG/M BBS or supplier.

I am not sure about all the ins and outs of the program. I do remember that all the Z80 source code was provided. This might make a good source for some networking software. In fact any machine with a Z80 SIO chip can be usd for HDLC communications (which is the basis for X.25). These all same XEROX 820's are the hardware behind many of the HAM radio packet systems. I tried to make mine do all the HAM stuff but was unable to get my 7910 single chip modem device to work properly. I think too many things came up and therefore I didn't get back to find out what went wrong. I believe the ARRL Handbook now has the schematic and guidelines for doing it yourself.

One of the reason for interest in this area is a friends desire to do a low cost printer network. Actually all his needs are is being able to hook up two computers to one lazer printer. A main desire is not to run cable through walls. I keep thinking about a cheap radio transmitter and reciever combination ala packet radio. Hopefully my next project will be along those lines.

Speaking of lines, looks like I have run out for now. Till later, keep HACKING!
Bill Kibler

Discover
The Z-Letter

The Z-letter is the only monthly publication for CP/M and 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.
Personality Decoder Boards
Run more than two drives when using TurboROM, $25.
Hard Drive Conversion Kits. Call or write for availability & pricing.

Call (916)483-0312
eves, weekends or write
Chuck Stafford
4000 Norris Ave.
Scaramento, CA 95824

***TCJ* MARKET PLACE**
Advertising for small business
First Insertion: $50
Reinsertion: $35

Rates include typesetting. Payment must accompany order. VISA, MasterCard, Discover, Diner's Club, Carte Blanche, JCB, EuroCard 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

8 BITS *and Change*

CLOSING OUT SALE!

All 12 Back Issues

for only $40

Send check to

Lee Bradley
24 East Cedar Street
Newington, CT 06111
(203) 666-3139 voice
(203) 665-1100 modem

S-100/IEEE-696

Compupro
Cromemco
IMSAI
and more!

Cards • Docs • Systems

Dr. S-100

Herb Johnson,
CN 5256 #105,
Princeton, NJ 08543
(609) 588-5316

SUPPORT OUR ADVERTISERS TELL THEM "I SAW IT IN TCJ"

Z80 STD USERS!

Cost Effective Upgrade
Clock Speeds to 10 MHz
1 Mbyte On-board Memory

Increase your system performance and reliability while reducing your costs by replacing three of the existing cards in your system with one **Superintegrated** Z80 Card from Zwick Systems.

A **Superintegrated** Card in your system protects your software investment, requiring only minor changes to your mature Z80 code. You can increase your processing performance by up to 300 percent in a matter of days!

Approximatly 35 percent of each **Superintegrated** Card has been reserved for custom I/O functions including A/D, D/A, Industrial I/O, Parallel Ports, Serial Ports, Fax and Data Modems or almost any other form of I/O that you are currently using.

Call or Fax today for complete information on this exciting new line of **Superintegrated** Cards and upgrade your system the easy way!

ZWICK SYSTEMS INC.
Tel (613) 726-1377, Fax (613) 726-1902