Operating Systems

Z-System Corner

Quality Control

Real Computing

Kaypro 5 MHZ

Center Fold

Support Groups for the Classics

Multiprocessing for the Impoverished

The Computer Corner
68XXX COMPUTER PRODUCTS From Peripheral Technology
68000 System Boards with 4 Serial/ 2 Parallel Ports, FDC, and RTC.
PT68K4-16 with 1MB $299.00
PT68K2-10 w/ 1MB (Used) $149.00
REX Operating System Included
OS9 V2.4 Operating System $299.00
With C, Editor, Assembler/Linker SCULPTOR V1.14:6 for Business Software Development - requires any version of OS9/68K. $79.00
Other 68XXX products available!

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

Cross-Assemblers as low as $50.00
Simulators as low as $100.00
Cross-Disassemblers as low 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 fully 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
  - Motorola 6800
  - Hitachi 6301
  - Rockwell 6502
  - Hitachi HD64180
  - 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

SAGE MICROSYSTEMS EAST
Selling and Supporting the Best in 8-Bit Software
Z3PLUS or NZCOM (now only $20 each)
ZSDOS/ZDDOS date stamping BDOS ($30)
ZCPR34 source code ($15)
BackGrounder-II ($20)
ZMATE text editor ($20)
BDS C for Z-system (only $30)
DSD: Dynamic Screen Debugger ($50)
4DOS “zsystem” for MSDOS ($65)
ZMAC macro-assembler ($45 with printed manual)

Kaypro DSDD and MSDOS 360K FORMATS ONLY
Order by phone, mail, or modem and use Check, VISA, or MasterCard.

Sage Microsystems East
1435 Centre Street
Newton Centre MA 02159-2469
(617) 965-3552 (voice) 7PM to 11PM
(617) 965-7259 (paw=DDT)
(MABOS on PC-Pursuit)
Editor's Comments ................................................ 2

Reader to Reader .................................................. 3

Support Groups for the Classics ............................... 8
A feature to help you find support.
By JW Weaver.

Z-Systems Corner .................................................. 10
Automating replies to GENie Mail.
By Jay Sage.

Real Computing ................................................... 15
JPEG, WORM, archivers, and grippers.
By Rick Rodman.

Operating Systems ............................................... 18
Exploring the beginning of CP/M.
By Bill Kibler.

Center Fold .......................................................... 21
The XEROX 820.

Quality Control Using the Commodore 64 .............. 29
BASIC I/O operations using a Classic computer.
By Ralph Tenny.

Mr. Kaypro .......................................................... 32
Making the 5 MHZ upgrade.
By Charles B. Stafford.

Multiprocessing for the Improvised ......................... 36
Part I, the 6809 Uniprocessor.
By Brad Rodriguez.

The Computer Corner ............................................. 47
IDE is actually "AT A".
By Bill Kibler.
Welcome to issue #61! It is jam packed with all the great features you have come to love. In fact our Reader To Reader section is full of love letters and messages that are a must read for everyone. Check out Tom Hoot's messages about a CP/M CD ROM.

From there we turn to support for users with JW Weaver's Support Groups column. JW covers a few newsletters we get at TCJ. For support of ZMATE macros, Jay Sage is back with more on using macros to automate downloading of messages from GENie.

The CP/M introduction article is here, and I review some basic concepts and history. We will have more detailed articles later, after I get a regular writer for the new column. So far Rick Rodman has indicated his desire to carry on with operating systems and CP/M internals help.

Rick hasn't forgotten his real computing section and this time talks about JPEG, WORM, and gripes. Well not too many gripes, but then anyone using PCDOS these days has plenty to complain about.

Chuck Stafford fills us in on the next step of his 5MZH speed up for Kaypros. He leads those first time hackers through the steps needed to make the changes without damaging your boards and pride.

Our S-100 support person, Herb Johnson was out of the country and returned only to move into a new home. Needless to say he was too busy with more important chores and unable to do his article. He promises to catch up next issue. So if you need S-100 help drop him a note and he will fill you in next time.

Our feature articles start out with Brad Rodriguez's hardware portion of the 6809 Multiprocessing series. This is part one of the hardware side and he gives us the schematics and some testing software. His part three of the software or Moving Forth series is in my in basket and will be in the next issue, #62. Brad has indicated that part two of the hardware will be in issue #63. So stay tuned for this "getting it going" series.

Our next feature is a classic study of using a Commodore 64 to collect test data in an automatic testing situation. Ralph Tenny sent this in for all those beginning hackers. It shows that BASIC still has a place in those simple first time projects.

The center fold section starts a three part series on the Xerox 820. These units were sold by the hundreds at swap meets and are very similar, well actually a variation of the Big Boards and Kaypro boards. Most users have trouble finding the jumper and connector information so I am including it with the schematics as well.

I cover some important to me topics in my Computer Corner. Whether or not you find them important is up to you. To find out what I mean, you will just have to read it.

RENEWAL NOTICES

Currently the TCJ mailing database is not working as well as it needs to. Should you have problems or meet a friend who says they have not gotten their issues, please have them contact our office. I keep a paper trail to back up the computer data, so I can usually find out what has happened and send you any missing issues.

The billing is currently being done every other issue and till I can change the database it will have to stay that way. This leaves it up to you to remember to renew your subscriptions. John Hall of FIG says most FIGgers don't renew until they get the third renewal notice. Shame on you FIGgers, pay up quicker.

Your label has the expiration issue number and you will get at least one renewal notice for now. Remember we make it simple to renew, just call the 1-800 number and give us your credit card number. My answering machine is on all the time so there is no excuse for not renewing early. Please be aware that our credit card agency no longer accepts Discover and American Express. The only credit cards we can accept now are Master Card, VISA, Carte Blanche, and Diner's Club.

TCJ WILL?

Since dealing with 68 Micro Journal, I have had reason to review TCJ's situation. 68 Micro Journal ceased operation when the editor died. Now I don't plan on dying soon, but just to be safe and sure that TCJ will continue should something happen to me, I am seeking names for the TCJ Will.

What I am looking for is a few reliable people to start handling some minor aspects of TCJ production. Rick Rodman has indicated his interest in picking up support (possibly doing the mailing list) and learning the publishing business. Are there two others willing to learn (on the west coast?) and do some work?

Then those that show they could handle TCJ, should the need arise, will be put in the TCJ Will. That means should something happen, hopefully one of the possible people would be able to pick up the pieces and carry the tradition forward without major interruption. Not only will I sleep better, but as readers it will nice knowing your support will not disappear one day, just like 68 Micro Journal.

Happy reading, Bill Kibler.
Dear Mr. Bill,

Well, the 60th issue of TCJ arrived and a jampacked issue it is.

I like the centerfold idea. Though the IMSAI documentation that came with my machine is fairly complete, two boards I inherited were orphans. Luckily, your first centerfold provided just what I needed for the original IMSAI MPU. However, I could use one for the 5 1/4" North Star floppy disk controller, which definitely falls into the weird stuff category.

I enjoy Brad Rodriguez's articles and look forward to the next one. I've picked up theoretical knowledge on hardware, but have been reluctant to start chips smoking. Like Art Carlson, I'm intrigued by embedded systems, but believe the pure standalone systems too tedious to work with. Much better would be a system you can either program or dump code to over a serial line (preferably Forth). Brad's project should also answer the question of how you jump start a new system. Would it pay to incorporate the STD bus? I've heard that it is widely used in controllers, but have not seen any information published on it.

I read Earl Bryant's lament, but if all he remembers of Chris McEwen's reply is that "TCJ was never meant to be light reading", he missed the point. Chris went on to implore him to write to TCJ and ask questions. Instead he waited for a KEY to strike him down on the road to Damascus and reveal all. Asking questions is like a race. Not every question may be answered, and not everyone wins in a race, but if you don't enter you will never win. So what is the price of knowledge? It starts at 29 cents.

Glad to see Charles Stafford's articles on breathing new life into old Kaypros. The upgrades are definitely worth the doing, especially to the II's. I've noticed, however, that it is more difficult to get new drives for the old systems. The double density drives are being phased out in favor of the high density ones. I haven't even seen ads for the 'Quad' drives (96 tpi 5" DSDD) in a year.

I have TEAC drives and they work just fine (though a bit noisy), but they require extra work to install. The signal edge connector on the TEAC is mounted upside down compared to the original drives and just about every other drive. This reverses the signals. You have to mount ribbon connectors on the back side of the ribbon cable and then twist the cable 180 degrees to what it was originally. Toshiba or Fujitsu drives would plug in without cable modification.

Moving the brightness control to the front is also worthwhile. You have to mount it toward the top of the case so the knob clears the keyboard when you close the case. Mine is 1 3/8" down from the top to the centerline of the knob.

I've mounted the speed switch on the front panel too, but find I rarely use it. The only time I need to change to low speed is in formatting and accessing single density disks for the antique Shugart disk drive on the IMSAI. The high clock speed would crash the system then.

The DataMaster 23 is an IBM business computer popular from the mid 70's to early 80's. I inherited it after it blew out the memory board and IBM wanted $1,100 to repair it. After nearly ten years of solid use, the owners decided it was time to upgrade to an IBM-AT. I was interested as much in the voluminous documentation as the hardware.

It is an all-in-one desktop unit the size of a large terminal, with two 8" DSDD drives, tactile keyboard, a 10" green monochrome display, and NO reset button. Processor has an 8-bit data bus and addresses 64K RAM, with option to add an additional bank of 64K RAM. The RAM uses a ninth bit for a run time parity check. A large number of ROM chips hold a solid interpreter BASIC, much of the DOS, and routines for extensive startup error checking. Disks have volume labels.

Does any of this sound familiar? Yep, the DataMaster 23 is the father of the IBMPC. However, in true Blue tradition it uses the EBCDIC character code rather than ASCII.

I wonder if anyone is confused by my saying that debugging should start with the top word? This refers to the notion that a program is shaped like a triangle with a Word at the top (or apex) that starts the program running. The rest of the code expands as a hierarchy of routines that ends with a large number of primitive routines at the base. In the Screen to Text file utility (TCJ#60,p.35), the top Word is S-T. You would start here and if a Word in the parameter list crashes, switch the DEBUG to it, and so on until you find the bug. Unfortunately, when you print out source code, the triangle gets inverted so that the top Word ends up at or near the bottom of the listing.
Yours truly, Walter J. Rottenkolber

Thanks Walter for those comments and your many articles. #60 seems to have been a bit hit, and the centerfold is going great. I think Charles has more on the Kaypro disk problems and Brad is moving along with his series. Earl has responded with some clarification of his first letter which I print next.

Like many readers you seem to want more on embedded systems. I am trying, but how about using that old S-100 system instead. After all you know the system well by now and what's more you know it works as a regular computer so if it doesn't work as an embedded controller it must be something you changed. I am interested in anything using the STD bus, but so far I have only seen a few cards on the used market, and never a bus plane (too expensive to buy new for me).

Thanks for the info on the DataMaster, guess I now know what IBM's second failure was before the PC. Wonder who knows what their first attempt at PC's was. I only know that the PC was their third attempt and it should have failed as well. What more can I say...Thanks.

Bill.

Dear Mr. Kibler:

Thank You for responding to my recent letter. Please consider this one to be an attempt to be really helpful. You do, indeed, have a tough row to hoe and I have no desire to make it tougher. I, PERSONALLY, stand to benefit very greatly if you can succeed in your stated goals.

The entire computer industry suffers from a lack of genuine information skills. There may be no way to make any improvement. If there IS a chance, it probably is limited to the small segment you have chosen to address.

It seems to be a fact -- a pretty awful fact-- that computer whizzes are almost without exception unable to understand the problem they have in communicating. It may be that, in some cases, there is an actual desire to be secretive--to exclude all those "other guys" from the fraternity. It may be that some who have absorbed the secrets pretty well think that all of the "outsiders" are incapable of understanding and should, therefore, be abandoned. Whatever the rationale, the fact remains, the written material concerning computer subjects is usually pretty bad!

I AM an outsider. I am not stupid. I could teach most of the authors of computer literature that I have been exposed to at least as much as they could, if they could only communicate, teach me.

I received my M.S. in Physics in 1951--a time when it was not at all possible to obtain PhD's in basket weaving. I worked for 40 years as an electronics engineer. I worked with vacuum tube digital logic systems. I designed very large transistorized data processing machinery before integrated circuits were ever thought of. My specific tasks actually isolated me from computers during their early stages of development.

I finally acquired a 64K CP/M machine after retirement. The instructional material accompanying the machine was out-and-out gibberish and just plain WRONG! Do the things exactly as stated in the books and NOTHING would happen! I spent 6 months of cut-and-try and was able, at last, to make it work.

I have made repairs and added soft- and hard-ware. I can do a little programming in Basic. Those efforts were based entirely on instinct and a willingness to experiment a lot. I would LIKE to do more but have yet to find any reasonable source of information.

I would like to get into other programming languages. I would like to add a hard disk. I would like to know why my double-sided 8-inch drive (an unsupported homegrown addition) simply duplicates data on both sides. I would like to try upgrading my clock rate a bit--might make it possible to put the 608K on each side of my 8-inch disks rather than the 241K I'm now allowed. I'd LIKE to make use of my IEEE-488 port but what, exactly, is it good for?

If folks interested in non-IBM compatibles really hope to keep their stuff alive they should have one guiding principle: MAKE IT UNDERSTANDBLE TO THE DUMMIES THAT JUST DON'T SEEM TO GET IT! You need new blood desperately and you ain't-a-gonna-get-it without meeting the "dummies" at least half way! TCJ's stated objectives SOUND ideal--it's in the execution that the goals get lost. Here are a few suggestions:

1. Publish no article using computerese without including a glossary defining every term which MIGHT be troublesome to ANYONE. One glossary central to each issue would work but letting authors do the job on each article would be preferable. This latter approach would keep the necessity for clarity right out there in front of each author. They seem all too ready to dispense with clarity for the masses in favor of impressing the other guys they know that have been working on the same stuff right along.

2. If your spelling needs improvement, use the spelling checker that's part of your word processor. Use the checker anyway--it will help track down all of the gibberish. No fair putting your mnemonics and acronyms into the speller's dictionary.

3. If your command of grammar is weak, find a buddy to proof and edit for you. School teachers will help if you volunteer to help THEM with computer problems.

4. Try a few of your essays on someone you KNOW to be no expert in the subject. The written language is really more powerful than the greatest computer ever built but just as capable of obscuring meanings if misused. You will NOT identify indistinct meanings by running them past your KNOWLEDGEABLE buddy that has been in on your effort since day one.

5. Establish a "department" in your publication directed precisely at the computer "illiterate". Think about it. Macintosh and Microsoft have sold millions of folks on computers by simply making
the machines "USER FRIENDLY"!
MS-DOS is no more than a derivative of CP/M. CP/M COULD be much more popular if its values were not consistently hidden behind insider communications skills deficiencies. Why SPE-
CIALIZE in USER UNFRIENDLI-
NESS?

If you can get even a few of your writers to step back and judge their written efforts in terms of real understandability you'll grant them the very great pleasure of much wider recognition. Your maga-
azine might attain some of your stated goals. You might even rescue some of the classic computerdom from an early death due to widespread disinterest.

I sincerely wish for your success in this tough job.

Yours very truly, Earl Bryant.

It is hard to know how to answer your letter, Earl. Your five steps could not be more on target. I sent one article back with those type of suggestions, only to find it published in another magazine (with my suggestions incorporated in it). Not paying for articles has always hurt TCJ and like everything around here it takes a very long time to change things. I have started making changes, but at best only one item gets corrected each issues. You indicated, TCJ has many items needing work, so expect the changes to take many issues before it becomes apparent they have changed.

I think my harping and your letters are starting to get through to the authors. Brad's 6809 article got help for people just like yourself ("Memory Decoding for Programmers"). In fact you are indeed the typical user and reader of TCJ. Many of my authors have thought otherwise, but your background actually fits closer to the typical than any other. Your highly qualified and skilled in many ar-
eas, but simply lack the hands-on background experiences that are required to understand computerese.

It always amazes me that Microsoft has been so successful when their own manu-
als in fact meet your description of things not working as stated. I have never meet

anyone who has found any manual on operating systems very useful. Maybe our new feature on operating systems will help, I hope so. At present I spell check everything I get at least two times or more. Amazingly I still get the magazine back from the printer and usually find at least one or two errors within 5 minutes.

I too started with vacuum tube digital systems, and yet find digital fundamentals are the same. What the problems is, the fundamentals, often don't get reviewed when discussing a new topic. I am trying to change things, just don't get upset if it seems like it is taking a long time, good change takes time.

Thanks again, Bill Kibler.

Dear Sir:

Enclosed is my check for $24 for a renewal subscription to TCJ.

Not having a bill in my little tickler file allows me to forget, but it also motivates me to put in some sort of a letter of transmittal with my check. This is good because then I add a few comments, which you often request. So here goes.

I look forward to each issue because each one is new adventure in computing that I can relate to. As you say, MS DOS machines are reduced to installing and removing cards. They do wonderful things and I have a couple of them.

As a way to stimulate interest in 8-bit systems, I suggest articles like Tom McDonough's Embedded Systems for the Tenderfoot in issues 45-47. Also I seem to remember that Bar Coding might be a possible subject for future articles. I know that there are programs out now that bar code mail. I have heard that there is a possibility that individual mail may get a reduced rate sometime in the future. I think it would be fun to get into that.

On the centerfold diagrams, I have a pretty complete set of diagrams and manuals for the old big board computer that I think was manufactured by Digital

Research Computers. I would be pleased to share what I have with anyone who is interested. I seem to recall an article in Micro Cormocopia to the effect that the Big Board was basis for the Kaypro, the Xerox 820, and others.

Even if I don't get around to doing many of these projects, as a retired electrical engineer, I enjoy reading about them and thinking them out.

Sincerely yours, Eliot C. Payson, Littleton, CO.

Well Eliot, you caught me at one of my major problems, billing of renewals. Our database is not what is needed and I haven't had time to correct it properly. Currently I only bill every other time so you only get one chance, the rest is up to you. Check out the centerfold, it is the Xerox 820 and you are right the Big Board was the basis for for it and others. I am doing the Xerox because the used boards were being sold by the hundreds. I have been trying to get more embed-
ded articles, but all my writers are working around the clock, and can't find time to write about their work. Maybe later we can catch up and get embedded projects back into TCJ.

Thanks, Bill Kibler.

Dear Bill,

Here's my subscription renewal, I've been a subscriber for about five years now, and although the magazine has undergone some subtle changes over that time there has always been something in it to interest me. My collection of com-
puters is pretty eclectic (a good way to say "no taste"): a 386SX PC clone to do work-work on, a Morrow MD-11 that used to have that task, a couple of VIC-20's, a ZX81 and an IMSAI S100 box that I acquired in "almost working" condition (and hasn't improved much!). The IMSAI was the reason I subscribed in the first place, as I wanted to learn how I could upgrade it.

I am an electronic engineer by trade, dealing mostly now with analog/RF hardware and interfacing. (I can program in assembly, but I've sworn to only do it in
self defense). Old machines are fun and a good way to learn how real computers work. I am working on a VIC-to-S100 card interface to help test out the motley assortment of S100 cards I've gotten, and I'll proceed from there. I've also helped out setting up an elementary school classroom with computers, that would make a good article, but I'm not sure if TCJ would be the place for it.

What would I like to see in the magazine? I don't know. I think the technical level of TCJ is about right, it's not for beginners but it's not Dr.Dobbs Journal, either. Multipart articles, or all-one-topic issues are O.K. but they should not be the rule. The rule over the past few years has been towards a mix of things, some multi parts, and some semiregular columns, and that's fine with me.

Wishing you good luck! Ken.

Well Ken, I wish I had written down your last name when I separated your money from you in more than one way. It is amazing but I have had several conversations with ZX-81 users. Those little early machines are still being used by some of our readers. Maybe if I can find a schematic of them, I will publish it in our center fold. Got a schematic?

S-100 is very good for cutting ones teeth on. I like them because of the many boards and parts that force you to learn about different topics and areas represented by each board's functionality.

How about a letter to our S-100 column with you problems listed and maybe we can help you get started in the right direction. Dr.Dobbs we will never be, especially since we have not become a C only magazine. Many readers dropped them when their Forth special issue only had ONE article on Forth. What kind of special issue is that when you only have a single article on the subject!

Thanks for the good words, it is glad to know we have not lost all our old readers. Thanks again, Bill Kibler.

Dear TCJ:

We use two MS-DOS computers, a 286 and a 486, for home/hobby and home office purposes. I have an AMPRO 1B (CP/M), and a MicroSolutions Z-80 plug-in card, both of which are currently in mothballs, and I sometimes use the MYZ80 emulator on my 486 machine.

As far as presentation of articles is concerned, I prefer to see complete articles in one issue as much as possible. I realize that sometimes it is just not practical to print a long piece of text and, say, accompanying schematic diagrams in one issue, but ordinarily I like to see complete articles.

I would like to see some articles on the subject of new uses and applications for old ('"classic'"") computers. As much as I enjoyed hacking on my old CP/M systems in past years, they really do not fill our needs for speed, file capacity and organization, and other factors in today's business applications. I'd like to see some new ideas for using these wonderful, reliable old friends.

In regards to TCJ generally, I must add my voice to those others who have asked for more careful editing and attention to the quality of writing that you print. Poor grammar, clumsy syntax, and misspelled words have no place in an otherwise high quality magazine. I understand that you can only do so much with the time you have available, but there are such things as spell-checkers, and being a technical person does not obviate the need for good communications skills.

I have always liked TCJ; it is one of the few remaining hands-on publications for those of us who are more than computer appliance users. Please keep up the good work!

Sincerely, Lawrence Sonderling, Northridge, CA.

Thanks for the comments. Lawrence. We have had many articles on the AMPRO you own, but none that I know of on the Z-80 plug in card. I got a Microlog Z80BXT Board at a swap meet last week and am now looking for documentation and software, have any? How about a few words from you on the MicroSolutions? Did it work and if so how well? You are right however in using the MYZ80, I have it too and it works great. That reminds me, I need to send him some money, since it is worth every cent he charges.

I am trying to improve things but until I find a part time job or consulting work, TCJ's hours will remain one or two stuffed between everything else I am doing (like working 12 hour days). Since I have been unable to catch up on producing TCJ on time, it seems that third and fourth review just doesn't get done. I would love to have help checking things over, but far too often I get the articles the week before I take them to the printer. Hopefully things will change soon for the better. Thanks again for your kind words, Bill Kibler.

Greetings!

I am reading your issue #60, my first (I subscribed from an ad in Nuts and Volts), and I am *tremendously* pleased. I've only gotten up to page 8 -- "Next Ten Years Part II" by yourself -- and I felt I had to blather.

* In your excellent letters column, a Preston Bricker had an "Intersystems S-100" system. Joy!! I hadn't seen the name for years. I worked there for a definite while, as bottle-washer and annoying-as-I-could-be self-appointed quality control expert. The story I still include in my resume is how I wrote a memory test program in machine language -- only choice, actually -- which failed every machine in the building!

* C- (what the world calls C++): It's very simple; it's built on the Wizard/Dope model: the Wizards will write all these incredibly wonderful functions, which we will *never have to modify*, and we, simple peasants that we are (i.e. the Dopes), will *use* these Perfect Functions to do all the stuff which, in the past, we were forced to code *ourselves*! Yes, absolutely. ... And if you believe that*....

* I am a real genuine software developer. I am paid to do it -- mostly as a consultant for a manufacturing tools
company you have never heard of called Loveshaw (they make really good machines to whack your cardboard boxes however you like).

* I don’t care about classic computers beyond reason (I gave my last Kaypro away to some nice religious people a few years ago), but I *do* use CP/M, in the form of an emulator card + software, because the Microsoft M80 assembler is *still* much better compared to disappointing cross-asser assembler offerings, i.e. it has more-or-less working macros and so on. Also, I have a *rare* (thankfully) “DynaByte” hard-disk CP/M machine in the basement; ... and the original Radio Shack Color Computer ... and the Timex TI-whatever it’s called an elderly friend of my parents pushed-off on me ... and a Teletype which doesn’t work anymore since I tried to fix the missing line feed problem.....

* One of our projects recently induced me to seek information about the IBM PC bus -- i.e., hardware-wise. I went to many bookstores; Daltons, Waldens, and many others in this region. They all had *endless* offerings about *software* about data-base-this, and lan-that, and guess-what-the-other. I DID NOT FIND ONE BOOK THAT DESCRIBED THE IBM PC/AT/ANYTHING HARDWARE DMA CYCLE -- which was what I was looking for. Actually, I didn’t find any book which described any aspect of the IBM PC standard, ISA, EISA, who-knows-etc., whatever, hardware anything in *ANY WAY*! The press whines about Japan, Europe, unemployment ... Why is it in America we can read about software until we vomit, but a hardware description isn’t saleable? ... *I* think it’s sick; I’m going to call Rush Limbaugh one of these days...

-- But Very Best Wishes, James Gregor Owen * CIS: 71121.625

Thanks James for the blather...

Which Z80 emulator are you running and how about some more information about it. Is it worth it?? Better than MYZ80? Always wanted an Intersystems, but got CCS and Cromemco stuff instead. Went looking for books myself last weekend and decided that the quality of user support is inversely proportional to the number of books available to support the software products. Based on that idea “C” must be the worst program out, because there were more books on it than any other (well maybe MicroSoft products as a whole were larger). Will be having some 6809 (Color Computer) articles soon so dust off those old machines and get ready for some old time fun. We are also working on more help putting hard disks on CP/M, got any advice from using the Dynabutes?

Thanks for the good words, Bill Kibler

Bill - Just got the issue you mailed and sent a check in return mail.

Would you ask around and see if there is truly any interest in a CD containing CP/M files. I feel this would be the best way to archive what is out there. CD’s do not corrupt like disks sometimes do. Also this would easily allow a DOS system to support the CPM community.

Tom Hoot

Well Bill, it looks like the CP/M CDROM is coming together... what follows is a message I just sent to Jack Velte at Walnut Creek.

Ok guys- lets get those file to him so he can press the CD! They can take tape.

Bob:

It was a pleasure to talk with you today on the phone and you cannot gauge the amount of my enjoyment that Walnut was going to come out with a CDROM on CP/M. In a word FANTASTIC!!!

Robert Bruce Walnut Creek CD-ROM 4041 Pike Ln E, Concord CA 94520 800-786-9007 510-674-0783 fax 510-774-0821 rab@cdrom.com or Jack Velte velte@cdrom.com

Sources for files:
1. wsmr-simtel20.army.mil
   a. pd2:<cpm> - you said you had all this
   b. pd2:<cpmug> - CPM User’s group collection
   c. pd2:<sigm> - Sig M User’s group collection

2. Beehive BBS in Australia (file list Beehive.zip to follow) BEEHIVE.ZIP, Sysop: Alan Sanders 300-2400 baud +61-2-975-4982, FIDO: 3:714/406, INTNet: 58.2100/211.6, INTERNET: alan.sanders@f406.n714.23.fidonet.org {this should work alright}, Postal: P. O. Box 496, Forestville, Sidney, N.S.W., AUSTRALIA 2087
3. Z-Node62 +61-9-450-0200 (file list to follow) Bruce Dudley (sysop), ZNODE62.ZIP
4. Z-Node3 617-965-7259 PASSWORD DDT, Jay Sage of Sage Micro Systems, 1434 Centre St, Newton MA 02159-2469 EVENINGS ONLY, sage@ll.mit.edu
5. CPM Zombies - 705-444-5654
7. Poseidon Electronics - Ralph Lees, 103 Waverly Place, Ny Ny 10011, 212-777-9515
8. * Eliam Associates - Bill Roach (talked w/him - very helpful), Box 2664, Atascadero Ca 93422,805-466-8440
9. * FOG - First Osborne Group {This is a must!} Box 1030, Dixon CA 95602, 916-678-7353, Mike Kaufman - He want’s to be included.
10. GENie - CP/M, Commodore 128 & 64, and others (lists to follow) GENCBM.ARC & GENCPM.ARC
11. Elephant’s Graveyard - 619-270-3148
12. Enterprise BBS - ENTER.ZIP

A good place to advertise - Specializes in ZCPR and CP/M, The Computer Journal, editor is Bill Kibler

The * indicates group newsletter <=== a way to spread the word. There will be some duplication, but these are the best sources I could think of. Waiting eagerly for the release of this unique collection.

--
Tom Hoot thoot@wixer.bga.com
7203 Smokey Hill Rd
Austin Texas 78736
Enterprise BBS 512-453-5079
FIDO 1:382/81
GEenie THOOT

Continued on page 9
SUPPORT GROUPS FOR THE CLASSICS

By JW Weaver

It is with great regret that I have to announce the demise of SKUG. This is a club I wrote about in issue 60. At the April meeting, the club voted to disband. Through the members generosity, I will be keeping alive the BBS, KRASH II.

Of which, a message was posted to me from a Loran Guyz of Salmon, Idaho, asking if this is the BBS to leave information about organizations, and their events. WELL, partner, it sure is, and I welcome any and all information about non-PC or non-MAC events / clubs / libraries. Lorans group supports the Coleco ADAM.

USER GROUPS

ACCESS - ATARI SUPPORT GROUP

Meets the 1st Thursday of each month, 7:30pm. SMUD Training Bldg, Room A 1708 59th St. Sacramento, California. Contact Person: Bob Drews Phone: (916) 423-1573 Mailing Address: PO Box 1354, Sacramento, CA 95812

Another small group, approximately 20 members, supporting the 8 bit systems, and the Atari ST, with 4 or 5 members knowing the inside and outsides of the Atari systems. Maintains an Atari library, and puts out a newsletter. I have not had the opportunity to attend a meeting, but hope to soon.

SACRAMENTO MICROCOMPUTER USER GROUP

Meets the 1st Thursday of each month, 7:30pm. Consumer Education Room, 6201 S St., Sacramento, California. Contact Person: John Bryant Phone: (916) 967-0988 Mailing Address: PO Box 161513, Sacramento, CA 95816-1513 BBS:(916) 372-3646

General direction of the group is to promote interest in and skills with computers. Although the focus is technical in nature, members interest encompass all types of hardware and software. Maintains an extensive library, of CP/M and PCDOS software. Newsletter is "Push & Pop", issued monthly to members, and carries articles on diverse subjects.

NEWSLETTERS

Newsletters under review this issue are: The Z-Letter, Sanyo PC Hackers Newsletter International, The Staunch 8/89'er. All of these newsletters have very interesting articles and letters from readers. They are published in California, Oregon, and Iowa.

PUSH & POP - Newsletter of SMUG

A 10 or 12 page letter, covers trade news, technical tips, MS-DOS 5.0 message reference, High Definition TV notes, Viruses, PKZIP v2.04c, and CD ROM technical summary. Address of SMUG was given above.

SANYO PC HACKERS NEWSLETTER International

14 page newsletter, published monthly, articles cover mainly the Sanyo PC versions of hardware (a non clone PCDOS machine), with associated software, help on problem solving, hacking the metal, and the software. Tips on patching Word Star, word processing in general, and attaching different configurations of printers to your system. A must if you are using the Sanyo 55x box. Contact Person: Victor R. Frank Address: 12450 Skyline Blvd, Woodside, CA 94062-4541 OR Mad Hacker, PO Box 762, Menlo Park, CA 94026. Phone:(415) 851-7031

THE Z-LETTER

About 20 pages per issue, produced bi-monthly, dealing with CP/M and Z-Systems, problem solving, upgrades of hardware. Kaypro, Z-100, Adam, Franklin Ace 1200, Morrows are some of the boxes covered in the Letters section. David is an avid collector of old systems and mainly old boot disks. Should you need original software to get your system going, this is the first place to start looking for help. Contact Person: David A.J. McGlone, Lambda Software Publishing, Address: 149 West Hilliard Lane, Eugene, OR 97404-3057. Phone:(503) 688-3563

THE STAUNCH 8/89'er

A bi-monthly publication of approximately 24 pages, main topic is Heith/ Zenith 89 systems. Catalog of CP/M and HDOS software available, letters from readers giving their solutions to problems, hardware and software. Lists contacts for other publications and suppliers. Contact Person: Kirk L. Thompson Address: PO Box 548, West Branch, IA 52358 Phone:(319) 643-7136

OTHER HAPPENINGS

I have been monitoring the INFO-CPM Digest, which has lots of traffic on CP/M, ADAM, and other beasties. I will try to monitor on a regular basis. Next time I will explain more about this service.

I understand SIMTEL20 contains a large library of CP/M software, and will be
researching this further with hope of writing about it soon.

Again, I request your help with information on your support group(s), upcoming events, suppliers, providers of repairs, etc. Copies of newsletters are a great help.

JW

Write to:
TCJ Support Groups
Drawer 180
Volcano, CA 95689
BBS: (916)427-9038
300/1200/2400 8N1

JW was unable to attend the local Forth Interest Group meeting here in Sacramento when John Hall presented the FIG stopped by. In our discussions of Forth and changes happening with FIG a couple of items surfaced. First off TCJ is not associated with FIG other than swapping ads and running articles about Forth. I have been aFIG member for many years and feel that it is still the primary source for Forth help and information. That is the second item about our meeting. FIG is trying to get the word out that it is still around supporting Forth users and "want to be" users.

The main point is, if you use or would like to learn about Forth the first place to start is FIG. John has started making changes that will help new users find support and publications. They are in the process of getting the rights to republish some out of print books on Forth. They soon will be on internet for users of that service.

All our members came away feeling better about FIG and especially John's renewed promise to support beginners and novice users. Should you feel that a little advice or have some written comments, please send John that letter with your comments as he is looking for more help and input in keeping FIG going another 15 years. Their address and phone number are on the inside cover.

Bill Kibler.

Reader to Reader continued.

This is great news Tom. Thanks for heading up this project. I hope all is still coming together since you sent me these two messages on Genie. I talked to Walnut Creek the other day myself, just to see how they were doing on the CP/M CD ROM. They indicated it should be done by end of June, with maybe as much as 500K of software! They sounded like nice people especially if they are willing to put together a CP/M CD ROM for us. Guess it means I will miss putting all of FIG's source code on it. Maybe the next CD will get our stuff. So please keep us informed as I know our readers are very interested in this development.

Thanks Bill Kibler.

Dear Mr. Kibler:
I am a long time subscriber to The Computer Journal, and as such have become quite attached to the magazine, and delight in receiving each new issue. I also subscribed to Micro-C and 68 Micro Journal, and find your magazine the only remaining "Hobbyist/Enthusiast" magazine in the business.

I was reading (more like "devouring"!) Issue 60 (Happy Tenth Anniversary!), and noted your mention of your interest in supporting Motorola (6800, 6809, 6811, 68K) microprocessor based systems, and this really sparked my interest. As I mentioned above, I used to receive 68 Micro Journal, and have quite a few 6800/6809 systems (even had a SK-DOS 68K system at one time). I have spent very much time with these systems and the FLEX operating system, and am quite interested in writing a few articles for The Computer Journal.

I am employed as a Hardware Design Engineer, and so spend most of my time with hardware issues. However, like all real computer Hobbyists, I have spent many, many night in front of the computer with Software projects. I am fully versed in Assembly Language (most microprocessors), "C", BASIC, and even some PASCAL. I have done a lot of programming on the 8080/8085 and the 6800/6809/68K.

As I mentioned, I have my own FLEX09 computer, and would be reviving it for use during this series of articles (I would be doing my writing on my IBM-PC clone). Also, I will revive my SK-DOS 68K computer for 68000 type projects.

I am particularly interested in answering questions, and giving them pointers in solving their "Embedded" microprocessor project problems. This is my strongest interest - applying Single-Chip "Micro-controller" units (the 6805, 6811, 683xx, 8051). I have extensive experience in embedded system interfacing - Stepper Motor controllers, Light dimmers, sensor interfacing, A/D and D/A circuitry, PLL, Phone Line interfacing, DTMF, Modems, etc.

I will begin writing my first article in the next few days, and will send you a copy of the result. I look forward to your response, and hope to be able to contribute significantly to The Computer Journal.

Sincerely, Carl Terrier, Dallas, TX.

Well Carl you are just what I have been looking for. All our regular embedded writers are too busy to write these days. Sounds like you must have time available or are so happy to be writing about 6800 type machines to take time off. We have just added a new advertiser (Peripheral Technology) which makes and sells 68xxx type systems. You probably had one of their 68K SK-DOS systems. How about writing about them and their products? Did you know they also are putting out a 68020 version of that PC compatible 68K mother board? I also understand you can get the old versions for $149 (been thinking about getting one myself), but the new version really sounds great. How about contacting them and just reviewing what they now have in relation to what you already have used and worked with.

Don't forget we need hardware and not just more software projects. I have plenty of requests for Stepper Motor circuits and explanations. Stepper motors are real black holes for many of our beginners and be sure to check out Earl Bryant's letter about making sure to explain all the computerese. Looking forward to your articles and Thanks Again...Bill Kibler.
The Z-System Corner
By Jay Sage

Advanced Applications of ZMATE
Automating Replies to GENie Mail

Two issues ago I started the discussion of advanced applications of ZMATE
made possible by its autoexec macro. This is the optional, user-written macro
that ZMATE runs when it starts up. In particular, I presented my general-pur-
pose autoexec macro that allows a macro command to be passed to ZMATE on
the operating-system command line. This opens a new world by allowing Z-
System aliases to generate automatic ZMATE sessions.

In issue 59 I presented a very simple example, my ALED alias for making
changes to my ALIAS.CMD file of ARUNZ aliases. This time I will present
a much more elaborate command and editing environment that I developed for
dealing with GENie mail.

It seems to be a general truth that the bigger and more powerful the computer,
the more primitive the user interface. This is certainly true of the GENie mail
system. Received mail can be read on the screen only sequentially by line; you
can't scroll back to look at something that has disappeared off the top of the
screen. Mail to be sent has to be entered line-by-line; there is no automatic line
wrapping, and the editing facility is extremely primitive. You can do is
insert a new line, delete an existing line in its entirety, or replace phrases in a
line with other phrases. There is also no way to quote from the message to which
one is replying. This is just the way the BBS software on my Z-Node works, and
it was, I believe, the first BBS program

ever written. GENie (and CompuServe)
still have 1979s messaging technology!

My approach for dealing with situations
like this is to migrate the operations from
the host computer to my own com-
puter. A MEX script dials up GENie and
captures all waiting messages into a
single file, which it automatically names
according to the date (e.g., GE-9305.10
for mail captured on May 10, 1993).
The scripts and macros I will describe
here automate the process of preparing
files containing reply messages. An-
other MEX script calls GENie and al-
lores me to upload the replies. I could
do this entirely automatically, but I prefer
to watch and have control over each
step.

The GENie Mail Replying Environment

Before going into the alias scripts and
ZMATE macros in detail, I would like to
describe what they are intended to ac-
complish and how they are used.

The process begins when I enter the
simple command "GEREPLY". The
GEREPLY alias generates a command
line that invokes ZMATE and runs a
macro. That macro loads the macro
stored in the file GEREPLY0.MAT into
an editing buffer and executes it. That
macro, in turn, loads a bunch of other
macros into other ZMATE buffers for
later use. You will see shortly how and
why things are done this way. The final
act of GEREPLY0.MAT is to read in a
list of captured mail files.

Now I move the cursor to the line with
the mail file I want to work with and
press the key sequence back-apostrophe
followed by period. My version of
ZMATE binds this key sequence to an
instant command that executes the con-
tents of text buffer 9 as a macro. Buffer
9 has previously been loaded with the
file GEREPLY.MAT. That macro
switches to buffer T and opens the file I
was pointing to for editing.

Now I read through the captured mail
file. When I get to a message to which
I wish to formulate a reply, I tag (by
pressing control-T) the beginning of the
FROM line of the message and move the
cursor down past any text I might want
to quote (and, in any case, past the sub-
ject line). Then I again press the back-
quote key followed by the period key.
This executes the macro in buffer 9 again.
That macro is smart; it knows that this
time it was called from buffer T, so it
performs a different function.

Here's what it does. It copies the tagged
text into buffer 1, where the reply mes-
sage will be formed. It automatically
processes the header information, taking
the GENie address in the FROM line and
turning it into a TO line. The subject is
extracted and prefixed with "$RE: ".
Other, extraneous information is deleted.

To make things specific, let's look at an
example. Listing 1 shows a sample re-
cived message. To start my reply, I tag
the beginning of the FROM line and then
move the cursor down to the line after
the actual message text, before the line
with "-- John". After I press back-
apostrophe-period, I am in buffer 1 look-
ing at the text shown in Listing 2.

Notice how the sender's GENie address
has been placed on the top line, as re-
quired when a message is to be uploaded.
My own name has automatically been
put into the second line, which lists the
people to whom copies of the message
are to be sent. This way I will have a copy of any reply messages that I send. One other important thing to notice is that the sender's message has been converted from text with hard carriage returns at the end of each line to text with soft returns. It is, thus, ready for editing.

Now I want to turn part of the message to me into a quotation. I delete the parts I am not interested in and then tag the block I want to quote. Then I press the back-quote key followed by the comma key. This invokes my instant command to execute the contents of buffer 8 as a macro. Previously, the contents of the file REPLY.MAT were loaded into that buffer. That macro formats the reply, placing the string """">>"""" in front of each line and adding hard carriage returns at the end of each line. I then add my own text, using ZMATE's format mode to provide automatic line wrapping. The result is shown in Listing 3.

When I am ready to save the message to a file, I invoke the macro in buffer 9 for another time. Boy is that macro smart! Now it sees that it was called from buffer 1 and performs yet another function. It converts the text I added into hard-formatted text, and it makes sure there are no blank lines by adding a space character on any previously blank line. Now that I think about it, I'm not sure that GEnic has any problems with blank lines, but many systems do, so I make a point of not allowing them. The final result is shown in Listing 4. The macro now prompts me for a file name and writes out the reply message to disk.

At this point I can go back to buffer T and continue reading the mail. If I see another message to which I want to reply, I can repeat the procedure described above.

Details of the Command Alias

Listing 5 shows the definition for the GEREPLY command alias in my ALIAS.CMD file. Note that the 'LY' part of the name, since it follows a comma, is optional. I do this GEnic work in a directory named TEMP, so the alias starts by moving me there. It finishes by moving me back to whatever directory I started in. ARUNZ translates the parametric expression "Sh" into the drive-user designation of the current directory at the time the alias is invoked.

The real work of the alias is done by the edit command sandwiched between the two directory-change commands. It invokes ZMATE, which I have renamed to EDIT.COM and put in my SYS directory. The important part of the command is the macro invocation in the command tail. Remember that a pair of dollar signs gets turned into a single dollar sign by ARUNZ. Everything following that is the macro string that ZMATE runs on start-up.

Here's what that the command-line macro does. The first command, "B1E", switches to text buffer 1. The command "XI" then reads in a file, in this case SYS:GEREPLY0.MAT. The dollar sign (don't forget that it takes a pair of them in an alias) ends the file name string. The command "":1" then executes the GEREPLY0.MAT macro that we just loaded. Having done it's work, GEREPLY0.MAT is no longer needed, so the final command, "B1K", deletes it from buffer 1.

Details of the Top-Level ZMATE Macro

Well, so far this is still pretty simple. The command alias just passed the buck to a ZMATE macro script. So we better take a look at that script; maybe that's where the real work is done. That macro is reproduced in Listing 6, with some editing changes. As usual for ZMATE macros in listings, ESC characters are printed as dollar signs.

This script is a little longer than the ARUNZ alias script, but even it does not do much. Mostly it just loads more macro scripts from files. It switches to buffer 9 and reads in GEREPLY1.MAT. Then it switches to buffer 8 and reads in QUOTE.MAT. It's last step is to switch to buffer 2 and generate a directory listing of all files whose names follow the format of captured incoming mail messages. It positions the cursor to the top of the list and lets the user continue manually from that point. One refine-

The one additional comment I want to make about the GEREPLY0.MAT macro concerns the lines that use the "'OQN" command. That is a very interesting ZMATE primitive. It sends its string argument directly to the BIOS console output routine. In this way it can generate screen output even before ZMATE's editing screen appears (and even if that editing screen never appears, as might be the case when a macro script ends with the "'XH" command that terminates the editing session). I use those commands here just to keep the user (me) advised of what is happening. The operations take a little time, and I don't like to be left wondering if something is going awry.

Details of the GEREPLY1 Macro

Now we turn our attention to the macro code that does most of the work. This is the macro loaded from the file GEREPLY1.MAT; it is shown in Listing 7.

The first thing GEREPLY1 does is to determine in which buffer the user was working when it was invoked. The active edit buffer is returned in the ZMATE variable @B. Because the text buffer has a name, "'T", that is not a number, the value returned by @B is a little weird. The text buffer is given number 0, and all the numbered buffers are offset by 1 from their names. The code in the macro works by checking, in order, for buffers 2, 1, and "'T" and jumping to a label in the macro. If none of these jumps is taken, indicating that the macro was invoked from an inappropriate buffer, then an error message is given.

If the macro was invoked from buffer 2, then a mail file is being selected for editing. The first task is to extract the name of the file and to copy it to buffer 0. From there it can be referenced by the file-editing command (XF). The listing,
The code for the case where GEREPY is invoked from buffer 1 and has to reformat the completed reply text to make it suitable for GENie is a little more complex. First, the entire contents of the buffer is converted to text with hard carriage returns. ZMATE’s soft returns generally cannot be interpreted by other programs. This is done by one of my permanent macros, whose name is control-H. This macro is distributed with ZMATE, and I do not have the space to discuss it here.

The next job is to make sure the message has no totally blank lines. Except at the top, where we know there is no blank line, a blank line would exist only where two carriage returns follow each other. Therefore, we search for that pattern repeatedly. We don’t want the macro to get stuck when it doesn’t find any, so we turn off error trapping with the “E” command before the search. As soon as the search registers a failure, we break out of the repeat loop. So long as we do find the pattern, we back up to the empty line and add a space character.

Now comes some very interesting code. I want the macro to prompt me on the command line for the name of the file under which to save this reply message, and I want it to appear to allow me to enter the information right on the command line. The ZMATE primitive “G” puts up a prompt, but how do we make that string a variable. The answer is by using buffer redirection. The command “G’A@?” gets its prompt string from buffer 7. It waits for the user to press a key and then returns the value of that key in the variable “@K”. I am never sure how long that variable will remain valid, since other macro commands also affect it, so I generally stick it immediately into a user variable, “@8” in this case.

The simple thing to do at this point would be to add the new key at the end of the prompt string in buffer 7 and then re-display the prompt. However, several complexities must be dealt with first. The code begins by checking for a carriage return, which indicates the end of data entry. In that case, it jumps out of the repeat loop. Next, simple editing is supported by recognizing the DEL character and backing up one character. The code has to be smart enough to backspace only over user-entered material and not over the original prompt string. Finally, as a tour-de-force, I have the code check for any characters that would not be legal in a Z-System file name.

The technique illustrated here can be very handy in many situations. In addition, I recommend that you refer back to Clif Kinne’s series of columns here in TCJ. Clif used a different approach for things like this. His macros would open a space in the current buffer and allow the user to enter the information directly, with all of ZMATE’s editing capability at his/her disposal. The macro would locate the information, move it to another buffer, and then restore the current buffer to its original state.

GEREPY then provides the nicety of adding the file extension MSG if none was entered explicitly by the user. One further error check is performed. Just before dealing with the file extension, the macro script uses the command “A@T,” to save on the number stack the first character in the information entered by the user. Now the script checks that the user actually entered something (“A@S>0”). In that case, trying to write out a file would cause an error. If a filename was given, then my permanent macro “O” is invoked to write the message out to disk. The “O” macro performs the same function as the ZMATE primitive “XO” except that if a file with the specified name already exists, the macro asks if it should be overwritten.

Now we come to the final of the three main blocks of the GEREPY macro. This is the one that operates when the macro is called from buffer T. Its function is to take the tagged block of text, copy it to buffer 1, and automatically generate the proper reply format. Since we will often be formulating a number of replies, it first considers the possibility that a previous reply message still resides in buffer 1. So it checks the buffer for contents and, if any are found, asks if they should be deleted.

That business out of the way, the macro proceeds to copy the tagged text into buffer 1 and to start working with it. The first job is to get rid of everything up to the name of the person who sent the message. This it does by tagging the beginning of the text and then searching for the string “From:”. If that string cannot be found, then the macro rings the bell, displays an error message, and terminates. Otherwise, it jumps over white space to the name of the sender and deletes everything back to the tag.

Next, anything after the sender’s name on the FROM line is also deleted. The macro adds my name on the line for copies. It then removes everything up to the actual message subject and sticks “RE:” in front of that. If there is no space there already, it adds one.

Now it uses my control-S permanent macro to convert the entire text into soft-carrige-return format. At this point, I have to come back to something I skipped earlier. Astute readers might have noticed that some blank lines were inserted for no apparent reason between the TO, FROM, and subject lines. Those blank lines were to prevent the conversion to soft returns from joining those lines together into a single line. Now that the conversion has been completed, those extra blank lines have to be removed.

The last step is to position the cursor to the subject line in case one would like to modify the subject. Then one can immediately go on to entering one’s text for the reply message.

Details of the QUOTE Macro

The QUOTE macro that was loaded into buffer 8 is shown in Listing 8. I’m going to try to let the listing stand as much as possible on its own, since I don’t know how many of you still have the energy at this point to devote attention to the details. So I will comment only on some trickier points.

One of these is the command “A@WF”. I want to make sure that format mode
turned on. I cannot use just "F", because that toggles the mode, and I can't be sure what the current mode is. When "F" is preceded by a number, then it goes into format mode with the right margin set to that numerical value. I don't want to change the margin, so I use the variable "@W" to get the current value, and I use that value with the "F" command.

Next there is some rather tricky stuff to deal with the fact that the rest of the macro is designed to work with a block that may be more than exactly what is tagged. Specifically, I want to work from the beginning of the first line of the block to the end of the last line of the block. I usually put the tag at the beginning and then move down in the text, but this is not required, and I like to make my macro scripts robust. This little block of determines the starting and ending character positions in the right order. It then defines the extended block and moves it to buffer 0 for processing.

A little later comes the code that forms the quoted text. Blank lines are ignored. Otherwise, the string ">>" is inserted at the beginning of the line. The "QR" macro is used to force reformatting of the text, since the added string might force some text to be wrapped to the next line. This rewrapping does not take place automatically within a macro script. Without the "QR", it might not happen until the macro is finished, and that would be too late; we have to do it for each line. To prevent unwanted rewrapping later, a hard carriage return is then placed at the end of the modified line.

After the quotation has been formed, the text is copied back into the original buffer.

Conclusions

In this column I have tried to give you a feel for how aliases that invoke ZMATE with complex macros stored in files can be used to build whole working environments. It takes a good bit of effort to develop such an application, but if the task is one that you perform often, creating such custom environments can make your life much more pleasant.

Next time I hope to show you an even more powerful example of how MATE can be used to add "artificial intelligence" to the computer. The application I will describe will come from the PC world and will involve a combination of PMATE (the DOS version of ZMATE) and 4DOS (the DOS equivalent of Z-System).

Listing 1. A sample received GENIE mail message. The underlined ">>" characters at the ends of lines indicate hard carriage returns as displayed by ZMATE in format mode.

<table>
<thead>
<tr>
<th>Item</th>
<th>3332727</th>
<th>93/01/10 21:21</th>
</tr>
</thead>
<tbody>
<tr>
<td>From:</td>
<td>GENIE.USER</td>
<td>John Doe</td>
</tr>
<tr>
<td>To:</td>
<td>JAY.SAGE</td>
<td>Jay Sage</td>
</tr>
<tr>
<td>cc:</td>
<td>GENIE.USER</td>
<td>John Doe</td>
</tr>
<tr>
<td>Sub:</td>
<td>Replying to GENIE Mail</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Good to hear from you, Jay. I was really thrilled to hear about your ZMATE macros and ARUNZ aliases for automating replies to GENIE mail. It sure must make things easier!</td>
<td></td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>John</td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>=END&gt;</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Listing 2. This shows how the ZMATE macro turns the tagged part of the received message into a properly formatted and addressed reply.

| GENIE.USER |
| JAY.SAGE |
| RE: Replying to GENIE Mail |
| >> I was really thrilled to hear about your ZMATE macros and ARUNZ aliases for automating replies to GENIE mail. It sure must make things easier! |

Listing 3.

| GENIE.USER |
| JAY.SAGE |
| RE: Replying to GENIE Mail |
| >> I was really thrilled to hear about your ZMATE macros and ARUNZ aliases for automating replies to GENIE mail |
| >> Thanks for the kind feedback. As usual, now that I am writing it up for TCJ, I am finding many things to improve. Soon I will be posting a new version for you. Please be sure to let me know if you have any suggestions. |

Listing 4.

| GENIE.USER |
| JAY.SAGE |
| RE: Replying to GENIE Mail |
| >> I was really thrilled to hear about your ZMATE macros and ARUNZ aliases for automating replies to GENIE mail |
| >> Thanks for the kind feedback. As usual, now that I am writing it up for TCJ, I am finding many things to improve. Soon I will be posting a new version for you. Please be sure to let me know if you have any suggestions. |

Listing 5. This is the ARUNZ alias that starts the whole process.

GEREP.LY temp; 
sys:edit $s01e.xi 
sys:gereply0.mat$1b1k; 
Shb:

Listing 6. Here is the master ZMATE macro that is invoked from the command line by the ARUNZ alias.

; GEREP.LY0 MAT (01/30/93) 
; This is the master macro that loads the various macros needed by ZMATE for generating replies to GENIE mail messages. This macro will be loaded into buffer 1.

b3e 0qn Loading buffer 9 with GEREP.LY1.MAT... $ xisys gereply1.mat$ b3e 0qn Loading buffer 8 with QUOTE.MAT... $ xisys:quote.mat$ b2e 0qn Getting list of mail files into buffer 2... $ xige???????

Listing 7. This is the macro that does most of the work.

; GEREP.LY1.MAT (01/30/93) 

```plaintext
; Branch depending on which buffer we are in.
@b=3(2)     ; IF in buffer 2, jump to "2."
@b=2(1)     ; IF in buffer 1, jump to "1."
@b=0(7)     ; IF in buffer T, jump to "T."

; Give an error message if in wrong buffer
qb = ring bell
; GMacro should not be used from this buffer!
OG$ = clear message
% = end macro

; == == ;
; IF in buffer 2, open the file whose name is under the cursor.
; ===========
; 2 ; label "2"
01 ; go to beginning of current line
t ; tag it
 l ; go to next line
-m ; back up to end of line
#bc ; copy to buffer 0
bte ; go to buffer T
xk ; make sure there is no file open
OGOpening file... ; inform user
x$A@A$ = open file listed in buffer 0
n = enter insert mode
% = exit macro

; == == ;
; IF in buffer 1, format reply message and write it out to a file.
; ===========
; :1 ; label "1"
OGConverting message format... ;
[ ; REPEAT
 e ; turn off error trapping
 s $ ; search for empty line
@e() ; IF not found, exit repeat loop
+m ; back up to empty line
] ; END REPEAT
a ; back to top of file
bte ; develop prompt in buffer 7
xk ; make sure it's empty
; initialize prompt string
iName of file for message: ;
@x7 ; remember starting column
[ ; REPEAT
bte ; return to buffer 1
G$A@7$ ; put up the prompt
@x8 ; save the key entered
@8=13[ ; IF carriage return
] ; ENDIF
@8=127[ ; IF delete
bte ; go to buffer 8
@8@7[ ; delete most recent
] ; ENDIF
^ ; loop back
] ; ENDIF

^ ; loop back
} ; ENDIF

; macro loop
b7e ; go to buffer 7
@8 ; insert the new character
] ; END REPEAT

b7e ; back to buffer 7
a ; to top of buffer
t ; tag it
@7q ; move to beginning of file name
#d ; delete prompt text
'; ; save first char (if any)
e ; turn off error trapping
s.$ ; see if file type given
@e() ; IF not
z ; go to end
i$mg ; add "$mg"
] ; ENDIF
bte ; back to buffer 1
@8>0[ ; IF file name given
."A@7$ = write out using given file name
] ; ENDIF
% ; exit macro

; == == ;
; IF in buffer T, extract tagged block to buffer 1
; and format it as a reply.
.T = label "T" ;
bte ; go to buffer 1
a ; position to top
@t=0{} ; IF buffer is not empty
qb ; ring bell
GDiscard existing text in buffer 1 (yn)? ;
OG$ = clear message
@(k=x"n)(@k=x"n) ; IF answer is NO
bte ; back to T buffer
% ; end macro
] ; ENDIF
xk ; clear buffer
] ; ENDIF
bte ; back to T buffer
#b1c ; copy marked block to block buffer 1
bte ; change to buffer 1
70f ; set format mode to
70-column width
a ; go to top of text
t ; tag it
e ; turn off error trapping
sFrom: ; find first occurrence
@e() ; IF not found
qb ; ring bell
GNot proper message format! ;
OG$ = clear message
% ; exit macro
] ; ENDIF
w ; move over white space
#d ; delete to leave just the user ID
s $ ; search for space after ID
-m ; back up to space char
k ; kill the rest of the line
13i ; end the line
13i ; add a blank line
IJAY.SAGES$ ; insert copy line
13i ; end the line
13i ; add a blank line
t ; place tag
sSub.$ ; move to subject designtator
#d ; delete everything up to that point
IRE.$ ; add "RE: " to subject
@t=7[qh] ; if no space, add one
a ; make sure we're at top
3] ; REPEAT 3 times
l ; skip line (TO, CC, SUB)
k ; kill the blank line
] ; END REPEAT
-sRE.$ ; position cursor on subject line
n ; make sure we're in INSERT mode

Listing 8. The text of the macro script for converting text in a tagged block into a "quoted" format.
; QUOTE.MAT (02/07/90)
@B=1{ ; IF we are in buffer 0
QB ; sound bell and warn
GCan't run this macro from buffer 0 $ ;
% ; abort
} ; ENDIF
@WF ; make sure format mode is on
@CV2 ; remember current character position
# ; go to other end of tagged block
@c<@2[ ; IF beginning is before
] ; end (usual case)
@CV1 ; save starting address in var 1
} ; ELSE
@2V1 ; switch end addr. into start addr.
@CV2 ; and save ending address
} ; ENDIF
@1-@CM ; go to beginning of block
T ; start from beginning of line
p ; put tag there
@2-@CM ; go to end of block
@X>0(L) ; IF not beg. of line, move to next
#BM ; copy block to buffer 0
@B. ; save original buffer on stack
BE A ; go to top of buffer 0
@T=0{ ; IF no text there
QB ; beep
GNo text tagged $ ; inform of problem
B$SE ; return to original buffer
% ; quit
} ; ENDIF

[ ; REPEAT
@T=13(L) ; IF a blank line, go to next line
; and loop
l>> $ ; insert header marks
QR ; redraw for force reformat
@L. ; remember current line
L ; try to go to next line
@L=8$ ; IF block ends without end-of-line
13i ; put in CR
] ; ELSE
-M ; back up to end of line
@T=13[ ; IF it ends with hard return
L ; just go on to next line
] ; ELSE
13i ; put in hard return
QR ; make sure text is formatted
@T="_D" ; delete leading spaces
; on next line
} ; ENDIF line ended with hard return
@T=0{ ; ENDIF block ended without CR
UNTIL end of buffer
B$SE ; return to original edit buffer
BG ; copy in revised material
QR ; redraw screen
; END.
```

Real Computing
By Rick Rodman

JPEG, WORM, archivers, and industry gripes

JPEG

JPEG stands for Joint Photographic Experts Group, a group of people who got together and designed a standard method for compressing color and grayscale images. Why do we need such a thing, you might ask, when we have PCX and GIF?

Let's step back and look at the big picture. There are two basic kinds of lossless compression that people use today. One is called "run-length encoding" (RLE), and this forms the basis of the fax compression schemes, Group 3 (Modified Huffman) and Group 4 (Modified Read). Since each pixel can only be either black or white, you can encode the count of white pixels, the count of black, etc. In Group 4, there is a "two-dimensional" scheme which encodes the movement of black-white transitions from line to line. RLE can be used on color images, too, as in PCX compression. However, the problem with photographic-type color and grayscale images here is that there are a lot of different colors subtly blending into each other, instead of big stripes or blocks of solid color. When you try to compress these with PCX, they sometimes get larger instead of smaller.

Most disk and tape compression utilities use a second scheme in which patterns are detected in the source material and replaced by codes. This is usually called LZW for Lempel, Ziv, and Welch, three folks who contributed to its design. This scheme works well when there are lots of repeated patterns in the source material, which is usually true of text files, executable programs, and drawings without lots of color - but this is seldom true of photographic images. Compressing such images is unlikely to yield even a 2-to-1 compression.

JPEG uses a direct-cosine-transform, where the image is remapped into the frequency domain, followed by Huffman coding. When the image is in the frequency domain, by throwing away high-frequency components, you can reduce the size of the image as much as you like, as long as you can accept the loss of fine detail that results. Actually, the loss of detail isn't that bad for even fairly high compression, like 20-to-1.

If there is any problem with JPEG, it is that the "standard" is loaded with options, features you can use or skip as you desire. For example, it is common in video to "subsampling" the "chroma space", that is, limit the bandwidth of color changes, since color changes less frequently than intensity. By digitizing your image in the YCbCr color space, intensity-blue angle-red angle, as in video, rather than the red-green-blue (RGB) space, you can reduce by three-quarters the amount of data for the Cb and Cr components without losing much picture quality at all. Then there's the issue of the file format, which the standard doesn't touch much at all.

The International JPEG Group (IJG) is a group of enthusiasts, more or less headed by Tom Lane, who have developed a code package implementing compression and decompression to and from JPEG using the JFIF file format. As with most projects where people work on it because they want to, this is a very robust and portable package. It also has pretty good performance, because the sines and cosines are all calculated in fixed-point. No viewer is provided; there are two programs, cjpeg to compress, and djpeg to decompress. Command-line options are used to select the amount of compression you want and any subsampling or other options. All source is included, of course, so you can easily make these into subroutines and link them into your own programs.

The software is available via ftp or from mail servers, or you may send me a diskette with return postage and I'll put it on there for you.

Alas, there is a black spot on JPEG, and it is the fact that some variations of it are covered by software patents. As we all know, the patent system has been twisted from a protection for the small into an anticompetitive bludgeon of the huge. Patents on software algorithms have been granted even though expressly prohibited by the law, and they have to be regarded as unethical at best, and a crime against the community at worst. IJG rightly refuses to include the algorithms in question, and I agree with them.

Ironically, while public-key cryptography has been considered the best scheme for private citizens, all the public-key schemes, like RSA, are covered with software patents. It is up to all of us, even private citizens who may not be aware of the issues, to reject any patented scheme out of hand, and ostracize the scam artists that would benefit from this abuse.

Next time, I hope to have some benchmarks running the code on images of various sizes on various platforms. Because this algorithm is heavy on num-
ber-crunching, I expect 32-bit environments to do substantially better.

WORM drives

Corporate Systems Center is offering 800 megabyte WORM drives for $269, with media cartridges for $99. Is this an incredible deal? It sure is! But before you write that check, think for a moment about the nature of WORM: Write Once, Read Many.

Each block of a WORM disk can be written only once.

Normal computer disk drives don’t make this restriction, and most software is not written to work that way. If you think about it, it has some considerable ramifications. On most WORM drives, if a sector turns out to be bad, you have to write another sector somewhere else. It so happens that the Maxoptix drives CSC is offering have a remapping feature, but most drives don’t. So, you have to write sectors in a way that you don’t have preassigned locations for certain things, like FATs.

There are three general approaches to using WORM drives with computers.

First and fastest, you can write drivers yourself and map the data any way you like. My drivers work this way; it’s fast and simple. I write out a file, then write a file header which tells where I wrote all of the blocks. If I get a bad sector, I skip over it. There’s no FAT or root directory. For speed, I keep a directory file with the front part of the file headers on my magnetic hard disk. This approach is operating-system-independent, of course, since the operating system doesn’t manage the WORM storage.

Second, slower, is the “block-mapping” approach. Here, the WORM, through a driver, either software or the Ten-X box CSC offers also, pretends to be an erasable hard drive. What really happens is that pretend hard drive blocks are mapped dynamically to WORM blocks. Each time you write, a different block is mapped. Now, the operating system sets up the pretend hard drive any way it likes, with FATs, root directory, etc. Corel and Opti-Driver use this approach. However, because everyone does things slightly differently, there is no portability between different computers - you’ve got to have the same drive, the same media, and exactly the same software.

Third, and slowest, is the “journaling” approach. Here, a true filesystem is set up, but an operating-system-independent one. Records are written sequentially to the WORM containing the changes to the virtual filesystem as they take place. As the disk fills up, things get slower and slower, because the computer has to read through the history of changes to construct an image of something stored on the disk. A filesystem driver (redirector) is necessary to allow an operating system like DOS to access files on the disk. The good side of this approach is that the same disk can be read on several different computers. Pegasus uses this approach.

The optimal would be a true write-once filesystem. Files, once written, would not be erasable. Instead, there would be versions, like under VMS (a version tracking system that allows retrieval of a selected version). However, normal operating system utilities could be used to access the file. Many people write about WORM as if this ideal solution existed, but as far as I can tell, it doesn’t. Thus, the realities of WORM fall far short of its potential.

Pundits have predicted for years that WORM will be replaced with erasable optical. It seems to be slowly happening today. However, WORM still has some advantages: first, the media is much cheaper; second, the media is more reliable, and data stored on it is truly permanent and will be there, almost guaranteed. Magneto-optical erasable drives are faster, but less reliable, and certainly not permanent, regardless of what certain manufacturers (mainly HP) will tell you.

There is also a tremendous new technology on the horizon: ICI’s digital paper. If ICI can get this technology out in affordable products, we could be talking pennies per gigabyte. And guess what? It’s write-once.

File archivers

In the PC world, Zip is the most common, although Zoo and Arc are popular. In the CP/M world, LU, usually combined with SQ, is the most popular. In the Unix and Minix worlds, tar, usually followed by compress, is the most popular.

Some time ago, we NS32 folks sort of settled on Zoo, because source code was available. Today I’m not so sure. I have machines running in all three environments, and I’d still like a common way to package files to transfer them around, and I still don’t have one.

In the Unix/Minix world, usually a tar file is constructed as a file, and the output of tar passed through compress. Tar is an ancient program which combines files in uncompressed 512 byte blocks with 512 byte file headers sprinkled throughout. Using compress makes this a two-step operation, since tar can only work on the uncompressed file. Even if you only want to extract a single file, you must decompress the entire tar file.

When we use LU under CP/M, if you want to compress, usually you compress the file with SQ before you put it in the library. Then, LU can still be used on the resultant archive. I assume that the Unix/Minix folks do it in the other order because of the relatively large 512-byte blocks and file headers, which greatly increase overhead. LU’s overhead is only 32 bytes plus padding to 128-byte blocks.

Another thing about tar which is not very good is the fact that the file headers are sprinkled throughout the archive file. To list the files on a tape on my Sun, for example, could take an hour or so. Zoo and Zip work this same way. If a file becomes damaged in the middle of the archive, files following it could become unreadable.

There are two variants of tar. One writes entries for the directories themselves as
it moves through directories, and the other doesn’t. This can be a significant problem if you’re unpacking an archive made by the latter type into a program expecting the former type - the directories won’t exist.

At the moment, it appears that a quick port of LU to each of the environments would give me a rudimentary capability to package files for transfer. LU doesn’t support subdirectories or time/date stamps. It’s also limited to “8.3” format filenames. But the great part is that the source is only 716 lines long.

It should be possible to make a version of tar which passes all of its file I/O through a lower layer consisting of the compress logic, since all of tar’s I/O is basically sequential. The resultant program would be able to read and write ‘.T’ (tar.Z) files directly. I’m hopeful that I could avoid the code bloat which has struck all of the PC packages as they compete with each other over bells and whistles.

**Industry gripes**

In the PC world, the KISS principle is dead and buried. But with code bloat inevitably must come bugs. Lots of them.

Microsoft’s Windows for Workgroups is full of bugs. It’s difficult to get it to work at all, and once working, it’s very fragile. It runs extremely slow unless Smartdrive is used, but Smartdrive still hasn’t been fixed and frequently locks up with a spurious “serious error”.

Then there’s DOS 6, which can be summed up in one word: Don’t.

And then there’s Windows NT. They’re calling the first release Version 3.1. From what I’ve heard, they should be calling it Version 0.6 - or maybe Version 0.31.

One has to wonder whether Microsoft’s support will be able to handle three buggy products coming out at the same time, all steeply discounted. Maybe it’s a ploy to alienate as many of their customers as possible. While I don’t see any point to DOS 6, the other two are potentially good products. It’s a shame neither lives up to its potential.

In an unrelated vein, Microcom’s Carbon Copy for Windows gets honorable mention for short-lived major version numbers. CCW 2.0 is out, only two months after the release of CCW 1.0. The simple reason is, of course, CCW 1.0 didn’t work. Only in the PC software market can companies build market share by shipping products that just plain don’t work. Can you imagine GM or Ford doing that?

**Next time**

In addition to JPEG benchmarks, I’d like to include some information about the various code archives on the Internet and ways I’ve found to access them. I’ll also include some news about what’s going on in the Minix and PC-532 worlds.

**Where to call or write**

BBS: +1 703 330 9049 (eves)

Corporate Systems Center
730 North Pastoria Avenue
Sunnyvale CA 94086
+1 408 737 7312

---

**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. **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.

<table>
<thead>
<tr>
<th>Subscription</th>
<th>U.S.A.</th>
<th>Canada/Mexico</th>
<th>Europe/Other Countries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Year Surface</td>
<td>$24.00</td>
<td>$32.00</td>
<td>$34.00</td>
</tr>
<tr>
<td>1 year Air</td>
<td>$34.00</td>
<td>$34.00</td>
<td>$44.00</td>
</tr>
<tr>
<td>2 year Surface</td>
<td>$44.00</td>
<td>$60.00</td>
<td>$64.00</td>
</tr>
<tr>
<td>2 year Air</td>
<td>$64.00</td>
<td>$64.00</td>
<td>$84.00</td>
</tr>
</tbody>
</table>

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.
Regular Feature

Operating Systems

By Bill Kibler

This is the first part of a new regular feature for beginners and advanced users of operating systems. I have received many requests to explain about operating systems, and CP/M in specific. The Computer Journal has many readers, new and old who have not used CP/M, but have worked on other operating systems.

For classic computer users, the CP/M story is also very interesting from an historic point of view. MSDOS and many of the small computer operating systems, all have their roots in CP/M. The Atari ST operating system, is actually a variant of CP/M 68K with window enhancements.

One of my enjoyments is tinkering with operating systems. I have found CP/M to be an excellent starting point for those seeking to understand the inside of an operating system. CP/M has another feature for tinkering that makes it special, mainly access to all the source code. Now MINIX source code is available as well, but I have found this code a bit hard to deal with if you are not an experienced “C” programmer. CP/M is done in 8080 assembler, is not very large in size, and is some what simple in design. All those features make it more suitable as a learning platform than MINIX.

HISTORY

To start out on this journey, I find it fitting to review how CP/M came about. This walk down memory lane has some personal significance for me. I was just starting to become involved in computers when all this was happening and have some first hand stories about the product and time period it happened in.

The time period is late 1970. I was aware that IMSAI had just made their version of S-100 computers the buzz words of the industry. The typical computer would have 3 or 4 4K memory cards and the IMSAI MPU A CPU card (8080). The I/O (or input/output) devices were limited to say the least. Those lucky enough to have a terminal or video screen and keyboard could actually type in data and not use tape readers or cassette drives for data.

As these early machines found more memory, programmers started to become serious about performing actual work and not just lighting up LEDs (solid state lights) in interesting ways. Trying to stay one step ahead of the competition, the folks at IMSAI decided they needed better storage systems. Now I am not 100% sure how all this came about and if my memory is correct or not, but then readers can certainly write and correct me if I error too far astray.

To continue, IMSAI needed better storage, and IBM had a floppy disk system they were using on the mainframe systems. These were 8 inch floppy drives that stored data much as tape cassettes did, only the medium moves around on a platter much like a record player. To this day the 8 inch format is called IBM Single Sided Single Density format and is the only true floppy standard.

Since there were no disk controller chips at that time (late 1970s), IMSAI designed a controller board using TTL logic, it was called IMSAI 8080 IFM disk controller set. Now reading and writing to the disk was no simple task. Another major problem was I/O could be anywhere in system space. Some systems used memory mapped versus the I/O addressing scheme of the 8080 CPU chip. Writing programs for general use in which the I/O would need changing before you could use it just didn’t seem practical.

Now this is fuzzy here, but I think IMSAI found Gary Kildahl, or Gary maybe found them. The results however was CP/M and for IMSAI users IMDOS or IMSAI Disk Operating System. Gary had worked on some mainframes early in his career and saw things he liked. When the microcomputer revolution happened he tinkered away many a late night and put together a Control Program for Microcomputers (thus CP/M). Since the IMDOS operating system came just as IMSAI was falling apart and was a separate variant of CP/M it was lost in history (I do have copies if you want them). However Gary had by that time ported CP/M to many other systems and formed Digital Research.

CP/M’s INNER WORKINGS

The reason CP/M was important, besides being first on the market, was the isolation of I/O and it’s modular design. As I had stated earlier, any program before CP/M would have to write directly to output devices. Change one of those devices, like buy a new terminal with different serial control lines and screen control codes, and zap, a rewrite of the code was needed. CP/M changed all that. Three separate and distinct layers make up CP/M, Console Command Processor (CCP), Basic Disk Operating System (BDOS), Basic Input Output System (BIOS). To write a program you simply make request calls to the BDOS and it will translate that request into as many requests to the BIOS as needed to perform the desired operation.
Since CP/M would provide the same disk system calls on any vendors products, programmers could now write a program that had no idea of what type of I/O the computer contained. Each vendor however did have to write their own BIOS and as a user, if you made major changes in the I/O design, you would have to change your BIOS as well. This still didn’t help with changes to different terminal control codes, as all the BIOS knew about was how to send the data to the terminal. What the data did was still the responsibility of the original program.

From a history point, remember there wasn’t very much I/O (Input/Output) around in these days. In Dave Bursky’s “The S-100 Bus Handbook” (circa 1979/80) he prices disk systems at $1500 for two drives and a controller. I remember paying $300 for IMDOS and seem to understand that CP/M was about the same. A serial controller card would cost $300 and terminal kits might be another $300 or more. That was all on top of the initial system cost of $500 and much more. For memory, a set of 16K RAM chips might cost as much as $20 each chip!

Since I worked at Micro Pro when they were designing and building the MPB1000 (actually did the schematic drawing and helped trouble shoot the original design), I was lucky to see source to an early versions of CP/M. Like any product, it went through many improvements and bug fixes. The first version was 1.0 followed by bug fixes 1.1. An enhanced version followed called 2.0 with rapid bug fixes to 2.2. In the early 1980’s, machines were getting more memory and like the MPB1000 they were paging (putting a movable window in memory that can be switched between different sections of memory) and a more complex CP/M was needed. People also wanted to run more than one program at the same time, just as they still do with MSDOS.

The people at Digital Research, the company formed by Gary Kildahl, satisfied users with new and improved versions of CP/M. MP/M is the multiuser version of CP/M. With MP/M (Multi-Programming Monitor Control Program for Microprocessors) you can have more than one person using the computer at a time. The MP/M operating system used a single CPU that time sliced (one user for half second, next user for half second and so on...) between tasks or users. For users with more memory, but only a single user, CP/M 3.0 was rolled out. 3.0 was suppose to have so much more to offer, that when IBM approached Gary for an operating system for the soon to be IBM PC, they refused to take time away from 3.0 production to modify their current CP/M86 product to IBM’s requests. IBM turned instead to Bill Gates, who bought a version of CP/M86 from Seattle Computers (a S100 vendor making 8088 powered systems) and the rest is history.

About this time I had started to work for Teletek in Sacramento California, where we were using a modified version of MP/M for their multiuser systems. What makes these systems different, was the multiple CPUs in use. Each user had a single Z80 based computer on a S-100 card, it talked to a master disk controller CPU card. The advantage was that each user only needed to use common resources when they talked to the systems hard disk or printer. At all other times the user was talking to their own CPU and the programs ran at full speed without any delays. Multiuser systems before that could slow to a creep if more than three or four users were working at one time.

A Look Back
As I look back over the years, I see how CP/M got microcomputer off to a good start. It was not the first nor last operating system. It was however one that appeared on the market at the right time. It works well for the tasks it needs to do. It is not overly burdened with tasks to perform, thus making it good for beginners to understand and deal with.

Wordstar would never had been a product, had it not been for CP/M to run on. Microsoft would not have been the giant it became without making money on Microsoft Basic for CP/M and later producing MSDOS. The roots and design of MSDOS are those of CP/M. Understand CP/M and you understand MSDOS.

The DESIGN
What do you get when you get CP/M. If we use a Kaypro as an example, CP/M comes on one disk. You get the operating system and BIOS as one product. No source is provided on the disk for the system or the Kaypro specific BIOS. That BIOS was available for an extra fee, although sometimes vendors would guard their BIOS’s and make you sign non-disclosure statements before shipping to you (after paying a fee as well). I personally was ransomed this way by several vendors and felt it did more to hurt the industry than help it.

The CP/M disk does contain several other programs, called utilities. Since each hardware platform had their own disk format, the vendor would supply a disk formatting program. To copy files from one disk to another, a copy program called PIP was supplied. A basic single line editor was also provide for emergency work.

CP/M did come with some built in commands and they are:

- ERA to erase or delete a disk file.
- DIR to display a list of file names and sizes.
- REN to let you change or rename a file.
- SAVE to save a section of memory as a disk file.
- TYPE to let you read the text information in a file.
- USER to change the group of files currently active.

CP/M has programs you can use to help maintain the system. They are:

- STAT which provides detailed information about disk space.
- ASM which can be used to assemble a program using 8080 assembler mnemonics.
- LOAD which takes a hex file (output of the assembler) and converts it into a .COM or executable program file.
- PIP which is a Peripheral Interchange Program for transferring files and data between two devices or disks.
- ED is the line oriented text editor.
A Future Goal

Operating systems have always interested me, and lately I have a special interest in getting an universal operating system going. That interests stems from having so many different types of systems that have totally incompatible operating systems and disk structures and yet each can offer some special service not available from other systems. Typically the operating system has little to do with the features I desire. It often in fact just gets in the way. I have looked at all the C based operating systems, and they are either too complex or no C system is available for my platform of choice.

I have turned lately to Forth, which can provide more than just operating independence. Forth provides tools and conceptually CPU independence. Forth creates an imaginary CPU design in software. You program to the conceptual design. The concept is moved form platform to platform so that programs properly written will run unchanged on different platforms. It is this use of code (Forth code) without changes on different architectures that interest me the most. It would also allow TCJ to provide programs that should run on any of the classic systems using Forth. Think about it, the same tools and commands on all your older systems, no matter size or type of CPU.

Over the next few issues, TCJ will explore more about CP/M in such a way as you can understand operating systems better and consider my personal goal of a universal operating system in better detail. Operating systems are complex for the outside observer until they understand their inner workings. As we look into CP/M's inner workings and consider how these same items are done differently by other systems, you too might develop a personal goal while understanding not only CP/M but operating systems in general.

So stay tuned for the next inside CP/M article.

---

**CLASSIFIED, FOR SALE AND WANTED**


Wanted: Digital to analog converter boards for S-100. Looking for multi-channel DAC with 8 to 16 channels with 8 bit resolution or better. Also good single channel 16 bit DAC. I believe Cromeco made some DAC boards? Also looking for good S-100 graphics card with doc/software. Need FULL C compiler with FP for CP/M 68K. Chris Christensen, 2780 Concord Way, San Bruno, CA 94066. (415) 588-6965.

Wanted: Information on interfacing a hard disk to NEC 8801A. Also any help with linking into PC Networks? Vernon Wankerl, 1504 S. Spring St. Springfield, IL 62704.

Wanted: Need help with XEROX 820-II, trying to find MBASIC, WS, and other utilities. Does W5.0 for Commodore 128's work on the 820-II. Would like to find 820-16 (PC plug in card) and any other information on new products. Gregg Eshelman, 400 W. Commercial, Wieser, Idaho. 83672, (208) 549-1539.


Wanted: Help fixing/repairing ZX-81/1000 membrane keyboard. Is there a plug and play replacement for it? Have only found ones that need to be rewired. BLU, 517 N. Reynolds Ave., Canon City, CO. 81212.

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 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 $.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.
The XEROX 820 information provided here is both historical and important for classic users. The first single board system and the unit that started Micro Comunicopia was the Big Board by Digital Research Computers of Garland Texas. Both Xerox and Kaypro used these boards with some modifications to produce their first microcomputer systems.

A number of years ago, Xerox unloaded hundreds of these systems and their older boards. I picked up two boards for $25. They still work and are still available at most swap meets. When I got my boards, the vendor had several boxes of 20 boards each. Typically documentation is not supplied with these do it yourself units. Software is fairly common, but what pins do what and how to hook up the serial ports are common problems.

The Ham Radio people found these units made excellent TNC computers (Terminal Network Controller for PACKET RADIO) and are discussed in various magazine articles and Radio Handbooks. X.25 operation is possible and public domain software is easily found to support these machines.

Over the next three issues we will present the schematics and interfacing information. Big Board and Kaypro are very similar and thus these drawings may help in servicing those units as well. BDK.
### Modem Connector

<table>
<thead>
<tr>
<th>PIN</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>Transmit Data</td>
</tr>
<tr>
<td>3</td>
<td>Receive Data</td>
</tr>
<tr>
<td>4</td>
<td>Request to Send</td>
</tr>
<tr>
<td>5</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>7</td>
<td>Ground</td>
</tr>
<tr>
<td>8</td>
<td>Carrier Detect</td>
</tr>
<tr>
<td>13</td>
<td>Transmit Clock</td>
</tr>
<tr>
<td>17</td>
<td>Receive Clock</td>
</tr>
<tr>
<td>20</td>
<td>Data Terminal Ready</td>
</tr>
</tbody>
</table>

### 8 Bit General Purpose Parallel Port Connector

<table>
<thead>
<tr>
<th>PIN</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>port A STROBE</td>
</tr>
<tr>
<td>4</td>
<td>port A READY</td>
</tr>
<tr>
<td>6</td>
<td>port A bit 0</td>
</tr>
<tr>
<td>8</td>
<td>port A bit 1</td>
</tr>
<tr>
<td>10</td>
<td>port A bit 2</td>
</tr>
<tr>
<td>12</td>
<td>port A bit 3</td>
</tr>
<tr>
<td>14</td>
<td>port A bit 4</td>
</tr>
<tr>
<td>16</td>
<td>port A bit 5</td>
</tr>
<tr>
<td>18</td>
<td>port A bit 6</td>
</tr>
<tr>
<td>20</td>
<td>port A bit 7</td>
</tr>
<tr>
<td>22</td>
<td>port B READY</td>
</tr>
<tr>
<td>24</td>
<td>port B STROBE</td>
</tr>
<tr>
<td>26</td>
<td>port B bit 0</td>
</tr>
<tr>
<td>28</td>
<td>port B bit 1</td>
</tr>
<tr>
<td>30</td>
<td>port B bit 2</td>
</tr>
<tr>
<td>32</td>
<td>port B bit 3</td>
</tr>
<tr>
<td>34</td>
<td>port B bit 4</td>
</tr>
<tr>
<td>36</td>
<td>port B bit 5</td>
</tr>
<tr>
<td>38</td>
<td>port B bit 6</td>
</tr>
<tr>
<td>40</td>
<td>port B bit 7</td>
</tr>
<tr>
<td>42</td>
<td>Ground (ETC #2 CPU only)</td>
</tr>
</tbody>
</table>

### Counter/Timer Option (Terminal)

<table>
<thead>
<tr>
<th>PIN</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>System Clock</td>
</tr>
<tr>
<td>3</td>
<td>ZC/TOO</td>
</tr>
<tr>
<td>4</td>
<td>ZC/TO1</td>
</tr>
<tr>
<td>5</td>
<td>ZC/TO2</td>
</tr>
</tbody>
</table>

### 80 Factory Settings

- 20 factory settings.

### Keyboard Connector

<table>
<thead>
<tr>
<th>PIN</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BIT 0</td>
</tr>
<tr>
<td>2</td>
<td>BIT 1</td>
</tr>
<tr>
<td>3</td>
<td>BIT 2</td>
</tr>
<tr>
<td>4</td>
<td>BIT 3</td>
</tr>
<tr>
<td>5</td>
<td>BIT 4</td>
</tr>
<tr>
<td>6</td>
<td>BIT 5</td>
</tr>
<tr>
<td>7</td>
<td>BIT 6</td>
</tr>
<tr>
<td>8</td>
<td>BIT 7</td>
</tr>
<tr>
<td>9</td>
<td>STROBE</td>
</tr>
<tr>
<td>13</td>
<td>+5 volts</td>
</tr>
<tr>
<td>20</td>
<td>Ground</td>
</tr>
</tbody>
</table>

### Printer Connector

<table>
<thead>
<tr>
<th>PIN</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground</td>
</tr>
<tr>
<td>2</td>
<td>Receive Data (Input to 20)</td>
</tr>
<tr>
<td>3</td>
<td>Transmit Data (Output from 20)</td>
</tr>
<tr>
<td>4</td>
<td>Clear to Send</td>
</tr>
<tr>
<td>5</td>
<td>Request to Send</td>
</tr>
<tr>
<td>6</td>
<td>Data Set Ready</td>
</tr>
<tr>
<td>7</td>
<td>Ground</td>
</tr>
<tr>
<td>8</td>
<td>Data Terminal Ready</td>
</tr>
</tbody>
</table>

### Modem Port Option (Terminal)

<table>
<thead>
<tr>
<th>PIN</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>(M) TXD to Pin 3</td>
</tr>
<tr>
<td>10</td>
<td>(T) TXD to Pin 2</td>
</tr>
<tr>
<td>13</td>
<td>(M) RXD from Pin 2</td>
</tr>
<tr>
<td>17</td>
<td>(T) RXD from Pin 1</td>
</tr>
<tr>
<td>19</td>
<td>(M) RTS to Pin 5</td>
</tr>
<tr>
<td>23</td>
<td>(T) RTS to Pin 4</td>
</tr>
<tr>
<td>25</td>
<td>(M) CTS from Pin 6</td>
</tr>
<tr>
<td>27</td>
<td>(T) CTS from Pin 5</td>
</tr>
<tr>
<td>29</td>
<td>Clock supplied to Modem as RX Clock</td>
</tr>
<tr>
<td>31</td>
<td>Clock supplied to Modem as RX Clock</td>
</tr>
<tr>
<td>33</td>
<td>Clock supplied to SIO with RX Clock</td>
</tr>
<tr>
<td>35</td>
<td>Clock supplied to SIO with TX Clock</td>
</tr>
<tr>
<td>37</td>
<td>Clock supplied to SIO with TX Clock</td>
</tr>
<tr>
<td>39</td>
<td>Clock supplied to Modem with TX Clock</td>
</tr>
</tbody>
</table>

### 80 Factory Settings

- All odd # pins are grounded.

Refer to Parallel Ports in the Software section of this manual for a description of these jumpers.

NOTE: (M) indicates modem (data communications equipment) function. (T) indicates terminal (data terminal equipment) function. For instance, exercising the (T) strap option will allow communication with a modem. Exercising the (M) strap option would allow communication with a terminal.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESSS5</td>
<td>9400 VDC 30 A DC maximum</td>
</tr>
<tr>
<td>Power</td>
<td>30 A p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Frequency</td>
<td>90 V p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Power</td>
<td>30 A p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Frequency</td>
<td>90 V p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Power</td>
<td>30 A p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Frequency</td>
<td>90 V p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Power</td>
<td>30 A p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Frequency</td>
<td>90 V p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Power</td>
<td>30 A p-p at 20 KHz maximum</td>
</tr>
<tr>
<td>Frequency</td>
<td>90 V p-p at 20 KHz maximum</td>
</tr>
</tbody>
</table>

The graphic representation shows the block diagram of the CRT. The CRT generates a signal to control the position of the electron beam on the screen, allowing for the display of characters and images. The diagram includes various elements such as the electron gun, deflection coils, and phosphor coating, which work together to produce the desired visual output. The CRT is a crucial component in display devices, allowing for the visual representation of data and information.

Resolution at center (without dot circles) = 1024 horizontal lines, 1024 vertical lines.
QUALITY CONTROL USING THE COMMODORE 64

By Ralph Tenny

Ralph sent me these old articles some time ago. The disks had been lost, so a friend of Rick Rodman retyped them for us. I like the article because it shows how old systems were and still can be used for simple industrial tests. I have heard that a local person is building pump controllers using Commodore 64's currently. I could have picked up several for $20 at last week's swap meet. Just try and buy a new pump controller for that price. Another feature of this article is using the tools at hand, namely BASIC. What Ralph did was simplify the task which allowed using the simple tools that came with the computer. BDK.

INTRODUCTION

Whenever a computer is used in a real-life situation, some choices must be made. The major choices are: "Which computer?" and "What programming language?" This discussion reports the details of an industrial testing application using the Commodore 64 computer running only Commodore BASIC. The emphasis is on explaining the techniques used to improve program speed and to describe program features which enhance the usability of the reports generated.

THE COMPUTER

The Commodore 64 is a low-cost computer which has more user-accessible I/O lines than any similarly priced computer. This combination makes the C-64 highly adaptable in industrial testing and control applications. Figure 1 details the User Port lines and their possible applications. The eight programmable I/O lines are the most accessible of the I/O features, and are used in the application described below. Besides several I/O lines, the C-64 has two Real Time Clocks which are available for user programming. Figure 2 shows the basic arrangement of the real time clocks. Four eight-bit registers contain six digits of time, giving time resolution to one-tenth of a second.

THE LANGUAGE

For applications which may change fairly often, easy program modification may be the most important factor. If BASIC is available on the computer being used, it may be a good choice. However, if the process being controlled or monitored operates at high speed, BASIC may not be able to do the job. If compiled BASIC is available, it will give additional speed at the expense of flexibility and ease of program changes. Also, additional program speed can be gained by using assembly language routines, since they can be linked to most dialects of BASIC. If you use this approach, many of the "quick change" advantages of BASIC may be lost. If assembly language must be used due to speed considerations, even minor changes in program and operation can become expensive.

THE TEST

The modules under test were microcomputer-controlled devices which use a telephone line to receive and transmit data automatically. Once the modules are put into service, they are expected to function continuously with no scheduled maintenance. Consequently, a thorough inspection and burn-in process is used to help insure reliability. Normal production test procedures include repetitive telephone calls to prove normal functioning. After the module passes all performance tests, it is placed on extended burn-in. The burn-in activity needs to be monitored to test the programming and hardware used for telephone calling. While this could be accomplished by allowing the calls to be placed over the telephone lines, testing of multiple units would require extensive telephone facilities.

If a module can't complete a call at the programmed time, it retries every ten minutes until the call is completed. The calling process is similar to dialing a normal telephone, except that it is controlled by a computer. Telephone modems initiate a call by placing a load on the telephone line when a relay contact is closed. The telephone company senses the load and responds with a dial tone. The modem then places the call by generating DTMF tones like the tones generated by a pushbutton telephone.

The Commodore 64 was connected so that each time a module relay closed, one line of the 8-bit User Port was pulled low as shown in Figure 3. Since the monitor circuit can't generate a dial tone, the module under test continues to try placing a call as long as the unit is working properly. With eight modules on test, one Commodore 64 replaces eight telephone lines. This makes the test process very cost effective. The test results are saved in two arrays which are printed out after the test to provide proof of performance.

THE PROGRAMMING

Two different programs were developed over a period of weeks, working to improve recording of test values and to minimize direct involvement by test personnel. The two listings below demonstrate different testing approaches. Listing #1 monitors units on test, while List-
ing #2 is a trouble-shooting tool for investigating problems of a failed unit. Program #1 can detect separate events which occur within two or three seconds of each other, and save results from eight units under test. The emphasis in Program #2 is to detect logic level changes and other events as soon as possible. The sampling resolution of program #2 is adequate to detect events occurring less than one second apart. If you compare the two programs, the first one stresses long-term monitoring of multiple units and moderate sampling resolution. The second program minimizes time between samples to give a more accurate chronological report of the test activity. Note that both programs have two kinds of resolution. The sampling resolution is determined by how fast the program can run the central loop. The time resolution depends on which time increment is read from the real time clock (RTC). Program #1 reads the “minutes” register and reports events monitored during the test in one minute intervals. Program #2 uses one second intervals in the time record.

Timekeeping in these programs has been streamlined to speed program operation. Rather than read all four timer registers and deal with six digits and the AM/PM flag, only one register is read. The result is compared to the previously saved value, and a counter (variable TM) is incremented each time the register changes. This procedure minimizes processing time during program operation, and reduces the amount of storage used for test results. If real time is important in the test report, the starting time of the test can be recorded and real time computed as the test results are printed.

Both programs initialize the test hardware the same way. For now, we will pass over the input prompts and begin with line #60. This line starts the RTC and line #65 reads one register of the RTC. This two-digit value is stored in D for future reference. Next, line #70 samples the port and skips any processing if no port lines are low. Otherwise, line #75 stores the port bit pattern in the results array (P) and the current time token (TM) in the time array (T). The array index (Q) is updated in line #80 and tested for end of array. If the array is full, control passes to the output routine at line #4500. If the array has space, the program drops into the main sampling loop at line #170.

The main activity in either program is centered in lines #170-190. In #170 a timer register is read and the contents compared with a previous sample. If the time has changed, line #175 saves the new time, increments TM and prints TM to the screen to show the program is running. If the timer register hasn’t changed, line #180 tests the keyboard buffer and branches to line #300 if a key has been pressed. Otherwise, line #185 samples the User Port and tests for a change. With no change, control transfers to #170 to start another loop. If the port has changed, the new port value is saved in F, and control goes back to line #75 to save the port image and time. Also the current time value is stored in array T. This loop continues until array P is filled or the test is interrupted.

**USER CONTROLS**

When the keyboard buffer is tested in line #180, presence of any character except ‘P’ (41 is the commodore code for P) is in the buffer, control is transferred to line #400. This routine reads the ten latest values in array P and prints the value for each port line which was active when the value was read. This allows the operator to verify that each unit is functioning at the start of the test. The subroutine at line #110 reads the port image and determines if a particular port line was low when the value was read. Thus, only active lines are reported to the screen.

If the character ‘P’ is found in the keyboard buffer, the test is terminated and the results printed by the routine beginning at line #4500. This routine also uses the bit sorting subroutine to identify which lines were low. Line #4520 calls a subroutine at #250 to print the difference between the current time reported and the previous reading for the same port. If this subroutine isn’t called, the actual time value (expressed as minutes or seconds since the start of the test) is printed. Figure 4 shows a sample report of time and Figure 5 shows a report of time differences.

**THE REPORTS**

Reports generated by test equipment can be as brief or elaborate as required to suit the test conditions and the type of equipment being tested. Ideally, each report generated by computer controlled testers should be automatically labeled to show the unit tested, the date and any special test conditions. Note the heading on Figure 4. On the first line appears a date, “3”, and the phrase “TM = 814”. The second line shows the equipment serial number, and the remainder of the record is test data. The “3” indicates that this was the third sequential test, which ran for 814 minutes after the previous test record was printed. (If you follow Listing #1, the test restarts at line #60 after finishing a print sequence.) Unless interrupted, the program will run indefinitely, making a new printout each time array P fills up, and each time the operator types “P” on the keyboard.

The “TM = 814” indicates that the test covered by this report ran for 814 minutes after it started (or restarted). The data in Figure 4 can be understood by remembering that any given module repeats a calling attempt on ten minute intervals. Take the first line of data:

```
5,5,5,16,16,27,27,27, etc.
```

The first 5 shows that the modem relay first closed five minutes after the start of the test. The last 5 shows when it opened again. The second 5 records the fact that another unit closed its relay before the first one opened. The first 16 shows that the unit tried to call approximately 11 minutes later. The actual calling process takes about 30 seconds, so intervals can be shown as either ten or eleven minutes. This can be seen more clearly in Figure 5, where time differences are shown instead of time intervals. The purpose of this testing program is to locate any module which exhibits a break in the normal pattern. In Figure 5, the first and last data points must be discarded, but all the rest are 1, 1, 10 or 11. Looking at Figure 4, it is possible to see
how these four figures come up. Figure 6 is a report of a test which was interrupted for 35 minutes near the end of the test. Note how noticeable this anomaly is at first glance. In Figure 4, every data point must be scanned and compared to the ones before and after in order to find a problem. If the actual time of failure is not important to the test results, the report style in Figure 5 is obviously preferable for finding problem units.

REPORT FORMATTING

Beginning with line #5, the program prompts for data needed for the report. At line #4000 through 4010, a loop prompts for device serial numbers. Line #4040 gets the date of the test and line #4050 sets the report number (R) to 1. At report time, line #4500 opens the printer and sets up a new page. During the printout, array P is scanned repeatedly, once for each bit position of the User I/0 port.

No attempt was made to make a "pretty" printout, since the printouts are inspection reports instead of reports for the board room! Line #4535 uses the subroutine at line #200 to force a newline after each set of data is printed. Line #4550 clears both arrays, and lines #3010 through 3025 close the printer, reset the time token and restart the program.

For more information contact: Ralph Tenny. Computer and Automation Hardware Consultant, PO Box 545, Richardson TX 75080

Ralph supplied two other articles that will be printed next time. One "Programming the 6526 C/A" which covers programming the Commodore 64's Complex Interface Adapter and provides sound fundamentals of talking to I/O devices with both BASIC and assembly routines. I know a lot of beginners have troubles working with optical couplers and Ralph provides a short introduction and explanation of the topic in "Optical Couplers Explained". All next time. BDK.

The Computer Journal / #61
THE NATURE OF THE MAGIC

Wherein we continue the process of transmogrification of a mouse into a barely tamed lion, or it doesn't take a whip to speed a 2.5mhz machine up to 5mhz.

HOUSEKEEPING

Several of you have been in contact with me since the last issue, demonstrating conclusively that an unbelievably large number of CP/M Kaypros are still in use, and doing quite well, thank you. There was, at its peak, an embedded base (that's marketing talk) of some 300,000 CP/M Kaypros, all more or less happily in use, mostly in a business use, (either real or imagined, any computer was too expensive to be a toy back then, so most people found a way to use it as a business "deduction") . I really enjoy hearing from you, and I don't mind answering questions, so don't hesitate to write. Please include your telephone number, because I do travel some and sometimes the telephone is easier.

At one of our periodic gatherings of the faithful, I was asked for clarification on the K-II to K-4 modification. The question asked was "When you solder a wire to one of the pins on a new socket, how do you plug that pin into the existing socket?" The questioner was referring to the 20 pin socket at U-59. The answer, my friends, is blowin' in the wind.... I apologise, folks, I should have been much more specific. The wire is soldered to pin 2 of the 20 pin socket AT THE TOP, WHERE THE PIN COMES OUT OF THE PLASTIC. The same applies to pin 8 of the 16 pin socket.

I will certainly strive to be more careful in the future.

ON WITH THE SHOW

This is one of the smallest, most effective, most fussy modifications ever devised. It isn't complicated, but the parts are small, the spaces even smaller, and it requires a lot of care. Read carefully, take your time and don't push it, and you'll end up with a character-based machine that runs faster than any XT and significantly faster than early ATs. This project, by the way, has been designed so that soldering on the motherboard is only required if U-86 and U-66 are not socketed, and then only to remove the ICs and install sockets.

PRELIMINARIES

Although most of those reading TCJ are already HIGHLY CERTIFIED WIZARDS, there are those of us, that could stand an introduction to, or a good review of, SOLDERING and DESOLDERING. Those HCWs who wish to, can skip-skip ahead to the next section, while the rest of us learn how to avoid a catastrophe.

THE HARDWARE

NECESSARY

Soldering Iron
Solder
NOKORODE soldering paste
A wiping device, sponge, rag, paper towels (you get the idea)
Diagonal wire cutter
Desoldering tool
Isopropyl alcohol
An old toothbrush
NICE TO HAVE
Knife
Forceps
Wire stripper

THE IRON

I use a plain old ordinary 15 watt single heat iron. Soldering stations are nice. Weller makes very good ones, probably the standard of the industry, but they take up a fair amount of room, on the bench, and my iron goes back in the drawer after it's cooled off, and my bench is clear again. My iron came from RadioShack and cost under $10 (a couple of years ago). I also have Weller soldering guns, relics from the early days BUT they are too hot, use AC flowing through the tip, and are too clumsy to boot. Use of a gun on a printed circuit board is an invitation to disaster.

TINNING THE TIP

The first time you heat the iron up, and subsequently whenever it seems to need it, the tip must be "tinned", that is, coated with a new clean film of solder. This is necessary to maximize the heat flow from the iron to the joint, and to avoid including dirt in the joint which would increase its resistance intolerably. It also makes soldering infinitely easier. Here's the most painless way to do it. You'll need the solder, the NOKORODE, and a damp wiping device, I prefer a damp rag, an old diaper works really well. While the iron is heating, open the NOKORODE. It's just a small can of rosin flux. When the iron is hot, plunge the tip into the can of flux, there will be
interesting sounds, and the flux will partially melt. Then coat the tip with solder liberally, and wipe off the excess with the damp rag (more interesting sounds). You should now have a bright shiny tip that will transfer heat efficiently. If the tip wasn’t hot enough when you plunged it into the flux, you’ll just have to wait a little longer for it to heat sufficiently to melt the solder, but the result will be the same. The reason for damping the rag is 1. to protect your fingers, and 2. to prevent scorch marks on the rag. If you fold the rag enough times, the thickness will protect your fingers, and if you don’t care about scorch marks, you can use the rag dry. I prefer to use a dry rag.

The same procedure (flux and then solder) also works for tinning a wire prior to connection to make soldering easier.

THE DESOLDERING TOOL

My desoldering tool also came from Radio Shack. It is a plunger type with a latch. You depress the plunger against a spring until it latches, as announced by a CLANK, hold the plastic tip against the melted solder, and depress the latch. The spring pushes the plunger out creating a vacuum, sucking the solder off the joint into the tool. There are also other types that are integrated with an iron which reportedly seem to work as well.

SOLDER

The solder should be 60/40 with rosin core flux. The most convenient is about 1mm diameter. It seems to be easier to handle, and the joints come out better. ACID CORE SOLDER IS A NO-NO, IT WILL EAT YOUR COMPUTER !!!

WIRE

There are several types and sizes of wire available. For circuit boards where the wire isn’t going to flex a lot, solid wire will work just fine and is easier to use. My favorite is 30 ga. Wire-Wrap wire.

For those places where the wire will be expected to flex periodically, stranded wire is the best, somewhere between 18ga and 26ga (the “ga” stands for “gauge”).

SURFACE PREP

As in other endeavors, here also “cleanliness is next to ......” Any HCW (HIGHLY CERTIFIED WIZARD) you ask will tell you that it is next to impossible to solder a dirty connection, and that’s where the isopropyl alcohol and the old toothbrush come in, both before and after. If you’re going to the store for a toothbrush instead of to the medicine chest, a DENTURE brush, which is wider and has longer bristles, is ideal. After you clean the joint the alcohol will evaporate, so you won’t have to wipe it off. CAUTION, this isn’t the drinking kind of alcohol, and it isn’t good to breathe either, so work in a well ventilated place !!!

THE SURGERY ITSELF

Collect all your tools (well, not all of them, just the ones for soldering) and parts and lay them out on a clean well lit work surface. Start the iron heating, unwrap a few inches of solder, and find an old circuit board, that you don’t care about, to practice on. Test the temperature of the iron by touching the tip with the end of the solder. The solder should melt on contact. If it doesn’t, wait a bit and try again. When the iron is hot enough, touch the tip to the surface to be soldered, and touch the solder to the intersection where the tip touches the surface. As soon as the solder wicks onto the surface remove the iron and the solder and inspect your work. The solder should be hard already, and should look like the wax around a newly lit candle, you shouldn’t be able to tell where the solder/wire interface is. It should look like one continuous surface. If the solder is just a blob sitting on top of the joint, or looks “frosted”, either the joint wasn’t hot enough or clean enough. This situation is the famous “cold solder joint”, bane of technicians and occurs even in production assembly.

UNSOLDERING

This sounds easier than soldering and indeed is somewhat simpler, but in some situations can require 5 hands and an extra elbow. Touch the hot bright and shiny tip of the soldering iron to the component lead on the component side of the board and when the solder on the other side of the board melts depress the latch on the desoldering tool sucking the solder off the board, and out of the component lead hole. The lead will probably remain attached to one side of the hole, so, on the solder side of the circuit-board use a small screwdriver to push the lead toward the clear part of the hole. You’ll hear a small click as the residual solder parts. You can then remove the lead without damaging the hole or solderpad. Using your junk circuit-board, practice until you are confident of your new skills.

THE FORCEPS

If you have to solder a component lead to a large object, such as a large ground bus, you can protect the component in question by applying a “heat sink”. Clamp the forceps on the component lead close to the component itself until the soldering job is done and cooled.

NOW that you’ve read all this and are thoroughly discouraged, TAKE HEART, it isn’t as difficult as it sounds, its really quite easy and once you’ve seen the conditions, you’ll never forget them. SO, run, don’t walk, down to your local surplus electronics store, or talk to that HCW you know of and ask for a junk printed circuit board ($2-$5 at the surplus electronics store, but an HCW will probably just give you one) and start practicing. Try removing components, running jumpers from pins to pads, and pulling up single pins. You’ll be surprised at how quickly you’ll become proficient.

BACK TO THE MOUSE

Here’s what we’re going to do.

1. remove the ICs at U-66 and U-86
2. install sockets if the ICs were soldered to the motherboard
3. build the speedup components
4. plug the new components in
5. change the Z-80
6. check it out

The Computer Journal / #61
THEORY

The CPU (Z-80) as it comes from the factory, runs based on a clock speed of 2.5mhz. All of the components in the computer with the exception of the original Z-80 and the original monitor ROM are capable of functioning at much higher speeds. A-Z-80B is rated at 6mhz so if we install one of those, we could change the clock speed to 5 mhz with no problems. Those of us that either did the K-II to K-4 modification or started with a K-4 already have a fast enough monitor ROM. Those of us who are speeding up a K-II will have to procure/buy/beg/borrow/create a faster ROM. Both the 2.5mhz and 5mhz signals are available on the pins of U-86, so a little surgery will yield the desired result. Some programs, notably the original "COPY.COM" will not work reliably at 5mhz, so for those folks who have not upgraded to either the Micro Cornucopia Pro-8 or the TurboRom will have to add a toggle switch to allow 2.5mhz operation when necessary.

There was a marginal timing signal for the CAS/MUX (Column Address Strobe/Multiplex) signal in the factory editions of the K-II and K-4. We'll modify U-66 to correct this and even if you don't run at the higher speed, you'll notice greater reliability over all. We'll do this by shifting pins on U-66.

NECESSARY PARTS

2 14 pin component carriers
SPDT toggle switch (optional)
2 14 pin sockets (unless your motherboard is fully socketed)
1 74164 IC (unless your motherboard is fully socketed)
1 74LS29 IC (unless your motherboard is fully socketed)
1 Z-80B

HERE WE GO

If your motherboard is fully socketed, using a small screwdriver, remove U-66, U-86, and the Z-80 and skip the next section.

If your motherboard is NOT fully socketed, desolder U-66 and U-86 remove them, and the Z-80. Insert the new sockets into the motherboard, being careful to line the notches up properly, and solder them in.

The hard (scary) part is done!!! At this point you could plug the ICs back into the appropriate places, and you'd still have the same machine you started the day with. This means, of course, that this entire modification is easily removable, should you wish to return to the stock configuration sometime in the future.

NOW, THE NEW PIECES

This is the fun part. We'll do the CAS/MUX mod first.

THE CAS/MUX MOD U-66

1. Find the IC marked 74164 and cut off pin 5.

The pins are numbered COUNTER-CLOCKWISE from the notch, when viewed from the top, remember?

2. Find one of the 14 pin component carriers, and line up the pins of IC from step 1 with the tops of the pins of the component carrier, bending pins 3 & 4 of the IC so that they line up with the tops of pins 4 & 5 of the component carrier, respectively. Look at the drawing, if you're as easily confused as I am. I personally relate to pictures much more easily than words.

3. Solder the IC pins to the component carrier carefully (this is the fussy part.)

That finishes the CAS/MUX mod, now, on to the clock.

THE CLOCK U-86

You must now make a CHOICE. If you need or want the capability of returning easily and temporarily to 2.5 mhz (ie. no TurboRom or Micro Cornucopia Rom) use the procedure in paragraph B below.

If on the other hand, You do have a TurboRom or Micro Cornucopia Rom, and have absolutely no intention of ever using 2.5 mhz again, use the procedure in paragraph A below. (This is the one I use, Toggle switches are for the faint-hearted.)

A. For real men

1. Find the IC marked 74LS29, and cut off pin 4.

2. Find the other component carrier, and line up the pins of the IC from step 1 with the tops of the pins of the component carrier, bending pin 5 of the IC so

---

**Advertising Rates For The Computer Journal**

<table>
<thead>
<tr>
<th>Size</th>
<th>1 Insertion</th>
<th>2-3 Insertions</th>
<th>4+Insertions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>$400</td>
<td>$360</td>
<td>$320</td>
</tr>
<tr>
<td>1/2 Page</td>
<td>$240</td>
<td>$215</td>
<td>$195</td>
</tr>
<tr>
<td>1/3 Page</td>
<td>$195</td>
<td>$160</td>
<td>$145</td>
</tr>
<tr>
<td>1/4 Page</td>
<td>$160</td>
<td>$120</td>
<td>$100</td>
</tr>
</tbody>
</table>

**SPECIAL ANNIVERSARY ADVERTISING RATES**

For the NEXT THREE ISSUES TCJ Advertising Rates will fall back to 1982/3 AMOUNTS! That is HALF the ABOVE PRICES or get TWO ISSUES for the price of ONE.

34 The Computer Journal / #61
that it lines up with the top of pin 4 of the IC.

3. Carefully solder the IC pins to the tops of the component carrier pins as in the other drawing.

Congratulations, you now have a 5 mhz clock!

B. For the prudent man

1. Find the IC marked 74LS29, and BEND OUT pins 4 & 5.

2. Find the SPDT (single pole, double throw) toggle switch, and solder one end of a 6 inch long piece of 20 ga stranded wire to each contact of the switch. The wire size is not critical nor is the length, but the switch will be mounted either on the front panel, or the back panel, so the wires should be long enough to reach. You should now have a switch with 3 wires soldered to it.

3. Find the other component carrier, and solder the wire from the center (common) contact of the switch, to the top of pin 4 on the component carrier.

4. Solder the other two wires from the switch, one each to pins 4 & 5 of the IC from step 1.

5. Now line up all the other pins of the IC with the tops of the pins of the component carrier, and carefully solder them.

Congratulations, you now have a switchable 2.5/5 mhz clock.

INSTALLATION

If you elected to use the switch, now is the time to remove the motherboard, (if you haven’t already) and drill the hole for the switch, using the same technique that you used to move the reset button. (Issue 57)

In any case, plug the new assemblies into the sockets for U-86 and U-66, plug in the new Z-80B, and reinstall the motherboard.

Finish mounting the switch, if you are using one, and it’s time for checkout. Recheck the alignment of all the notches, they should face left when viewed from the front of the computer. Be careful, those component carriers are sneaky, they have been known to swap ends as they were being plugged in.

At this point, you’re ready for power and the pleasure of a very responsive machine, that can type faster than you can, and really RUNS Ladders.

If you turn it on and just get flashing letters and numbers on the screen, turn it off, and carefully recheck your work, correct the error and try again.

THE FUTURE

We have had good response to our plea for circuit board help, and hope in the near future to be able to do the 4 drive decoder for the TurboRom as well as the Micro Cornucopia Rom. For those of you who have the ‘84 machines with the ‘new’ motherboard and the real time clock that independently decides to change its time if you power down, good news, we’re working on a project for you that fixes that problem cold.

Next Issue, replacing the power supply with a really healthy one.

A LITTLE MAGIC DUST and POOF, I’m gone.
(See Chuck’s ad on back page for address and phone number. BDK )

Alternate Clock (U-86)

Notes:
1. Pins 8-14 and 8-13 Omitted for clarity.
2. Leads to toggle switch should be kept as short as possible.
3. Speed-up MOD requires 1 Clock & 1 CAS/MUX modifications.
MULTIPROCESSING FOR THE IMPOVERISHED

by Brad Rodriguez

INTRODUCTION

"We need hardware articles," says TCJ's Editor and Publisher. So I'm taking a short break from my discussion of Fort kernels, to let you in on the Big Picture -- my Grand Project toward which all these articles are leading: a multiprocessor 6809 system. This month I'll describe the 6809 core; in a future issue I'll describe the multiprocessor bus and arbitration logic.

HISTORY AND DESIGN DECISIONS

Why on earth would anyone build a multiprocessor 6809 system?

Several years ago, when I was a "captive" employee, I developed a Fort kernel for a multiprocessor 68000 system. (I believe this was the first published application of Fort to tightly-coupled multiprocessors [ROD89].) The system employed a VME-bus crate, four 68000 processor boards, and a common RAM board and cost about $10,000. Alas, I left that employer shortly afterward.

I've always wanted to continue that research into multiprocessing Fort, but VME bus products are out of my price range these days. Finally I decided to build my own multiprocessor system, using cheap technology -- preferably with components I could buy from Jameco, JDR, or B.G.Micro.

Two essentials for a multiprocessor CPU are a shared-bus request mechanism, and an indivisible memory operator (for semaphores). The 8086/8088 provides these through its bus LOCK mechanism, but only in the "maximum" mode, which requires a pile of support logic. The 68000's has a TAS (test-and-set) instruction, but 16-bit-wide memory doubles the number of memory devices, and who can find 64-pin DIP sockets? (Forget the 68008 -- only distributors carry them.) I briefly considered the Z8002 -- a new CPU would be fun to learn, and it has versatile multiprocessor support -- but it too uses 16-bit memory, and Z8000's are even harder to find than 68008s. (A local distributor quoted $25 each. Times four? Ouch.)

It finally dawned on me that what I wanted was a dirt cheap, commonly available, 8-bit microprocessor, like a Z80 or 6809. And while designing tentative bus arbitration circuits for the 16-bit micros, I realized that I really didn't need special multiprocessor support from the CPU! I now claim that I can parallel any CPU that:

1. can "stretch" its memory cycle to accommodate slow memory, i.e., has a READY or WAIT input;
2. has some direct memory modify instruction, such as ROR M or INC M.

(Strictly speaking, this second condition can be relaxed . . . but at a substantial cost in performance and in extra support logic.)

I was tempted to design around the Z80, since it's just about the cheapest suitable CPU -- but I'm up to my ears in Z80 systems. The 6809 costs only $1 more, is far superior for Fort work . . . and I had some old 6809 hardware designs lying around, and a Fort assembler and Fort kernel for it.

(I will, however, throw in some Z80 design notes. If someone wants to design and prototype a Z80 multiprocessor system on this model, I'm sure TCJ would be happy to print it!)

THE 6809 UNIPROCESSOR

The journey of a thousand miles begins with one step.
- Lao-Tsze

. . . and the journey to a multiprocessor begins with a single CPU. New CPU boards are hard enough to debug without introducing peculiar memory logic. So, first build a uniprocessor and make it work; then add the multiprocessing extensions.

Figure 1 shows the 6809 "core": CPU and memory. The signal labels follow the Orcad convention, using a backslash instead of an over-bar to indicate logical inversion (e.g., RDA is "read-bar," an active low read strobe).

The power-on reset circuit and the clock oscillator are "external to CPU board." This is for two reasons:

a) in the multiprocessor system, all CPUs can share a common reset and clock, so these don't need to be replicated on each CPU card.
b) the 4 MHz 6809 can share the 3.6864 MHz clock required by the 2681 DUART's baud rate generator, saving one crystal.

If you're just building a single 6809 processor, you can use the 6809's internal oscillator by tying a crystal (4 MHz or less) to the 6809 pins X1 and EX2. A second crystal (3.6864 MHz) can be tied to pins X1 and X2 of the 2681.

Note that the oscillator and reset circuits both use sections of U8, although they require a 74HC04 and 74HCT14, respectively. The oscillator follows the T.I. application note for HCMOS inverters [TEX84], except that I lowered R3 to 10K to get it to oscillate at 3.6864 MHz. (If the circuit oscillates at too low a frequency, R3 is too high.) My original reset circuit used the 74HC04, but every variation I tried gave noisy reset pulses. Finally I was forced to do it "properly," and I plugged in a 74HCT14 Schmitt trigger inverter. Fortunately, the oscillator still seemed to work with the 74HCT14, although by all rights it shouldn't have. Probably you should play safe and use two different parts.

The full multiprocessor CPU design puts 4 gate loads on the oscillator. For eight CPUs this is 32 loads, which even in CMOS is too much for comfort. So, the oscillator input to each board is buffered by U7B. U7C (on figure 2) is the complimentary clock for the 2681. Note that U7 must be an HCMOS part, to output a sufficient "high" level for the 2681's clock input. You can use LSSTL if you add 470 ohm pullups to +5 to the 2681's clock inputs X1 and X2.

You might be tempted to use the 6809E for this project. Don't! The 6809E, an external clock version of the 6809, requires a quadrature clock input, and does not include the MRDY logic. Use a plain 6809, or the higher-speed versions 68A09 or 68B09.

The 6809 is easy to interface to memory. Every CPU clock cycle is a memory cycle, and the E clock output is also the active-high data strobe. R/W is high for read cycles and low for write cycles. When the processor is doing internal operations, address FFFF is output on the address bus and R/W is high, causing memory reads to location FFFF. (This must be a valid memory location, since it is the low byte of the 6809 reset vector.)

I find separate RD\ and WR\ strobes more useful for most memory and I/O devices. These can be generated from R/W\ and E with an inverter and two NAND gates; but I prefer to use half of a 74HCT139 (USB). It's easier to wirewrap, and I had a spare '139 section on the board anyway.

Figure 3 shows the timing diagram for the CPU [MOTR83]. The CPU runs at 1/4 of the oscillator frequency, and outputs clock signals E and Q in phase quadrature (Q leads E by 90 degrees). With a 3.6864 MHz clock, the data strobe E is high for 540 nsec, plenty for even slow memories. Note that the address is valid at the rising edge of Q. You can generate a wider data strobe by using E+Q (the logical OR of E and Q); this may be

---

MEMORY DECODING FOR PROGRAMMERS

You say your most intimate contact with a CPU has been assembly language programming? Or you've been a Z80 man all your life, and don't know what the E signal is? Read on...

Most computers have separate CPU and memory chips, so some kind of control signals are needed to transfer data between them. If you've programmed in assembly, you're familiar with the ADDRESS. The CPU always sends this to the memory, to tell it which memory location to access. On Z80s and 6809s there is one physical line -- a pin on the chip, and a wire on the board -- for each of the 16 address bits. These are called A0 through A15. Most CPUs call the least-significant bit A0, and the most-significant A15. When an address bit is '1', +5 volts is output on the corresponding pin; when it is '0', 0 volts is output. These are "TTL levels," and "active high".

(Some other processors "multiplex" the address, but don't worry about this now -- it would just confuse things. Also, the "high" voltage can be anything from +2.4 to +5 volts, and the "low" voltage anything from 0.0 to +0.8 volts, but we usually say "+5" and "0" for brevity.)

You're also familiar with the DATA. Unlike the address, this can go two ways. If you're reading the memory -- as in a Load instruction -- the data lines are output by the memory chip, and input by the CPU. If you're writing the memory -- as in a Store instruction -- the data lines are output by the CPU, and input by the memory. That's why these pins on the chips are called "bidirectional." Like before, the eight data lines are numbered D0 to D7, with D0 usually the least significant bit; and like before, +5 volts is put on the line for a binary '1' and 0 volts for a binary '0'.

From this you can see that two important pieces of information are needed to transfer data, namely:

(a) which way is the data going, to or from the CPU?, and
(b) when should the transfer take place?

This information is always provided (output) by the CPU. There are two common schemes for this, which I call the

---

Continues at end of main article

Coming Soon
Part II of 6809 CPU project
Part III of Moving Forth
desirable if you use a faster 6809.

Part of the design problem is deciding how much RAM and ROM to include. I’ve divided the 64K address range as follows:

- 0000-1FFF on-board RAM #1, 8K
- 2000-3FFF on-board RAM #2, 8K
- 4000-5FFF external (multiprocessor) bus, 8K
- 6000-7FFF on-board I/O
- 8000-FFFF on-board PROM, 32K

I contemplated using an 8K PROM, which is enough for a Forth kernel. But I have extra 27256’s in my parts box, and I thought that, as I develop the multiprocessor extensions to Forth, I’d want to put them in PROM. Since the 6809 holds its reset and interrupt vectors in high memory (addresses FFF0 to FFFF hex), the 27256 PROM, U2, is mapped into the high 32K of the address space. Whenever address line A15 is high, U6A asserts CSROM low.

The remaining 32K of address space is divided into four 8K regions by U5A. Whenever address line A15 is low, U5A will assert one of four strobes, depending on A14 and A13. I have a surplus of 6264 8K RAM chips left over from a previous project, so I designed the board to hold two of these (U3 and U4).

In any multiprocessor system it is important to have “private” memory for each processor, and “public” memory for all processors to share. An 8K region is mapped to this “public” memory. When -- and only when -- the CPU accesses any address in the range 4000 to 5FFF, it requests the use of the external (shared) memory bus. (More on this in the next article.) I’ve arranged the address map so the on-board RAM and the external RAM form a contiguous 24K region.

For simplicity of decoding, I’ve allotted an 8K region to the 2681 DUART. This is extravagant, since the 2681 requires only 16 memory locations.

Figure 2 shows how to use a supplementary address decoder to divide this 8K space into four 2K regions; this allows three I/O devices in addition to the 2681. Note that some of the decoding done by U6A is duplicated in the 74HCT138, U10. This illustrates an important design principle. You might generate eight chip selects instead of four, by routing CS6 to an active-low gate input of U10, and then decoding A10 through A12. But this would cascade the propagation delays of two decoders! After many CPU projects, I have concluded that address decoding should be done in parallel, preferably with no more than one stage of decoder delay to any address strobe.

U5, U6, and U10 can be 74LS parts if desired. Note also the temporary connections of the 6809’s FIRQ and MRDY to VCC. These inputs will be used later, for the full multiprocessor CPU.

Figure 2 shows the serial I/O, built around a Signetics SCN2681 DUART (Dual UART). This versatile chip has two UARTs with programmable baud rate generators, a counter/timer, a 7-bit input port, and an 8-bit output port. (This and the Z8530 are my two favorite serial chips.)

The 2681 is unusual in that it requires an active-high RESET signal, provided by U6D. Incidentally, I’ve used two sections of a quad NAND and two sections of a quad NOR to provide four inverters because the full multiprocessor design needs NAND and NOR gates. You see the “leftover” gates here as inverters.

The interrupt-request output of the 2681, INTR, is tied to the 6809’s IRQ input. Since this is an open-drain output (the MOS equivalent of open-collector), it needs a 10K pullup to +5.

The “A” port of the 2681 is an RS-232 port, using a 1488 and a 1489 for drivers and receivers. Since I have extra 9-pin D connectors, plus an abundance of adapters, cables, null modems, and whatnot for the IBM PC/AT, I’ve made this serial port look exactly like a PC/AT serial port. This means I have to use a null modem or other reversing cable to connect to my PC.

The 1489 has one receiver too few, so the RI (Ring Indicator) signal is left unimplemented. The 1488 has one extra driver, so I use it to drive an LED. I always find a way to put an LED on a new CPU board; it is invaluable for debugging.

Port “B” of the 2681 is a bidirectional RS-485 port. I’m experimenting with two-wire RS-485 local area networks, and the 2681 supports the 9-bit (address mark) format used by many modern microcontrollers. One DUART output line is used as a direction control.

The high three bits of the output port are reserved for a page select on the multiprocessor bus (more on this in the next article). One output is unused. Note that the four unused inputs are tied to ground; this is most important for a MOS device like the 2681! I learned the hard way: floating MOS inputs pick up noise and cause rapid transitions, which in turn puts noise on the power bus and sends the device’s power dissipation through the roof.

Another warning from the 2681 School of Hard Knocks: do not access the 2681 registers which are “Reserved!” (Read Registers 02, 0A, and 0C, and Write Register 0C.) Even reading a reserved register will throw the 2681 into a self-test mode that ignores the RESET input. Once you’ve done this, only powering off and back on will restore normal operation.

The power connections and the bypass capacitors on each chip aren’t shown. I use a three-voltage (+5, +12, and -12) supply so that I can use the 1488s and 1489s. I could have used MAX232’s and a single +5 supply, but when you multiply the cost by eight CPUs, the three-voltage supply is the better
bargain.

TEST SOFTWARE

Listing 1 is the test program I have used to exercise this board. I include it here mainly to illustrate the initialization of the 2681; refer to the Signetics manual [SIG86] for more details. This code is written for PseudoCorp’s freeware PseudoSam Level I assembler, A689, which I obtained from the Real-time Control Forth Board (RCFB) BBS in Denver, Colorado, phone (303) 728-0364. PseudoCorp also advertises their professional assemblers in TCI.

A full Forth kernel for this board, and a metacompiler that runs under F83, will be appearing soon on the Forth Interest Group Roundtable (FORTH) on GENie.

SOME IMPOVERISHED OBSERVATIONS

I call this system the ScroungeMaster I, because I’m particularly proud of how cheaply I’ve obtained the components. I found 44-pin .156” edge connectors at a Toronto surplus shop, Vectorboards to match, and wirewrap IC sockets, came from the Trenton Computer Festival. The 8K RAMs and 27256s were leftovers from old projects, and almost all of the glue logic came from closeout sales. The 6809s were $2.50 apiece from B.G.Micro. Ironically, the most expensive part was the 2681 DUART, for which I had to pay Jameco the full retail price of $6.95 each! (Maybe I should have redesigned to use up all those 28530s I have lying around . . . )

REFERENCES


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 brad@genie.geis.com on the Internet, a.k.a. B.RODRIGUEZ@GENIE. The telecommunicationally disadvantaged can write to him at Box 77, McMaster University, 1280 Main St. West, Hamilton, Ontario L8S 1C0 Canada.

MEMORY DECODING FOR PROGRAMMERS continued.

“Motorola” and “Intel” approaches. They are illustrated here:

<table>
<thead>
<tr>
<th>6809 CPU</th>
<th>Z80 CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>(“Motorola”)</td>
<td>(“Intel”)</td>
</tr>
<tr>
<td>A0-A15 ⏞</td>
<td>A0-A15</td>
</tr>
<tr>
<td>D0-D7 ⏞</td>
<td>D0-D7</td>
</tr>
<tr>
<td>RWR</td>
<td>RDA</td>
</tr>
<tr>
<td>E</td>
<td>MREQ</td>
</tr>
</tbody>
</table>

The Motorola scheme uses separate control signals for (a) and (b). The signal R/W is +5 volts for a CPU read, i.e., data transfer from memory to the CPU (Load). It is 0 volts for a CPU write, i.e., data transfer from CPU to memory (Store). The name is mnemonic, and should be read as “Read / Write-Bar”. The backslash is an Orcad convention -- this is as close as the ASCII character set can come to an over-bar, the usual symbol for logical inversion. This means that when the signal is high, it’s a read, and when it’s low, it’s a write.

The signal E is the “data strobe.” When E goes active (high), it means that the address has been output on the A0-A15 lines, and the data transfer can take place. If it’s a CPU write, it also means that the data has been output on D0-D7. If it’s a CPU read, it means that the memory can now put data on the data lines. When E goes inactive (low), the data transfer is over. The CPU stops outputting address and (on a write cycle) data.

Note that the memory has no control over the timing: when on a read cycle, the memory had better output its data before E goes inactive!

The Intel (and Zilog) approach also uses two control signals, but they carry the information in a combined form. One signal, RDA (“Read-Bar”), simultaneously informs the memory that it is a CPU read cycle, and that it’s time to transfer data. The cycle begins when RD\* goes low, and ends when RD\* goes high. This is called an “active-low” signal, and is why it’s labelled Read-Bar. The second signal is WR\* (“Write-Bar”), which simultaneously says that it’s a CPU write cycle, and that it’s time to transfer data. This too is an active-low signal.

Two other signals appear on some Intel and Zilog chips, such as the Z80. This is because these chips distinguish between memory devices and I/O devices. When MREQ (“Memory Request”) is low, it means that the read or write cycle is to a memory. When IORQ is low, it means that the read or write cycle is to an I/O device (such as a UART or disk controller). Obviously, these two signals are never both active (low) at the same time. Electrically, memories and I/O devices look the same to the CPU, so Motorola treats them both as though they were memory.

A final word about decoding: if there are several memory (or I/O) chips, only one can transfer data at a time. This because they all share the same data lines (the data “bus”). So, you need some way to select which chip is to participate in any data transfer. This is usually done by recognizing, or “decoding”, certain combinations of the high address bits. For example,
this article’s 6809 board uses extra logic devices to decode the following combinations of the three highest address bits:

<table>
<thead>
<tr>
<th>A15</th>
<th>A14</th>
<th>A13</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

"X" indicates that either a ‘0’ (low) or a ‘1’ (high) will be accepted. These are called “don’t-cares”. To facilitate this decoding, most memory and I/O chips have a “Chip Select” (CS) or “Chip Enable” (CE) input. Usually these are active low: to access any given chip, its Chip Select must be held low. When the chip select input is held high, that chip completely ignores the address, data, and control signals.

WHAT ARE THOSE OTHER FUNNY SIGNALS? So you understand address, data, R/W, and and what are all those other signals on the 6809 CPU?

X1 and EX2 are where you connect a crystal, to provide the frequency reference for the 6809’s clock oscillator. The 6809 can also accept a clock signal from an external source: put the clock signal on EX2, and tie X1 to ground. The “E” in “EX2” reminds you that it is the pin to use for External clock.

E is (you may recall) the active-high data strobe output by the CPU. It is also the internal CPU clock, which runs continuously at one-fourth of the crystal (or external oscillator) frequency. The design of the 6809 is such that every clock cycle is a memory reference cycle, so this one signal can do double duty as clock output and data strobe.

Q is the “quadrature” clock output. This is another internal clock signal of the 6809. Q is the same frequency as E, but 90 degrees out of phase. Q leads E by 90 degrees. This is sometimes useful for odd timing logic, such as that required by dynamic RAM. Q can also be used to “stretch” the memory cycle of the 6809. Often it can be simply ignored.

MRDY is the “Memory Ready” input. Elsewhere I said that the timing of the memory cycle was under control of the CPU. This signal gives memory and I/O devices a way to inform the CPU that they need a longer cycle. When MRDY is held low (“Not Ready”), the CPU will keep outputting the address and (if appropriate) data, and will hold the E signal (data strobe) high. When the memory device has had enough time, it pulls MRDY high (“Ready”), allowing the CPU to complete the memory cycle. On the Z80, the equivalent signal is called WAIT.

The RESET\ input, when pulled low, stops and completely resets the CPU. Any information in the CPU is lost; but the memory usually remains valid. When RESET\ is pulled high, the CPU starts executing machine code at an address which is stored in memory locations FFFE:FFFF. Obviously this should be PROM!

IRQ\ is the Interrupt Request input. When this is pulled low, the CPU starts executing machine code at an address which is stored in memory locations FFF8:FFFF. Unlike RESET\, IRQ\ saves the state of the program which had been executing. When the interrupt code is finished, it can resume the previous program exactly where it left off. Software can “turn off” this input.

NMI\ is the Non-Maskable Interrupt input, which works the same as IRQ\ except that a) it fetches a code address from FFFC:FFFFD, and b) there is no way for the program to disable this input.

FIRQ\ is the Fast Interrupt Request input, which works the same as IRQ\ except that a) it fetches a code address from FF6:FFFF7, and b) it doesn’t save all the CPU state. The “unsaved” part of the CPU state can be saved by the software, if desired. A note for the technical purists: IRQ\ and FIRQ\ are “level-sensitive,” which means they cause an interrupt as long as the input is low. NMI\ is “edge-sensitive,” so it only causes an interrupt when it makes a transition from high to low.

The HALT\ input, when pulled low, stops the CPU. But unlike RESET\, no data is lost. When HALT\ is pulled back high, execution of the program will continue where it left off. Z80 fans should note that the Z80 has a HALT output, which performs a different function. Note: The schematic shows RESET\, IRQ\, NMI\, FIRQ\, and HALT\ as active-low, not with an over-bar, but with a little inversion circle along the input to the chip.

The DMA/BREQ\ input stands for “Direct Memory Access/Bus Request.” This provides a way for some other device to read and write the CPU’s memory chips. When DMA/BREQ\ is pulled low, the CPU suspends its program and stops outputting R/W, address, and data. Then some other device -- usually a “DMA controller” chip -- can use the address and data busses and the R/W line to transfer data to and from memory, independently of the CPU. (The DMA controller is expected to use E -- which the CPU continues to output -- as its data strobe.) When large blocks of data need to be transferred, Direct Memory Access is usually faster than software. Many disk controllers use this technique. On the Z80, this signal is called BUSREQ\.

BA and BS stand for “Bus Available” and “Bus Status,” and are output by the CPU to indicate one of the following four conditions:

<table>
<thead>
<tr>
<th>BA</th>
<th>BS</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>normal (running)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>interrupt or reset acknowledge cycle</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>synchronize (wait for interrupt)</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>halted, or Bus Request (DMA) granted</td>
</tr>
</tbody>
</table>

The most common use of BA and BS is to provide the Interrupt Acknowledge and DMA Grant signals required by some I/O chips. The Z80 uses an entirely different scheme to indicate processor status, so there’s no easy comparison.
; Simple test program for The Computer Journal's 6809 Uniprocessor
; B. Rodriguez 11 March 1983

.org $1900
.equ duart, $6000 ; base address of 2681 DUART
.equ ndty, 4
.equ ndtyl, 1

; 2681 Initialization Table. Each word in this table contains
; register-number:contents in the hi lo bytes, respectively.

; inibit:
; dw $0230 ; Command Register A: reset nx, disable nx & bx
; dw $0240 ; Command Register A: reset error status
; dw $0210 ; Command Register A: reset MR pointer
; dw $0013 ; Mode Register A(1): 8 bits, no parity
; dw $0007 ; Mode Register A(2): 1 stop, RTS & CTS manual
; dw $010b ; Clock Select A: bx & nx 9600 baud
; dw $0105 ; Command Register A: enable nx & bx
; dw $02a2 ; Command Register B: reset nx, disable nx & bx
; dw $0330 ; Command Register B: reset bx
; dw $0e40 ; Command Register B: reset error status
; dw $0a10 ; Command Register B: reset MR pointer
; dw $0813 ; Mode Register B(1): 8 bits, no parity
; dw $0a07 ; Mode Register B(2): 1 stop, RTS & CTS manual
; dw $0e6b ; Clock Select B: bx & nx 9600 baud
; dw $0e05 ; Command Register B: enable nx & bx
; dw $0400 ; Aux Control Register: counter mode, xtal16
; dw $062d ; Counter Upper, and
; dw $0700 ; Counter Lower: 2000 hex = 50,000 nsec
; dw $0000 ; Output Port Configuration: all manual
; dw $0e03 ; Set Output Bits: CO and OPI low
; dw $0500 ; Interrupt Mask Register: all disabled

endbit:

; The test program enters here on a reset.
; This program doesn't use stacks, so the stack pointer doesn't
; need to be initialized.

entry:
; Initialize DUART from the table above

ldy #nibbl
ldx #duart
lola $y++, ; fetch a b from table
stb a, x ; store b at duart+a
cmpy #nibbl
bne loop

loop:
ldx $y++, ; Simple memory test, to check locations 2000 to 3FFF hex.
lea $w+2, ; character means good
lea $w+2000, ; starting address
mtest:
ldx $w+1 ; rotate this through all bit positions
bitest:
stb x
cmpx $w
bne bad
asl
bne bitest
bra good

bad:
and $w+5 ; error encountered: change '1' to '0'
good:
lea $w+1 , ; next address
cmpx $w+4000
bne mtest

; Output the character in the A register, over serial port A
loop:
ldx duart+1
and $w+03y
beq loop
sta duart+3

; Print all remaining ASCII characters up through 7F hex.
inca
lblr loop

; Now wait for a received character, and echo it and all following characters.
loop:
ldx duart+1
and $w+3y
beq loop
sta duart+3
bra loop

; The reset vector for the 6809 is located at address FFFE hex.
.org $ffe
.dw entry
.end

---

FIGURE 3. 6809 Memory Timing (4 MHz osc)

---

The Computer Journal / #61 41
The Computer Journal

Back Issues
Sales limited to supplies in stock.

Volume Number 1:
Issues 1 to 9
- General Interfacing and Modern transfers
- Floppy disk formats, Print spooler
- Adding 8087 Math Chip, Fiber optics
- S-100 Hi-RES graphics
- Controlling DC motors, Multi-user systems
- VIC-20 EPROM Programmer, CPM 3.0
- CPM user functions and integration.

Volume Number 2:
- Issues 10 to 19
  - Fortran tutorial and Write Your Own
  - 68008 CPU for S-100
  - RNM vs CPM, BIOS Enhancements
  - Poor Man's Distributed Processing
  - Controlling Apple Stepper Motors
  - Facsimile Pictures on a Micro
  - Memory Mapped I/O on a Z801

Issue Number 21:
- Designing an 8035 SBC
  - Using Apple Graphics from CPM Turbo
  - Pascal Controls Apple Graphics
  - Sorting & Other Strange Tails
  - Build an S-100 Floppy Disk Controller
  - W2797 Controller for CPM 86K

Issue Number 22:
- Extending Turbo Pascal
  - Customize with Procedures & Functions
  - Unearthing The Arcane Art
  - Analog Data Acquisition & Control
  - Connecting Your Computer to the Real World
  - Programming the 8035 SBC

Issue Number 23:
- NEW-DOS Write Your Own Operating System
  - Variability in the BBS C Standard Library
  - The SSCI Interface, Introductory Column
  - Using BBS Files, The Ampro Little Board Column

Issue Number 24:
- C Column: Flow Control & Program Structure
  - The Z Column: Getting Started with Direct & User Areas
  - The SSCI Interface Introduction to SCCI
  - NEW-DOS: The Console Command Processor
  - Editing the CPM Operating System
  - INDEXER Turbo Pascal Program to Create an Index
  - The Ampro Little Board Column

Issue Number 25:
- Selecting & Building a System
  - The SSCI Interface, SSCI Command Protocol
  - Introduction to Assembly Code for CPM
  - The C Column: Software Test Files
  - Ampro 168 Column: Installing MS-DOS

Issue Number 26:
- Bus Systems, Selecting a System Bus
  - Using the SB180 Real Time Clock
  - Using Bipolar Software, the SSCI Adapter
  - Ampro Computers
  - NEW-DOS: The CCP Commands

Issue Number 27:
- 68000 TinyGiant: Hawthorne's Low Cost 16-bit SBC and Operating System
  - The Art of Source Code Generation
  - Disassembling 2.80 Software

Feedback Control System Analysis: Using Root Locus Analysis & Feedback Loop Compensation
- The C Column: A Graphics Primitive Package
- The Hitchhiker: HD4410 New Life for 8-bit Systems
- The Z-Column: Command Line Generators and Alased
- A Tutorial Program in Forth: Writing a Forth Tutor in Forth

Issue Number 28:
- Build Your Own BBS
  - Build an X.COM Converter for the Ampro Little Board
  - HD4410: Setting the Wait States & RAM Refresh using PRT & DMA
  - Use a ROM for Real Time Timers
  - Open Letter to STD Bus Manufacturers
  - Patcher Turbo Pascal
  - Compilation of a Language for Machine Control

Issue Number 29:
- Software Filter Design
  - Modifying a 1 meg RAM Disk to Ampro Little Board, Part 1
  - Using the Hitchhiker Embroider, Embroidered Program Design
  - HD6800: Why use a new OS and an 80007? Detecting the 8087 Math Chip
  - Floppy Disk Drive Track Information

Issue Number 30:
- Double Density Floppy Controller
  - ZCRP3 IOP for the Ampro Little Board
  - 3200 Hackers Language
  - MS-DOS: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 2

Issue Number 31:
- New-Preemptive Multitasking
  - Software Timers for the 68000
  - Li叩Pe Internet
  - ZCRP3 Corner
  - The Ampro Little Board Column

Issue Number 32:
- Using BASIC for Generalized I/O
  - Communicating with Floppy Disks: Disk I/O and their Variations
  - XBDOS: A Replacement BIOS for the SB180
  - K-DOS ONE, and the SAGE Demystifying Operating Systems
  - Remote Designing a Remote System Program

Issue Number 33:
- Language Development: Automatic Generation of Parsers for Interactive Systems
  - Designing Operating Systems: A ROM based OS for the Z80
  - Advanced CPM: Boosting Performance
  - Systematic Elimination of MS-DOS Files
  - Selecting Root Directories & an In-Depth Look at the FCB
  - Advanced CPM: Batch Processing
  - Advanced CPM: How to Write Self Relocating Code

Data File Conversion: Writing a Filter to Convert Foreign File Formats
- Advanced CPM: Z80PLUS & How to Write Self Relocating Code
- Database: The First in a Series on Data Bases and Information Processing
- SCCI for the S-100 Bus: Another Example of S-100 Virality
- A Mouse on Any Hardware Implementing the Mouse on a Z80 System
- Systematic Elimination of MS-DOS Files

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
  - ZCRP3: Relocatable code, PRL files, ZCRP34, and Type 4 programs
  - New Microcontrollers Have Shirts, Chips with BASIC or ROM in ROM are easy to program
  - Advanced CPM Operating systems extensions to BDOS and BIOS, RSX's for CPI M 2.2
  - Macintosh Data File Conversion in Turbo Pascal

Issue Number 35:
- All This & Module 2-A: A Pascal-like alternative with scope and parameter passing
  - A Short Course in Source Code Generation
  - Desassembling 0380 software to produce modifiable assembler source code

Issue Number 36:
- Information Engineering: Introduction
  - A list of reference books
  - Temperature Measurement & Control
  - Agricultural computer application
  - ZCRP3 Corner: Z-Nodes, Z-Plan, Amstrad computer, and ZFLE
  - Real Computing: MS-DOS/2032 hardware for experimenters, CPUs in series, software options

Issue Number 37:
- C Pointers, Arrays & Structures Made Easier: Part 1, Pointers
  - ZCRP3 Corner: Z-Nodes, patching for NCOM, ZFLE
  - Information Engineering: Basic Concepts, fields, definition, client worksheets,
  - 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 NS3200X
  - ZDOS: Anatomy of an Operating System Part 2

Issue Number 38:
- Programming for Performance Assembly Language techniques
- The Z-System Corner: System enhancements with NCOM
- Generating LaserJet Fonts: A review of Digi-Fonts
- Advanced CPM: Making old programs Z-System aware

Issue Number 39:
- Programming the LaserJet: Using the escape code
  - Beginning Forward Column: Introduction
  - Advanced Forward Column: Variant Records and Modules
  - LINKPRI: Generating the bit maps for PRL files from a REL file
  - WordTech's dXKL: Writing your own custom designed business program
  - Advanced CPM: ZEX 5.0+the machine and the language
  - Programming in Assembly Language techniques
  - Programming Input/Output With C: Keyboard and Printer functions
  - The Z-System Corner: Remote access systems and BIOS C
  - Real Computing: The NS3200X

Issue Number 40:
- Forth Column: ADTs. Object Oriented Concepts
  - Improving the Ampro LB: Overcoming the 8864 hard drive limit
  - How to add Data Structures in Forth
  - Advanced CPM: CPM is hacker's haven
  - Z-System Command Scheduler
  - The Z-System Corner: Extended Multiple Command Line, and aliases

Issue Number 41:
- Programming disk and printer functions with C
  - LINKPRI: Making RSXes easy
  - SCOOPY: Copying a series of unrelated files

Issue Number 42:
- Dynamic Memory Allocation: Allocating memory at runtime with examples in Forth
  - Using BZX with NCOM and C
  - The MS-DOS Screen Character Attributes

Issue Number 43:
- Forth Column: Lists and object oriented Forth
  - The Z-System Corner Genie, BQZ 2 and Z-System Fundamentals
  - 68020 Embedded Controller Application
  - Advanced CPM: PIPerfect Writer and using BIOS with REL files
  - Real Computing: The NS3200X

The Computer Journal / #61

44
The Computer Journal

Back Issues

- Z-System Corner, Programming for Compatibility
- Z Best Software
- Real Computing, X10 Revisited
- PMATE/ZMATE Macros
- Controlling Home Heating & Lighting, Pt. 3
- The CPM80: A High Performance Single-Board Computer
- The Computer Corner

Issue Number 53:
- The CPI/200
- Local Area Networks
- Am Arbitrary Waveform Generator
- Real Computing
- Zed Fest '81
- Z-System Corner
- Getting Started in Assembly Language
- The Z8000/200
- Z-BEST Software
- The Computer Corner

Issue Number 54:
- Z-System Corner
- B.Y.O. Assembler
- Local Area Networks
- Advanced CPM
- ZCPI on a 16-Bit Intel Platform
- Real Computing
- Interrupts and the Z80
- 6 MHz on a Ampro
- Hardware Heaven
- What Zilog never told you about the Super6
- An Arbitrary Waveform Generator
- The Development of T1-DOCS
- The Computer Corner

Issue Number 55:
- Fuzzology 101
- The Cyclic Redundancy Check in Forth
- The Internetwork Protocol (IP)
- Z-System Corner
- Hardware Heaven
- Remapping Disk Drives through the Virtual BIOS
- The Bumbling Mathematician
- YASEM
- Z-BEST Software
- The Computer Corner

Issue Number 56:
- TCG - 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

The Computer Journal

SPECIAL DISCOUNT
15% on cost of Back Issues when buying from 1 to Current Issue.
10% on cost of Back Issues when buying 20 or more issues.
Maximum Cost for shipping is $25.00 for U.S.A. and $45.00 for all other Countries.
| K1 | MODEM PROGRAMS | K18 | SYSTEM DIAGNOSTICS |
| K2 | CP/M UTILITIES | K19 | PROWRITER GRAPHICS |
| K3 | GAMES | K20 | MICROSHERE'S COLOR GRAPHICS BOARD |
| K4 | ADVENTURE | K21 | SBASIC & SCREEN DUMP |
| K5 | MX80/GEM 10X GRAPHICS | K22 | ZCPR |
| K6 | TEXT UTILITIES | K23 | FAST TERMINAL & RCPM UTILITIES |
| K7 | SMALL C VER 2 | K24 | KEYBOARD TRANSLATOR & MBASIC GAMES |
| K8 | SOURCE OF SMALL C | K25 | Z80 MACRO ASSEMBLER |
| K9 | GENERAL UTILITIES | K26 | EPROM PROGRAMMER/TOOLS |
| K10 | Z80 AND LINKING ASSEM | K27 | TYPING TUTORIAL |
| K11 | CHECKBOOK PROGRAM & LIBRARY UTILITIES | K28 | MODEM 730 SOURCE |
| K12 | KAYPRO FORTH | K29 | TURBO PASCAL GAMES I |
| K13 | SOURCE OF FIG-FORTH | K30 | TURBO PASCAL GAMES II |
| K14 | SMARTMODEM PROGRAMS | K31 | TURBO BULLETIN BOARD |
| K15 | HARD DISK UTILITIES | K32 | FORTH-83 |
| K16 | PASCAL COMPILER | K33 | UTILITIES |
| K17 | Z80 TOOLS | K34 | GAMES |
| K35 | SMALL C VER 2.1 |
| K36 | SMALL C LIBRARY |
| K37 | UTILITIES PRIMER |
| K38 | PASCAL RUNOFF WINNERS FIRST - THIRD |
| K39 | PASCAL RUNOFF WINNERS FORTH & FIFTH |
| K40 | PASCAL RUNOFF WINNERS SIXTH PLACE |
| K41 | EXPRESS 1.01 TEXT EDIT |
| K42 | PASCAL RUNOFF-GRAPHICS |
| K43 | PASCAL RUNOFF-GAMES |
| K44 | PASCAL RUNOFF-PRINTERS |
| K45 | PASCAL RUNOFF-UTILITIES |
| K46 | PASCAL RUNOFF-TURBO UTILITY |
| K47 | 256K RAM SOFTWARE |
| K48 | C CONTEST WINNERS I |
| K49 | C CONTEST WINNERS II |

**Micro C Disks are $6.00 each plus shipping costs.**

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

Added these costs $1.00

Shipping costs are for GROUPS of 1 to 3 disks.
Computer Corner
By Bill Kibler

This last month has been pretty busy, but I have a few good words for you to ponder.

CP/M & HARD DRIVES

If you need a new hard drive, or have a bad hard drive on your old system, I recently talked to some people that Charles Stafford recommended and might help you out. RME in Northern California specializes in fixing and exchanging old drives. When I talked to Mary Harrison (at 916 939-7500) she indicated that their engineers can even (sometimes) match new components to old systems. They do data recovery but 90% of the time the hard disk has failed and so they will simply exchange the drives for a fee. So if you need hard drive help for an old system (they also support new systems as well) try RME at 5075 Hellsdale Circle, Eldorado Hills, CA 95762.

Now let me remind everyone one that I am willing to give you a plug if you support older systems. Just send me a company flyer or fact sheet and I will gladly put the information somewhere in TCJ. Finding help for problems is what TCJ is for. Finding information can in fact be simply knowing what the official name is.

IDE or "AT A"?

As you know I have been trying to find out more on the IDE drive specification. Finally caught up with one of Maxtor’s engineers (Lawrence Lamers) who informed me that my problem was simply having the wrong name. IDE is basically a sales name, when “AT Attachment Interface for Disk Drives” interface is more correct. “AT A” is actually not correct either, X3T9.2 is the true ANSI number for the soon to be standard. Because of the ANSI regulations, I can’t print the whole standard, but I can talk about it and review it (later).

Should you want your own copy, you can download it just as I did from the SCSI bulletin board in Colorado. There number is 719-574-0424 and they have various versions (print formats) of the standard to be chosen from. As I understand it, the interface appears as a WD1010 hard disk controller chip. Now I will be checking this out in more detail later. Rick Rodman is also looking over the specifications with an article in mind. For the Timmann Reh IDE interface project, he is preparing the next installment and hopes to have it done for issue #62, time permitting.

I am also working on getting some older S-100 hard drive interface information and schematics to feature in the centerfold later. I’ll run the hard disk BIOS as well so you can see how it was done for those classic systems. All in all our hard disk interfacing looks like the next year’s focus point for TCJ.

FOCUSING ON USER ITEMS

I have been considering some minor problems that have surfaced lately. In getting Brad’s article I was lucky that both of us use Orcad. Terry Hazen sent me a hand drawn schematic for his SCSI EPROM Burner and his article will be in next issue after I transfer it to Orcad. The problem is getting high quality drawings to print, and especially a CAD package that all our readers can get and use. What I would like to do is standardize on some Public Domain or near free products that I can ship or give to readers and writers. This would be helpful as well if we get into shipping article information on disk.

The problem is finding the CAD package and checking it out. Tim McDonough found one and has sent a copy to Terry Hazen. Terry was too busy to try it, but indicated he would be checking it out later. I found some public domain versions on a few BBS’s but haven’t tried them yet. If you have a favorite one, or know of one that is perfect for our use, message me or send me a copy. It needs to be simple, do schematics, output on a number of devices (dot printers, laser jets, and HP Plotters). If it can layout PC boards, so much the better, but not really that necessary.

The ideal CAD program would be done in “C” or Forth code making it portable to other small systems. A PC clone only product would be acceptable if not working on CP/M is it’s only fault. It must however be simple enough that you need not be a super hardware person or computer guru to use. I would like it to be portable to other systems so it can fit into my one option for all.

ALL FOR ONE?

On the same idea of an universal CAD program for TCJ users, I would also like to start our readers considering a universal small systems standard language and platform. When TCJ gets articles with code, one major problem is making it work with all the different systems available. Supporting 6800 through 68000 systems, as well as 8080/Z80 and 80xxx versions is rather a big problem.

I have been pushing Forth as it will meet the bill. The only problem with Forth is
not having a real standard yet. There are many variations and some of the newer versions for the PC have tools not available on older models due to the lack of video and disk support. I would like to recommend using F83 by Laxen and Perry as they have versions for CP/M and PCDOS, with others having ported it to 6809/Flex and many 68000 based systems. I think we can find a version for all small systems. What would be necessary is to created a test suite that we can use to check out compatibility between different adaptations and thus set the standard which others can then write their programs for.

I dislike choosing only one option and got some help in picking another language when I came across a catalog from the C Users Group (2601 Iowa, Lawrence, KS 66046 (913) 841-1631). These people have been around longer than TCJ (although not by much) and have amassed a large collection of programs and libraries. What caught my attention was the number of adaptions of Small C.

This program by Ron Cain (v2.0), covered by J.E. Hendrix in his Small C Handbook was originally done for CP/M and is available in the Micro C disk collection (K7 & K8 v2.0, K36 & K37 v2.1). Several versions also appear in the C users disk collection (CUG222 & CUG223 SMALL C v2.7 enhanced by F.A. Scaccitichi for CP/M, CUG221 for 6809 FLEX, CUG146 for 6800 FLEX) which you can get from the C Users Group (PCDOS and UNIX formats only) or from Elliam in other formats.

What I would like to see is the revival of the Small C language for small systems. In other words, start redoing it so we can use it on as many of the small units we support as possible. We would also need to develop a set of tests to make sure that the extensions in the code are portable between all models. I need input from you C programmers on this, as I am not a C guru but do feel this is important to the magazine. From the readers point of view, this should be a great solution to the compatibility problem and all for so little energy and money.

IS IT WORTH IT???

For TCJ to meet its desired goals of supporting and educating our readers, we need tools that are portable to as many different and diverse systems as possible. I can only see that happening if we narrow down our scope and projects. If we can provide the tools and programs in one or two languages and everyone can use them, then the trade off’s are worth it. Now using higher level languages is not possible for everything, and I plan on keeping those assembly language listings. The point is to have three options, assembly for the specific tasks at that level (like porting the Small C or Forth code), and then Small C or Forth for tools and general applications. Forth for maybe the more embedded type of operations, and C for the Text oriented and general purpose items. As with anything, I am sure the lines between the uses will cross over many more ways than we can think of at present.

Do I feel it is worth it? I certainly do! I have been harping about an universal operating system and thus a language standard is but one step in that direction. The main objective is making sure our readers get and learn the most from our articles. For our writers it provides them with a set of tools that they know will work with all the different systems. Beginners can be brought up to speed by using tools that are not overly complex or expensive to acquire. Remember I want to be able to give or ship these items to readers around the world for practically no cost.

Is it do-able? If I can get your support and action this can happen. It will take some time, coding and testing always take forever, but it can happen. What I need is your support, action, some research on your part, and lastly letters and messages about the projects. Let me know.

A BIG THANKS!

I want to thank Fred Emmert for giving me and TCJ's readers a full CCS system.

Fred gave me everything he had acquired three years ago. Now why is this important to TCJ? Well the system has two hardisks and of course a controller. My plans will be to print in the center fold all the hard disk information that came in the VERY complete set of documentation. Fred had even printed out the hard disk BIOS.

I tried to talk Fred into giving us an article on his new system (at least new to him) a YASBEC with 3.5" hard drive. It was just great, here I was taking this 3 foot by 3 foot cube of very heavy iron and he was pointing to these two little boards and hard drive. The new unit takes less than a cubic foot and has many times more speed, hard disk capacity, and uses lots less power. The contrast was mind boggling.

I plan on having more about this and YASBEC's in general. So far now, thanks again Fred!

What next?

Well next time, I hope to have some tails about new CP/M systems that have entered the TCJ headquarters. Keep Hacking.

IN ISSUE #62

MOVING FORTH Part III.
Brad answers some questions then moves on to some special and very important words.

Terry Hazen returns with a SCSI EPROM Programmer. This two part discussion covers the hardware in part I. Software follows in part II.

Ralph Tenny completes his classic programming for beginners with Programming the 6526 CIA for Commodore 64's. An excellent review of I/O programming techniques and solutions.

TCJ's regular writers will provide their ever wonderful hints and helps to keep you up and running on Classic Computers.
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, weekend or write
Chuck Stafford
4000 Norris Ave.
Sacramento, CA 95821

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 in US funds. Resetting of ad constitutes 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

New from M&T Books!
Forth:
The New Model
by Jack Woehr
$44.95
1-55851-277-2
Available at bookstores everywhere or call 1-800-468-3987

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 current Z80 code. You can increase your processing performance by up to 300 percent in a matter of days!
Approximately 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