Small Scale Computing Since 1983

Supporting the *Trailing Edge of Technology*

The Computer Journal

www.psyber.com/~tcj/

Issue Number 81 Spring 1998 US $4.00

ISSN # 0748-9331

Hands-on Hardware and Software

The Computer Journal
www.psyber.com/~tcj/

Founder
Art Carlson

Editor/Publisher
Dave Baldwin

Previous Publishers
Bill D. Kibler
Chris McEwen

Technical Consultant
Bill D. Kibler

Contributing Editors
Ronald W. Anderson
Richard Rodman
Helmut Jungkunz
Tilmann Reh

The Computer Journal is published by DIBs Electronic Design and mailed from *The Computer Journal*, P. O. Box 3900, Citrus Heights, CA 95611, (916) 722-4970.

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

Registered Trademarks

TCJ *The Computer Journal*

Issue Number 81, Spring 1998

Editor's Column

New and Old PC's

And in this issue...

New Computers / PC98

If you don't think that MS-DOS PC's with ISA bus plug-in slots aren't the *Trailing Edge Of Technology,* that means you haven't seen the **PC98** specs from Microsoft and Intel. In order to qualify for the WIN98 logo, your motherboard is not allowed to have any user accessible ISA bus slots. None. Zero. All I/O is to be done thru PCI slots or one of the two serial busses, USB or IEEE 1394.

For the latest and greatest systems running only new software and desktop applications, that's probably a good step. PCI and the serial busses unload the CPU to a certain degree (they require specialized chips) and they all are a part of Plug'n'Play. For small companies selling ISA bus I/O cards and writing low level software, this is Not good.

PCI and the serial busses require network style protocols to communicate with them. They are also licensed technologies that must be 'registered' for the Plug'n'Play software to be able to identify it correctly. How do you get licensed and registered? First you reach for your wallet and ...

Though ISA bus may be *'old and slow'*, you didn't have to ask anybody's permission to play. Many small companies made a decent living from building specialized ISA bus I/O cards and software. You are supposed to pass the FCC tests, but many didn't bother.

With the expense and complexity of PCI and the serial busses, some of these companies will be put out of business. The Win98 machines are heading towards becoming a closed system with only the *'big boys'* allowed to play. Another example of pricing the *'little guys'* out of the market by making the up-front cost of the technology too high for them to play. Microsoft says it's not a monopoly. ?? I think they'd like to be.

Because there is so much business in the ISA cards, there may be a split in the PC industry. There are already several alternatives to MS-DOS. Some of the smaller motherboard manufacturers may find a niche in the ISA bus motherboard business.

There are a number of articles on related subjects in this issue. In The Computer Corner, Bill Kibler writes about **'The Move to Linux'** which is *'the move away from NT'* and *'the move away from Win95'*. My interest is in control and communications so I wrote an article on 'I/O Software and Operating Systems' which is right in front of Bill's article.

Old Computers / Year 2000

I had thought that my old PC's would be just fine with the Year 2000 since DOS supports dates until 2038. Then Bill Kilber called me up and told me about BIOS problems with the PC's Real time clock. I downloaded a Y2K test program and found out that none of the five PC's I have in use will properly support the Year 2000 at the BIOS level.

After doing some research on the web, I found out that AMI and Phoenix didn't upgrade their BIOS's until 1995 or 1996. The suggestion was that users should get upgrades from their motherboard manufacturers. I did some more research and called up the manufacturer of my 486 motherboard made in 1990 to see about getting a BIOS upgrade. They said they don't support motherboards that old. !!!

Since new PC98 motherboards aren't going to support the ISA boards and applications I run, the choice isn't as simple as getting new motherboards. I have a number of customers in the same situation although many of them don't know it. I'm looking at the possibility of modifying the BIOS roms in my machines. If I succeed, you'll hear about it here in TCJ.

Dave Baldwin

TCJ Editor/Publisher

Many of TCJ's authors are using Linux and this is reflected in several articles this time. Frank Sergeant talks about using a Linux kernel compilation in **Stress Testing PCs with Linux**. Rick Rodman covers using Samba on Linux as a file server for PCs and Linux developers resources in **Real Computing.** Bill Kibler has two **Computer Corner** articles this time and they're both about Linux, especially compared to Microsoft products. And since people are constantly asking about writing I/O programs, particularly serial I/O, on different operating systems, it seemed appropriate to put my article on **I/O Software and Operating Systems** right before Bill's articles.

Part 2 of the article by Hal Bowers and Cameron Cotrill on **Banked/Portable I/O System (B/P BIOS)** is in this issue. Rick Rodman mentions the **Unoffical CP/M Web Site** in Real Computing and the next article is **The Unofficial CP/M Web Site** by Tim Olmstead who maintains it and collects CP/M related source code and manuals for the site.

Brad Rodriguez is back with **Multitasking 8051 CamelForth**, an extension to his 8051 CamelForth which was published in TCJ. Ron Anderson switches from C to BASIC this time with an article about tracking the Lottery along with a few opinions and some comments about older PCs and source code for a BASIC interpreter he found.

On the hardware side, we have two installments of Dave Brooks **Simplex III** article. He goes over the microcode and schematics for one of the cards. Hal Bowers show how to modify the YASBEC in **High-Density FDC for YASBEC**.

About The 'Net

by Ted Deppner

Internet Info

Technology

A little beyond all the glammor and glitz surrounding the 56K modem battleground is the wonderful world of reality, where the physical laws of the universe (and the FCC) say, 'ha! 48Kbps is all you get!' Due to FCC limits on how hard you can drive a phone line, and the limits of 56K technology, the maximum bandwidth you can achieve is 48k. In practice of course, this is 42k to 46k on average.

We sit here on the brink of 56k modem technology becoming the standard, and look at the changes over just a few years. Ten years ago 1200 baud was top dog. Soon came 2400, then 9600! Each time of course, we hit the impenetrable ceiling, and 'this is the fastest it can go.' 'The world is flat,' 'the moon is green cheese,' when will we learn?

It used to be that that copper could do a maximum of 10 megabit traffic, and that fiber could do 2 gigabit. Then Category 5 wire came into widespread use, and other improvements followed. Today copper can do 2 gigabit, and your modem can do 56 kilobit with ease.

Well, only for today. Rockwell International recently announced their plans for what is beyond 56k for the modem market. The plan calls for a 1 megabit modem, based on Digital Subscriber Line technology. DSL technology (and ADSL in particular) can take a single pair of copper wire, and pump 1.5 megabits over it.

In the good old days, we made modem servers by buying 32 modems, multi-port serial cards, and writing customized software to turn a unix PC into a *terminal server*. This process involved various things, (sacrifices, ritual and prayer) and often they worked.

Today, the 'box' that does 56k has 4 plugs: power, 2 T1 phone lines, and a local network. It can handle 46 users at a time, modems from 300bps to 56K, and ISDN at 64k or 128k. It doesn't have a screen or a keyboard, only 4 idiot lights. Oh, and it only costs $17,000.

That box (specifically the PortMaster 3 made by Livingston) does its magic by removing one step in the way modems work. Modems, as you know, are Modulator/Demodulators, and work by turing digital signals into analog ones. The "T1 phone lines" are T1 circuits, that have 23 fully digital 'lines' on them. The PortMaster takes those digital lines directly, avoiding the need for another modulate/demodulate step. Less signal conversion means less noise and more usable bandwidth available for communication.

On the horizon are DSL technologies, internet telephony products (or 'TeLANphony'), cable modems, and broadband satellite to name a few. The cable modems and broadband satellite don't seem viable for the mass market. If they are fully deployed nationwide, there aren't enough satellites up there, nor is there enough bandwidth in the world to supply every cable modem subscriber.

In the last 3 years the cost of the service provider's network connection hasn't changed much. In that time the average consumer speed has gone from 14.4k to 56k. If the consumer speed doubles from 28.8 to 56k, and the service providers costs remain the same, the provider's user capacity was just divided nearly in two. Supporting cable modems at 400k to 700k each is an even worse situation.

As always technology marches forward. While some of the emerging technologies stand the chance of outstripping the carrying capacity of the net, there is no doubt it's going to be a fun ride.

Side Notes:

Livingston was bought by Lucent Technologies a few months ago (Lucent is the technology 'branch' of what used to be AT&T). Lucent (formally Bell Labs) is credited with developing a minor thing in the industry, the transistor. You can read about it at *http://www.lucent.com/ideas2/heritage/transistor/inventors.html*, it's good history, and it all began only 50 years ago.

Information about Rockwell International's next consumer modem technology can be found at *http://rockwell.com/News/PressRel/PR971028.html*.

Internet telephony takes those T1 phone lines and sends voice through the internet at a fraction of the cost of a direct dialed call. Using compression and other bandwidth optimising technologies, instead of just 23 phone lines per T1, you can reach 100 to 200 per T1 with relative ease. This is a big win for large companies for their internal phone traffic, or for upstart local and long distance companies.

Ted Deppner takes care of operations at *psyber.com* where TCJ's Web pages are.

Letters and News

All Readers

MINI Articles

READER to READER

Send your letters for Reader to Reader to:

The Computer Journal
P.O. Box 3900
Citrus Heights, CA 95611-3900

Subject: Subscriptions; progress on various fronts

From: Herb Johnson <hjohnson@pluto.njcc.com>

Read through the TCJ Web page, glad to see your progress. I see you are asking for subscriptions from your advertizers, so I will mail you today a $24 check to start my subscription from issue #80, which I don't believe I recieved. I've missed Rich Rodman's articles in particular: his interests and mine coincide on small networking like his "tiny TCP" stuff - has he advanced that work lately?

I've said I would write something on "classic Macs", or compact Macs or "9-inch" Macs, which I'm now also selling as well as my S-100 stuff. But this is something new to me, and I've waited for my sense of that "domain" to jell into something that might also be interesting for your/my readers. One thing I like about the small Mac owners is their enthusiasm for their systems. In addition, these are relatively useful systems even by today's standards for CASUAL personal computing. I will consider some kind of writeup in that context. As these systems are not bus-based, few folks are doing any elaborate hardware development. There are serial-based odds and ends however, and the old Mac CPU cards themselves are potentially hackable: problem being the development tools are scarce. The opposing market force to providing Mac tools and technologies is the number of inexpensive single-board computers and their free development tools. Still, even a Mac as terminal and filestore is a useful accessory application. If you see an article in this I'd be encourged.

I'd like you to consider referencing my Web page: http://pluto.njcc.com/~hjohnson/ which describes the kind of S-100 documents I offer, as well as some tidbits on S-100 systems. As time progresses I will also list the S-100 cards I have available, but of course those interested can ask. I'm also buying S-100 cards and systems at very modest prices, for sale at modest prices.

The short story in the S-100 world is the obscene prices collectors are paying for Altair 8800 systems. Most of the rest of the S-100 world sells for zero to tens of dollars per card - or system. The increased use of the Internet has permitted more of this stuff to be traded around: the comp.os.cpm maillist is more active these days with people asking the usual questions about what to get, where it is, and how to use it. I've had more interest from those wanting documentation for their "new" computers, less interest in cards, modest interest in complete systems (which I rarely provide as it's a lot of work!).

The Web page also lists early Mac systems for sale, also well-priced; and my astronomical interests. I'm spending a lot of my time on amateur astronomy: I'm invloved with the TASS amateur sky survey project, which was written up in Sky and Telescope magazine a few months ago. Amateurs with CCD cameras monitor the sky around the celestial equator, looking for variable stars and whatever is out there at 10th-12th magnitude. It's an interesting study on shoestring science.

My regards to you and to your readers, and my previous readers.

Herb Johnson, still "Dr. S-100"

Ed. - I've been trying to collect Digital Group systems. Here's an email conversation I had recently with someone who knows about them.

Subject: Digital Group System in photo

From: Rex Widmer <rwidmer@sound.net>

I was surfing the net and saw a photo of you behind a vintage Digital Group System - dress cabinet, in the usual style of lid removed! *(on the Vintage Computer Festival web pages, www.siconic.com/vcf/)*

I wonder if any of the post bankruptcy products were in the machine. Hundreds of Z80+ (higher clock rate, real time clock, multiple boot eprom, heartbeat, etc) processor boards were built down here in my dungeon and sold into the market. Many improved motherboards (one foot square, 1/8" thick PC), Also lots of I/O Processors, really similar to the (later on the scene) Hayes ESP cards, a fair number of TV80's (H89 emulator, but bus connected) and a tiny number of solid state Pseudo Disks (512K in 1981), and various hard drives.

There are still some of these machines (with all of the RWS replacement parts instead of the Digital Group) running 24 hours per day. Feed them a new Z80 chip every now and then and they just keep working.

Sometimes I wish they would all be museum pieces - but the customers paid real money for them, and they (at least the RWS versions) were designed to run "forever with the exception of things like EPROMS that may need reprogramming and fans which need replacing.

Dave Baldwin / The Computer Journal wrote:

I actually know very little about the Digital Group machines except that

they were comtemporary with the Altairs and Imsai's. I read about them first in a Byte article fifteen years ago. After I took over TCJ, I started collecting them but I'm short on documentation. If you can help with any of that, I'd appreciate it. I have two machines/ motherboards, two quad tape drive units in cases and one dual eight-inch drive unit.

I'd love to hear about these machines, especially if you could write some articles for TCJ about them.

Subject: Re: Digital Group System in photo

From: Rex Widmer <rwidmer@sound.net>

The Digital Group machines emerged primarily from the Ham marketplace as opposed to the computer hobbiest. Lots of overlap between the two areas.

I beleive they were first shown at the Dayton Hamvention in 1975 - matching what was shown in Dr. Suding's original Byte articles - interchangable 8080 and 6800 cpus, not much later with the Z80 which consitituted most of the production.

The company entered Chapter 11 - reorganization in early 1979, and was liquidated in August 1979 (one creditor would not sign up for the 100 cents on the dollar with interest recovery plan). A liquidation auction was held at the Digital Group facility in Denver in the fall of 1979.

Several small companies (mine included) stepped in to continue support.

The intellectual property - board layouts, copyrights etc, was very poorly protected by the Bankruptcy trustee. Some folks (not me) paid tens of thousands for the original artwork, but then found others with "working tooling" churning out boards from that artwork. Was a mess for those who spent the money.

After seening the original artwork sell for ludicrous prices, I executed a plan to build compatable, replacement cards all of which had significant improvements. The first was a new processor card - Z80 but with many improvements, the next was a 64K static ram card (replaced 8K static, 32K static and 32K dynamic cards) using 2K x 8 technology, also a new motherboard - better power distribution, more accurate geometry, and big enough to not require expansion. A new disk controller was produced, with a design that was about 3/4 generation newer, and much more manufacturable. A series of 4 peripeheral cards was produced using dedicated Z80 coprocessors with onboard eprom and ram (made design and update work much easier). This series included a smart I/O card (buffering and protocol in the card), a 512 K solid state disk, a 80 x 24 (or more) video card, and a co=processor prototype. The last component produced was a daughter card for the processor which provided a SCSI hard disk interface.

A number of articles are possible, mainly an issue of getting enough time.

Speaking of your machines, the quad tapes were an on-going challange (as were several other components). The design was good on paper, then was poorly executed - i.e. the voltage to the drives was not properly regulated, the heads of all 4 drives were wired in parallel, and the head preamp was on the end of a long noisy cable. A chap at Stanford published a newsletter with lots of tips on how to fix all of these issues - Hugh McDonald (later of Tall tree systems as I recall).

The 8" disk subsystems might have one of several disk controllers attached. Original single density card (42 pin NEC LSI), noted for needing mods and wiping out disks if powered down with disk in drive. DIgital Group (or direct clone) double density board - high density card with WD1791 chip. Difficult to manufacture, lack of proper voltage regulator for critical components, (made card not portable, sensitive to system's power supply voltage), and card had too much noise sensitivity. RWS type III controller - fixed all of that (only slight bragging), used newer data seperator technology and proper board layout.

Many operating systems were available for the disk based units. More for the double density boards. CP/M 1.4, 2.2. 3.0, MCOS (CP/M clone), OASIS, DISKMON (also with a variant called PHIMON for the digital tapes you have) and others.

The machines still running are typically involved in process monitoring - a testing lab in Cincinatti has a bunch - Dow Chemical Co had a bunch (last time I heard), and several local medical centers who have dedicated message switch applications running on them.

Dave Baldwin / The Computer Journal wrote:

Thanks for the info. May I print your email in TCJ's 'Reader to Reader' column?

If you get the time, I'd like to see some articles. My impression is that the Digital Group equipment is rather neglected compared to the Altair and Imsai crowd.

Subject: Re: Digital Group System in photo

From: Rex Widmer <rwidmer@sound.net>

You may certainly print the letter.

Digital Group stuff has certainly not attained the "collectable" status of the Altair / IMSAI S100 machines.

Many of the orginal machines were without fancy cabinets - the kits were frequently sold that way. Also no fancy front panels - nice cost savings by doing all of that type of stuff via EPROM and a mini monitor loaded from audio cassette.

Like the Altair, original machines frequently needed a tweak or two. 8K memory boards had driver (to CPU) problems, origianl motherboards had keystoning in the phototools which made the dual connector I/O cards act up, motherboard voltage drop was a problem when you might have 25 Amps of +5 running 56K of 8K memories (the RWS 32K used about 500 ma!)...

But... the systems could be made to work reliably, once you knew the tricks, and perhaps got rid of a few first generation parts.

Biggest difference was that there was far fewer vendors of significance until after the bankruptcy. Godbout did make a suitable 32K static card, that probably hit the street prior to the failure. Doug Bell A.K.A Bell Controls adadpted a fine Tarbell S100 floppy card to the system prior to the failure, but that adaptation took a fair amount of mechanical fiddle work and wire wrap to lash a S100 card into the D.G. chassis / bus structure.

The surprising point is that there were active user get togethers at the Dayton Hamvention for over 10 years after the bankruptcy. Even now, almost 19 years after, there will no doubt be at least an informal gathering of the faithfull in 10 days at the Hamvention.

Online with TCJ

Internet and BBS resources

TCJ has both an Internet Web site and a BBS that you can download files from. I keep a lot of the same files on both systems, but each is quite different. The web site provides easy access to anyone with an Internet account but is limited in disk space.

The BBS requires you to dial it up directly (which is a long distance call for most), but is limited only by the size of the hard disk I can provide. You can also access the BBS with modems as slow as 1200 baud (and as fast as 14.4Kb) where most Internet providers don't allow access below 9600 baud now.

I've been trying to get the resources togther to convert the BBS over to Linux. When that (finally) happens, the Web site and BBS will have a lot more in common.

The BBS phone number is (916) 722-5799. If you don't want to go through the signup procedure, you can use the generic TCJ logon. Use "Computer" as first name, "Journal" as last name, and "Subscriber" as the password.

File areas on TCJ/DIBs BBS.

1) **General** - files of general interest that don't fit into the other areas.

2) **TCJ** - files and articles from TCJ.

3) **Assembly Languages** - files that don't fit into the areas below.

4) **Z80 and CP/M** - files and source code for Z80 and CP/M systems.

5) **ASM 8051** - source code and assemblers for the 8051 and it's decendents.

6) **ASM-86** - DOS and x86 code and programs.

7) **ASM-68 (Mot)** - a few files for 68xx processors.

8) **Forth** - files including Frank Sergeant's Pygmy Forth and Brad Rodriguez's CamelForth.

9) **C Language** - general C files including source code for some of my programs that haven't been in TCJ.

10) **Communications Programs** - modem and serial port programs and info.

11) **Programmers Utilities**

12) **General Electronics**

13) **MS-DOS Shareware** - some DOS shareware I've found useful.

14) **TCJWeb** - HTML and other files from the TCJ Web site.

15) **FAQS** - Internet FAQ's from the TCJ FTP site.

The TCJ Web site now has 15 main pages and dozens of links and files. The TCJ Web pages are at:

"http://www.psyber.com/~tcj/"

TCJ Web Pages

Asynchronous Serial Data Communications
Datasheets and applications notes in PDF format, source code from TCJ articles and other sources, terminal and serial port info programs, and links to other sources of information.

Current Issues
Table of Contents of recent issues along with the text of some articles and source code from some articles.

Back Issues
Complete Back Issue listing for TCJ with the table of contents.

The TCJ Store
Items for Sale from TCJ.

Advertisers
Links to and files from TCJ advertisers.

TCJ FTP
FTP directory with FAQ's, schematics, and software.

FAQs and Files
Links to Internet FAQ's and file archives.

People and Places
Links to persons and places on the Internet.

Support Groups
Support listings for older systems with links to those we know are online.

TEOT
Links to web pages with information about *The Trailing Edge of Technology*. Older systems, terminal info, and 'online' computer museums.

Kaypro
TCJ Kaypro catalog - ROM Upgrades and stuff

Tools
Some software tools and files

GIDE
Tilmann's Generic IDE interface for the Z80

TCJ Writers and Editors
Links to web pages put up by TCJ authors and editors.

TCJ Readers Resources
Web pages by some of TCJ's subscribers.

Stress Testing PCs with Linux

By Frank Sergeant

Special Feature

Stress Testing your PC with Linux

How can you get your work done if you cannot rely on your computer? Fortunately, I had at least two unreliable computers of my own to experiment upon. Intermittent troubles are very difficult to track down. Is it the hardware or is it the software? I will describe the test I now use to answer that question.

***The Test**

Compiling the kernel of the Linux operating system uses the resources of the computer so intensely that it makes a good test of the hardware. If the compile "blows up", then the hardware is bad. This may take a leap of faith at first, but I will try to explain why I think it is a reasonable hypothesis. You might think that the compile blowing up means a software bug in the compiler. But, if that were true, you would expect the compile to blow up at the same place each time. Compiling the kernel consists of running the compiler on many different source files, producing an object file for each one, then linking the object files together to form the final kernel. (The kernel is the heart, so to speak, of the Linux operating system.) Typically, if the compile blows up, it does so during the compile of a different source code file each time, sometimes after 45 seconds, sometimes after 2 or 3 or 20 minutes, etc. It is this random nature of the problem that suggests bad hardware and not bad software. More details about this test, examples of hardware problems and what to do about them, plus additional case histories are available from http://www.bitwizard.nl/sig11/ or in the file CGG-SIG11-FAQ available on Linux distributions and ftp sites (for example, see http://www.infomagic.com).

Remember, we are dealing with an intermittent problem. If the computer didn't work at all, we wouldn't have a problem, we'd just replace it. But, when it crashes once a day or once a week, especially if we are running Windows, how do we know it isn't Windows causing trouble? The uncertainty is disturbing, not to mention the problem of losing your work, corrupting your data, and upsetting your employees and customers. Because we are dealing with an intermittent problem, compiling the kernel once does not prove the hardware is good. The more times in a row you compile the kernel successfully, the more likely the hardware is good. However, a single failed kernel compile proves the hardware is bad. Throughout this article, a "test" or a "stress test" will mean running a kernel compile. Since even bad hardware can sometimes run a single test successfully, we must run a series of tests, 10, 20, 50, or more, depending upon our patience.

Following is the shell script to run the test multiple times. I name it 'stress.sh' and invoke it with 'stress.sh 10' to run a series of 10 tests. It reports the success or failure results to the screen and also writes them to a log file named 'stress.summary'.

```
———————— stress.sh ————————————————
#!/bin/sh
cd /usr/src/linux
passnum=1
while [ ${passnum} -le  $1  ]
do
 echo -ne 'starting compile #'$passnum at $(date)\
        | tee -a $HOME/stress.summary

 # perform the compile
 (make dep; make clean; make zImage) 2>&1 > /dev/
null

 if [ $? -eq 0 ]   # test result code from running
                     the compile
 then
 echo ' - success -'  | tee -a $HOME/stress.summary
 else
 echo ' - failure -'  | tee -a $HOME/stress.summary
 fi

 # increment the pass number
 passnum=$((passnum+1))
done
————————————————————————————————
```

***What is Linux**

Linux is a free clone of the Unix operating system. It runs on various hardware, including PCs ('386 or better). It is a more reliable operating system than DOS or Windows. It can co-exist with DOS or Windows on a PC, allowing you to decide at boot time which operating system you wish to use.

***Case Histories**

****First Case History: Pentium 90 MHz (24 MB RAM)**

***Symptoms

This computer worked most of the time. It would lock up perhaps once a day or more often in Windows 95. It would lock perhaps every few days in plain DOS. It essentially never locked up in Linux. Even in Linux, though, there were signs of trouble: a kernel compile would never complete successfully and a large LaTeX compile would fail about

one in three times. It would blow up at random places. Even then, the computer did not lock up, but just reported an error and returned to the command prompt. So, this is why I say that Linux is more reliable than Windows or DOS: on the very same hardware, even though the hardware was defective, Linux did not lock up but the other two did.

***Testing

I first suspected a RAM (memory) problem. I spent several days substituting various combinations of RAM chips and re-running the tests. None of the combinations of RAM chips solved the problem.

Next, I pulled all of the cards that weren't essential to running the test. This didn't help. Next I checked the motherboard settings with the manual. The CPU was rated at 90 MHz and I wanted to make sure the motherboard hadn't been misconfigured to run at a higher speed. The settings were correct.

Then, I changed the clock settings to reduce the external clock speed from 60 MHz to 55 MHz. This "fixed" the problem. That is, a series of tests completed successfully. I replaced the cards and it still worked. I reset the jumper to 60 MHz and again the tests failed. I reset the clock to 55 MHz and the tests passed. The motherboard has two sets of jumpers. One set controls the external clock speed (60 MHz, 55 MHz, etc.) and the other controls the multiplier for the CPU clock. The lowest this would go was 1.5, thus, for a 90 MHz Pentium (really an AMD chip), the external clock should be set to 60 MHz and the multiplier to 1.5, giving 90 MHz for the CPU. I was disappointed in having paid for a 90 MHz machine but only receiving an 82.5 MHz machine. But, I was much happier to have a reliable machine at the slower speed than to suffer the data loss and aggravation of the faster speed. One test run took about 17 minutes. I ran a series of 30 tests overnight — all successful.

The CPU has a fan and a heat-sink combination attached to it with a plastic clamp — the heat-sink is sandwiched between the CPU and the fan. The fins of the heat-sink face the fan and the smooth side of the heat-sink touches the CPU. The idea is that the fan blows air over the fins to cool the CPU. I was surprised to see that the heat-sink was merely placed "bare" against the CPU. In the old days, voltage regulators and power transistors and other hot-running ICs usually were mounted to their heat-sinks with thermal compound (heat-sink grease) that was designed to improve the heat transfer. I hear that the lack of using the thermal compound is typical these days for CPUs. I am suspicious of this practice and think it may be laziness or ignorance on the part of computer assemblers. I remounted the heat-sink to the CPU using heat-sink compound, reset the external clock speed to 60 MHz and ran another series of tests. Yes, this fixed it. Now the machine runs at full speed.

Second Case History: Pentium 166 MHz (16 MB RAM)

***Symptoms

This machine, in a network of three computers, all running Windows 95, would lock up occasionally. It was a new machine and no reason or pattern was obvious. The installer planned to return it to his whole-seller for replacement. I suggested we run the stress test on it first to see if the problem really was the hardware.

***Performing the Test

There were some problems setting up the test that I worked around by temporarily installing my spare 120 MB hard disk. I installed Linux on that disk and was able to run the test without disturbing the primary hard disk on the machine. I felt I needed to do the tests this way because the primary hard drive was set up as a FAT32 file system with a single partition and would have been difficult to repartition.

***Results

One run of the stress test took about 13 minutes on this machine. To my surprise, the first test was successful. I set up the stress test to run overnight and it ran successfully 57 times in a row. Of course, this does not prove the hardware is good. The stress test might have failed on the 58th time. Also, the stress test does not test everything. Nevertheless, 57 successes in a row with no failures is a good sign that the hardware is probably good.

Our next best guess as to the cause of the lockup problem was that it was caused by WordPerfect version 6.0a (a 16-bit Windows 3.1 program) running on that machine. I had heard from an experienced WordPerfect user that version 6.0 was particularly buggy. WordPerfect was upgraded on that machine. The computer has not locked up since then, although it has only been running for a few days. We will keep an eye on it and hope for the best. Meanwhile, it appears this was a software problem due to WordPerfect.

Third Case History: '486 40 MHz (20 MB RAM)

***Symptoms

This machine gave frequent problems under Windows 95 and under Windows NT. The registry would become corrupted, software installs would fail, the computer would lock up, etc. Even under Linux there was trouble, although not as much as under Windows. Linux would usually lockup if left to run overnight.

***Special Considerations

This machine presented an interesting problem due to its speed and its value. Because of its relatively slow speed, the stress test takes a long time to run. This makes the testing extra expensive in terms of time. Then, because of the machine's age and speed, it is not worth very much. So, the test costs more to run and the best possible final result, a working '486 machine, has little value. Therefore, if a '486 system seems to be causing trouble, instead of testing and repairing it, it might be more "cost effective" simply to replace it with a faster machine.

Nevertheless, the stress test might be useful even on old, slow machines as a quick "Good/Bad" test. If the test fails, replace at least the motherboard and CPU, don't try to locate the faulty component. If it passes the test, let it run overnight or over a weekend to give more confidence that the hardware is ok and that the trouble lies elsewhere.

***** Test Results**

Since I had the 120 MB hard drive already loaded with Linux after testing the Pentium 166, it was not too difficult to open the case of the '486 and connect this drive. Because of other troubles I had been having with the '486, I was fairly sure it would not pass the stress test. Sure enough it failed right away with lots of errors.

I changed various this around over a period of about a week, running lengthy tests between changes. The final result appears to be a reliable 12 MB machine. It seems there were two problems: (1) probably the 16 MB SIMM was bad and (2) the CPU needed a heat-sink and fan.

The length of time each test took depended upon the amount of RAM and whether the external cache was turned on or off. A single test with 4 MB of RAM was still running after eight and a half hours when I finally killed it. It might have passed the test, but I'm not going to run a series of tests where each one takes that long. Thereafter, I tested with at least 8 MB. Normally, a series of 10 tests would take between 8 and 15 hours.

After removing what later turned out to be the bad 16 MB RAM and replacing it with 8 MB of "known good" RAM that had passed the stress test in another machine, I ran a series of 10 stress tests. This took about 15 hours. Yes, with a '486-40 and only 8 MB of RAM (and a fairly fat Linux kernel) each compile took around and hour and a half. It passed the first 6 tests and the 8th test, but failed the 7th and 9th and 10th tests. You see how tricky this is? If I had stopped after only 5 or 6 tests, I might have concluded incorrectly that the hardware was now fixed. Once the hardware is fixed, how many tests would it take to prove it had been fixed? We need to consult a statistician, but I would say certainly more than 6 and probably more than 30.

Although I tried a number of changes, the next beneficial change was to put a fan and heat-sink on the CPU. This allowed a successful series of tests. To be sure the problem wasn't due just to the lack of heat-sink and fan, I replaced the 16 MB SIMM and reran the tests which again failed. So, the final configuration has 12 MB of RAM and appears (after passing about 22 tests in a row) to be fixed.

***Is it Worth It?**

It depends upon whether you are doing this for fun or not. If you are not doing it just for fun, it doesn't make much sense to spend a lot of time running the tests and swapping out memory and rerunning the tests time after time if you have a slow '386 or '486, especially if you have less than 8 MB of RAM. With at least 8 MB of RAM, it might make sense to run the stress test on an old '386 or '486 as means of verifying the hardware is good. Forget it, though, if the machine has less than 8 MB of RAM. Even then, expect it to cost a number of hours with, at the very best, a final diagnosis of "probably good". If any of the tests fail, the machine probably should be replaced immediately because continued component replacement and retesting won't be worth the cost in time of doing the tests. If the old machine tests bad, you might be better off buying a new motherboard and CPU. The latest ad I have from DreamTech (800-237-3263) shows a 150 MHz Pentium motherboard/CPU for about $230.

On the other hand, if you are using your computer for important work and are experiencing problems, performing the test could be very worthwhile. If your problems really are due to bad hardware, this is a way to fix them and to confirm that the hardware is now good.

***New Machines**

On new machines, a 150 MB partition could be reserved for Linux from the beginning and Linux installed in it. It is hard to buy a machine new with as less than a 1.2 GB hard disk, so this might be a reasonable use of about a tenth of the disk or less. Run the stress test at least overnight before putting the machine in service. Later, if there are questions about the machine's reliability, it is a fairly simple matter to boot to Linux and run the stress test — you don't even need to open the case.

***Efficiency**

You might complain that this stress test is overkill. Why should you need to install an entire operating system, compiler, etc. just to test the hardware? I agree. The problem is that RAM testers apparently do not work reliably. So, you take your RAM into a computer store where the clerk tests your RAM and says it's good. All that means is the RAM is not always bad. The RAM that causes you trouble is the RAM that is almost good and fails only in rare circumstances, perhaps once or twice a day or week. It is the intense and effectively random testing of memory that occurs as a by product of compiling a Linux kernel that really gives the memory a proper workout. It stands to reason that the same sort of testing could be done by a program designed for just that purpose, making the test a lot easier to perform (and perhaps faster). It should fit on a floppy instead of requiring many megabytes of hard disk and RAM. I suspect such as test already exists or will exist, but I do not want to write it or search for it or verify it myself. I'll wait until someone finds it for me. Meanwhile, the Linux kernel compile is overkill, but it's all I've got and it is a lot better than guessing.

***Conclusions and Speculations**

Don't tolerate flaky or questionable hardware. Now that a test exists you don't need to keep wondering whether it is the hardware or the software.

Always use thermal compound when mounting a heat-sink to the CPU. The goal is to have the CPU itself run at room temperature, not just to have a cool heat-sink.

Old, slow computers may not be worth fixing unless doing it for fun. The test is more expensive to run and the end result is still an old, slow computer. For business purposes, your time and money might be better spent upgrading the motherboard and CPU.

The test is expensive and tedious to perform. Pre-install Linux whenever possible, if only to use for hardware testing. Since that might not always be practical, I plan to put together a Linux file system that will fit on a ZIP drive so I can copy it to a temporary Linux subdirectory on an ordinary DOS (FAT16) disk and run the test from the DOS disk. This probably won't work if the disk is a Windows 95 FAT32 or a Windows NT file system. Because of this, I suggest, if you must install Windows, that you use a plain FAT16 file system.

Expect a cheaper, easier test to appear. If you find it, please tell me about it. It shouldn't take 120 MB of disk space and shouldn't require the installation of a full operating system and compiler just to check out the hardware. On the other hand, beware of simplistic memory tests, including your computer store's RAM tester, which will pass bad memory.

Don't ever buy any Windows-only hardware. Watch out especially for modems and printers that work only with Windows. Watch out for Windows in general. I read over the bug list in Kermit 95 for Windows 95 and Windows NT where the programmers listed bug after bug in Windows 95 that they had to work around. From that bug list, it sounds like Windows NT might be the better choice if you insist on running Windows. On the other hand, some programs that run under Windows 95 do not run under Windows NT.

Always make proper backups of your important files. Modern desktop operating systems are too complex to trust completely. Hardware can and will fail eventually even if it is not failing right now.

Am I unfairly down on Windows? Possibly so. After all, the troubles I've had with it have been on machines that I now know had hardware problems. I'm interested to see how it works on the fixed the hardware. Nevertheless, let's not go out of our way to tie ourselves to Windows. Let's keep our options open.

Banked/Portable I/O System (B/P Bios) Pt 2

by Harold F. Bower and Cameron W. Cotrill

Special Feature

Z80/180 Systems

Advanced BIOS

As we discussed in the first part of this article, our efforts in the design and construction of our Banked/Portable Bios were directed towards building an architecture that would viably support applications on Z80/180 computers for a number of years. One of the initial ground rules was a prohibition on the use of Self-Modifying code. As in our development of ZSDOS, we want to retain the ability to place operating system components into ROM if desired. This goal was met, but at the price of slightly increased code size, particularly in non-banked systems such as those placed on system boot disks. Exceptions to this rule do exist, but primarily in system components which are intended specifically for execution from RAM such as portions dynamically relocated at load time.

Another key consideration addressed in B/P Bios was the need to provide a mechanism to identify the Bios and to locate Bios-specific data areas and options. The solution to this was Bios Function 30 which returns, among other things, a pointer to the start of a standardized data structure containing configuration information. This vector allows examination or alteration of various parameters such as Data Rates, Drive parameters and configuration flags. Using this vector instead of assuming absolute locations within the BIOS is crucial to obtaining and maintaining portability and standardization.

The Configuration Area pointer serves a secondary purpose as well. The six bytes preceding the returned address point to a string which must begin with the Ascii letters "B/P". The following three characters contain a suffixing identifier for the hardware base. Between us, we have either implemented or planned B/P Bios installations to varying degrees on the following systems:

"-18" - MicroMint SB-180 (HD64180 CPU, 9266 FDC, 5380 SCSI)
"-FX" - MicroMint SB180FX (Z180 CPU, 9266 FDC, 53C80 SCSI)
"-YS" - YASBEC (Z180 CPU, 1772 FDC, DP8490 SCSI)
"-DX" - D-X Designs Pty Ltd P112 (Z182 CPU, 37C665 FDC, Flash ROM) (Add-on 5380 SCSI supported)
"-CT" - Compu/Time S-100 board set (Z80 CPU, 1795 FDC, 1MB Memory)
"-TT" - Teletek (Z80 CPU, 765 FDC)
"-AM" - Ampro Little Board (Z80 CPU, 1770 FDC, T. Hazen's MDISK)
"-XL" - XL M-180 SBC (HD64180 CPU, 9266 FDC, PIO SCSI)

This method of identification can serve to prevent customized programs from operating on the incorrect hardware. For example, the HDBOOT hardware-specific program distributed for YASBEC and Ampro Little Board computers will not modify a Hard Disk boot record unless the system on the Boot Tracks contains either the "-YS" or "-AM" identifiers.

With the additions made to the ZCPR3 Environment Descriptor in Version 3.4, it readily became apparent to us that the Environment belongs to low-level hardware-dependent portions of the Operating System, with allowances being made for high-level use. While this decision may be debated, we adopted the ENV structure and require it within the Bios with only one necessary change. The four bytes associated with the second printer have been usurped to provide data on a resident User Space. To provide compatibility with other vectors added to Operating System components, the four bytes are reallocated as:

1 byte - Number of free 128-byte blocks
2 bytes - Pointer to start of User Space
1 byte - Total size in 128-byte blocks

The User Space is always assigned below all other ZCPR3 System Segments and the Starting Address Pointer serves double-duty as the lowest address in reserved memory. This is needed in hard disk systems since the ALV buffers are dynamically calculated at system load (boot) time. For Non-banked systems, if the amount of space needed by the ALV buffers extends beyond the base of the User Space pointer, a warning is printed to alert users that a smaller system is required to allow full use of the hard drive without overwriting System Segments. In banked systems, the check is performed against the end of the primary 32k System Bank. The use of a byte to indicate amount of free space remaining will allow multiple RSX-like additions to be chained into the User Space in a (hopefully) controlled manner. A similar construct is being developed for banked applications, but has not yet been fully developed.

As most of us are painfully aware, Floppy Disk formats in the CP/M compatible world are woefully non-standardized. To maximize the efficiency of B/P Bios, we have NO required format. Our system may be assembled with hard-coded invariant formats (the smallest code requirements), with calculated skew values or indexed tables, or with a user-configurable suite of non-conflicting formats, including 3.5", 5.25" and 8" drives as well as "High-Density" formats where the hardware is supported. For standard distribution, the

Ampro/SB-180 5.25" formats are included in an auto-select mode. 8" drive capability is in the SB-180, SB180FX, P112, Compu/Time and Teletek versions, but not in YASBEC or Ampro Little Board since those controllers will not handle the higher data rates. Using this scheme, tailoring options may be used to gain every byte and clock cycle possible, for example, by deleting 8" formats if you only have 3.5 or 5.25" drives connected to the system.

The B/P BIOS is divided into a number of files, some of which are machine dependent, and some are generic and need not be edited to assemble a working system. Much use is made of conditional assembly to include option-dependent modules and relocation bases. The Basic file, BPBIO-xx.Z80, specifies which source code elements are used to assemble the Bios image under the direction of an included file, DEF-xx.LIB, which selects features and contains Hardware-dependent equates. Modules requiring customization for different hardware systems are given names which end with a generic "-xx" designator to identify specific versions. These names correspond to the suffix embedded in the identifier string covered above. By maintaining the maximum possible code in common modules which require no alteration, B/P Bios is relatively easy to convert to different machines.

While some of these versions cannot take full advantage of B/P Bios, such as the Ampro Little Board which, without Terry Hazen's MDISK, contains only 64k of memory, the benefits of having only one set of support utilities and common operating procedures across different computers often outweigh the disadvantages of a slightly larger Bios. Those of you with several different types of computers will readily understand.

With B/P Bios, systems may be changed "on the fly" in a manner similar to that employed with NZBLITZ/NZCOM. The system placed on the boot disk's system tracks must be relatively small to fit in the limited space, often sacrificing features. Once started, however, the rules change. A special loader is used to move pre-configured system images in place and start execution. These images are built from an assembled B/P Bios REL file, ZSDOS version 1 or 2 ZRL file, and a Command Processor REL or ZRL file. The utilities to build the image file and load it into position account for both banked and unbanked component locations. The build utility also allows you to alter many of the Environment parameters as the image is built. For example, the extra large Command Processor demonstrated several years ago at the Trenton Computer Fest includes most of the common features from RCPs. Consequently, RCP space is often not needed and may be removed typically freeing 2k for the TPA.

Different system images may be generated with BPBUILD and then loaded as specific functions or configurations are desired, without regard to the system on the Boot Tracks. Tailoring of these image files is performed with the B/P Bios configuration utility, BPCNFG, to include Startup Alias file names unique to that system. In this manner, different images may chain from one to another as system sizes and needs change. A given image file might be duplicated with only minor configuration changes. Hal uses this technique in system backups where two hard disk partitions are activated for backups to external hard drives, but different parameters are configured for various backup media and drive types.

To provide some comparative numbers, a YASBEC with a 20MB hard drive is typically limited to a 53K non-banked system for placement on the Boot Tracks. A 56K system is achieved simply by banking portions of the Bios. With the banked ZSDos2 and the associated large Command Processor, Z41, a 58.0K system is standard with the default 2K RCP and 1.5K IOP space reservations. This expands to 60K with no RCP and 61.5K if no IOP is needed. The large expansion in size with the banked ZSDos2 is due to banking of the ALV and CSV bit storage areas needed for hard and floppy drives. Placing these elements into the system bank means that large system sizes are no longer dependent on the hard disk sizes (yes, even a 200 MB hard drive on a YASBEC sports a 60K system configuration), but with the penalty that normal programs which count ALV bits to determine disk free space and file size no longer function correctly. This difficulty has, in part, been addressed by providing modules in the DSLIB utility library (a companion to SYSLIB and Z3LIB) which "know" about ZSDos2 and properly return disk free space values. Source code for DSLIB was released to Z-Nodes several years ago, and many of the included routines are used in our utilities, such as the ZXD directory lister provided as part of the B/P Bios package.

If even more Transient Program Area is needed, one additional means exists to expand available memory, but at a possible expense. Part of the BPBUILD process is to size the Code and Data requirements of the relocatable modules, and locate them at static addresses for loading. Since the resident portions of both the Z41 Command Processor and ZSDos2 are smaller than their non-banked counterparts, the lower limit for applications programs, which are allowed to overwrite the Command Processor, and Resident System Extensions (RSXes) which nestle immediately below the Command Processor see a higher memory address. The caveat to this approach, however, is that only programs which use the enhanced environment parameters compliant with ZCPR 3.4 and later will function properly. To avoid such potential problems and provide maximum commonality with existing CP/M programs, BPBUILD also allows the user to direct that the Command Processor and resident DOS elements be linked to begin at "standard" locations reflecting sizes of 2K and 3.5K respectively.

THE INTERFACE.

B/P Bios entry points are contained in a Table of 3-byte Absolute jumps at the beginning of the Bios Image. Parameters needed for each function are passed to the Bios in specified registers. To avoid future compatibility problems, some of the ground rules for Bios construction include; No alteration of Alternate or Index registers as a result of Bios calls, and all registers listed in the documentation as being Preserved/Unaffected MUST be returned to the calling program in their entry state. The first seventeen jumps (indices 0-16) constitute the standard CP/M 2.2 jump table. Following those are additional sequences patterned roughly after

CP/M 3 and the B/P-unique extensions. The Bios entry points listed in order of their appearance in the jump table are:

0 - CBOOT Execute Cold Start initialization on the first execution. The code is later overwritten, and the argument of this jump then points to the IOP Device jump table. (all registers used)

1 - WBOOT Execute Warm Restart initialization, reload the Resident part of the Command Processor and log onto the default drive. (all registers used)

2 - CONST Return Console Input Status as; A=0FFH if Character is ready, A=0 if No character is ready. (Uses AF)

3 - CONIN Read a character from the Console, waiting until one if ready, then return it in A masked as specified in the Bios, Normally with Bit 7 set to 0. (Uses AF)

4 - CONOUT Send the character in C register to the console masked as specified in the Bios (Uses AF)

5 - LIST Send the character in C register to the List Device (normally a printer) masked as specified in the Bios. (Uses AF)

6 - AUXOUT Send the character in C register to the Auxiliary Output masked as specified in the Bios. (Uses AF)

7 - AUXIN Read a character from the Auxiliary Input port masked as specified in the Bios. (Uses AF)

8 - HOME Position the head(s) on the selected drive to Track 0. (Uses All primary registers)

9 - SELDSK Select the drive specified by the value in Drive C where Drive A=0...P=15. (Uses All primary registers)

10 - SETTRK Select the Logical track contained in Register BC for a future disk operation (All registers preserved)

11 - SETSEC Select the Logical sector contained in Register BC for a future disk operation (All registers preserved)

12 - SETDMA Set the address in Register BC for a future disk operation (All registers preserved)

13 - READ Read a Logical 128-byte sector from the Disk, Track and Sector set by Functions 9-11 to the address set with Function 12. On return, Register A=0 if the operation was successful, Non-Zero if Errors occurred. (Uses All primary registers)

14 - WRITE Write a logical 128-byte sector to the Disk, Track and Sector set by Functions 9-11 from the address set with Function 12. If Register C=1, an immediate write and flush of the Bios buffer is performed. If C=0, the write may be delayed due to the deblocking. (Uses all Primary registers)

15 - LISTST Return A=FF if the Printer is ready to accept a character for printing, otherwise return A=0. (Uses AF)

16 - SECTRN Translate the Logical Sector Number in register BC (Only C used at present) to a Physical Sector number using the Translation Table addressed in the Selected Drive's DPH. (Uses All Primary regs)

This ends the strict CP/M 2.2-compliant portion of the Bios Jump Table. The next series of entry Jumps roughly follows those used in CP/M Plus (aka CP/M 3), but with corrections to what we perceived to be deficiencies in the calling parameters and structures.

17 - CONOST Return A=FF if the Console is ready to accept another output character, otherwise return A=0. (Uses AF)

18 - AUXIST Return A=FF if the Auxiliary Input has a character waiting, otherwise return A=0. (Uses AF)

19 - AUXOST Return A=FF if the Aux. Output is ready to accept another character for output, otherwise return A=0. (Uses AF)

20 - DEVTBL This corresponds roughly to an analogous CP/M Plus function although precise bit definitions vary somewhat. The Character IO table consists of four devices; defaulting to COM1, COM2, PIO, and NUL. Each has an input and output mask, data rate settings and protocol flags. Not all defined settings (e.g. ACK/NAK and XON/XOFF handshaking, etc) are implemented in all versions, but are available for use. (Uses HL)

21 - DEVINI Initialize Character IO settings and other specified functions. This is very close to the CP/M Plus function and can be used torestore IO configurations after alteration by programs which directly access hardware such as modem programs. (Uses all primary registers)

22 - DRVTBL Return a Pointer to the DPH table for Drives A-P where a 16-bit 00 entry means that no drive is defined. (Uses HL)

23 - MULTIO <Reserved for Multiple Sector IO>

24 - FLUSH Write any pending Data to disk as mentioned in Function 14 above. (Uses all primary registers)

25 - MOVE Move number of bytes specified in BC from the location starting at (HL) to an area addressed by (DE). For banked moves, the Source and Destination banks must have been previously specified with an XMOVE function. Note that this function reverses the functions of the DE and HL register pairs from the CP/M Plus function. (Uses all Primary registers except A)

26 - TIME If Register C=0, Read the Date and Time to a 6-byte field addressed by (DE). If C=1, Set the Date and Time from 6-byte field addressed by (DE). On exit, register A=1 if the operation was successful, A=0 if an error occurred or No clock exists. The Date/Time string is in ZSDOS format as opposed to Digital Research's format used in CP/M Plus for this function. Also, This function must conform to additional requirements of DateStamper(c) in that on exit, register E must contain the entry contents of (DE+5) and HL must point to the entry (DE)+5. If the clock supports 1/10 second increments, the current .1 second count may be returned in register D. A recent addition (suggested by Terry Hazen in TCJ #79) uses BC to return the address of a free-running 8-bit counter decremented every 100 milliseconds. Such a feature is invaluable in user programs such as modem drivers. (Uses all primary registers)

27 - SELMEM Select the Memory Bank specified in the A register and make it active in the address range 0-7FFFH. (All registers preserved)

28 - SETBNK Set the Bank Number in A for the next Disk IO. (All registers preserved)

29 - XMOVE Set Source and Destination Bank numbers

in registers C and B respectively for a future Move (Function 25). (All registers preserved)

This marks the end of the CP/M Plus "Type" jumps and begins the unique additions to the B/P Bios table to support Banking, Direct IO and interfacing.

30 - RETBIO Return the Bios version number and a pointer to internal BIOS data areas as:
A = Bios Version Number
BC —> Page Address of B/P Bios
DE —> Start of Configuration area
HL —> Start of Device Vector table

31 - DIRDIO Execute low-level functions directly on Floppy or SCSI devices. (described below)

32 - STFARC Set the bank number for a subsequent Function 33 execution. (All resisters preserved)

33 - FRJP Switch to bank number specified with Function 32, then execute routine at address in HL, returning to address on stack top. (Uses all primary registers)

34 - FRCLR This entry is used for error exits from banked routines to return to the entry bank. (Uses all primary registers)

35 - FRGETB Load the byte addressed by HL from the bank number specified by the C register into the register A. (Uses A)

36 - FRGETW Load the Word addressed by HL from the bank number specified by the C register into the DE register pair. (Uses DE)

37 - FRPUTB Store the byte in register A to the memory addressed by HL in the bank specified in register C. (All registers preserved)

38 - FRPUTW Save the Word in DE to the memory addressed by HL in the bank specified in register C. (All registers preserved)

39 - RETMEM Return a Byte identifying the Memory Bank number currently in context in the A Register. (Uses A)

DIRECT DISK IO. BIOS Function 31 permits low-level access to Floppy and Hard Disks (currently via SCSI interface) by specifying a Driver Number and desired Function. While some hardware types do not support all of the parameters specified, particularly for Floppy Drives, this architecture supports all types, although specific systems may ignore certain functions. In this manner, for example, a single Format program supports NEC765, SMC9266, SMC37C665, DP8743 and WD1770/1772/179x controller types with widely differing interfaces. Floppy Disk functions are accessed by entering a 1 value into Register B (Floppy Driver Number) and the desired function number in Register C, then jumping to or calling BIOS Entry jump number 31.

FLOPPY DISK SUBFUNCTIONS:

0 - (STMODE) Set the Floppy Disk Controller for Read/Write opns. On entry, Register A contains a Density Flag (0 = Double, 0FFH = Single Density). No data is returned from this function. NOTE: This routine assumes that Functions 1 (STSIZE) and 3 (STSECT) have been called first. (Uses AF)

1 - (STSIZE) Set Drive Size (3.5/5.25" or 8"), Drive Speed (300 or 360 rpm) and Motor Needed flag. On entry, A=0 for normal floppy speed (synonymous with "Normal" 250 kbps MFM Double-Density), A=0FFH for High speed motors (synonymous with "Hi-Density" 500 kbps MFM). Register D=0 if the motor is always on or no motor control is needed, while D=0FFH if motor control is necessary. Finally, register E must contain the drive size as; 0=Hard Disk, 001B=8" Drive, 010B=5.25" Drive, and 011B=3.5" Drive. Nothing is returned from this command. While all of these functions may not be supported on any specific computer type, the interface from using programs should always pass the necessary parameters for compatibility. NOTE: This routine assumes that Function 2 (STHDRV) has been called first. Call this routine before calling Function 0 (STMODE). (Uses AF)

2 - (STHDRV) Set Head and Drive Number for Disk Operations. This routine is entered with register A containing the Floppy unit number coded in bits 0 and 1 (Unit 0=00, 1=01 ..3=11), and the Head in Bit 2 (0=Head 0, 1=Head 1). Nothing is returned from this function. (Uses AF)

3 - (STSECT) Set Physical Sector Number, Size and Last Sector Number on the Track. On entry, Register A contains the desired physical sector number desired, D contains the sector size where 0=128 byte sectors, 1=256..3=1024, and E contains the last sector number on a side. Normally register E is unused in Western Digital controllers, but is needed with 765-compatible units. Nothing is returned from this function. (Uses AF)

4 - (SPEC) Set Step Rate and Head Load/Unload Time. On entry, the A register contains the drive step rate in milliseconds. Within the Bios, this rate is rounded up to the nearest slower controller rate if the specified rate is not an even match. Implementation of this function in the Bios also accommodates the need to adjust values reflecting the step rate when the data rates are changed from 250 to 500 kbps and vice versa. Register D should contain the desired Head Unload time in milliseconds, and E to the desired Head Load time in mS. With some controllers such as the Western Digital 177x and 179x family, only the Step Rate is universally variable. In these systems, rates signaled by the Bios settings are rounded up to the closest fixed step rate such as the 2, 3, 5, or 6 millisecond rates in the 1772 (YASBEC) or 6, 10, 20, or 30 millisecond rates used in the 1770 (Ampro Little Board) and 1795 (Compu/Time). Nothing is returned from this function. (Uses AF)

5 - (RECAL) - Recalibrate Drive (moves the head to track 0). There are no entry parameters for this function. On exit, the zero flag is Set and A=0 if no errors occurred, otherwise the Zero flag is Reset and A is Non-Zero. NOTE: This routine assumes that STHDRV, STSIZE and SPEC have been called first. (Uses AF)

6 - (SEEK) - Set the Track for disk operations and seek to it. On entry, Register A is set to the desired Track Number, D signifies whether Verification of the seek is required (D=0 for No Verify, D=0FFH if verifying), and E indicates whether or not to double-step (E <> 0 for Double-Step, E=0 for No Double-Step). On exit,

A=0 and the Zero Flag is Set (Z) if the operation was successfully completed while A <> 0 and the Zero Flag is cleared (NZ) if an error occurred. NOTE: This routine assumes that STHDRV, STMODE and SPEC have been called first. (Uses AF)

7 - (SREAD) - Read from the floppy disk. On entry, HL must point to a Buffer to receive the data read. It assumes that Mode, Track, Head/Drive, and Sector have been previously set. On exit, A=0 and the Zero flag is set (Z) if the sector was satisfactorily read, A <> 0 and the Zero flag is cleared (NZ) if an error occurred. (Uses AF and HL)

8 - (SWRITE) - Write to the floppy disk. On entry, HL must point to a Buffer from which to send the data. It assumes that Mode, Head/Drive, Track, and Sector have been previously set. On exit, A=0 and the Zero flag is set (Z) if the sector was successfully written, A <> 0 and the Zero flag is cleared (NZ) if an error occurred. (Uses AF and HL)

9 - (READID) - Read the first correct ID information on a track. There are no entry parameters for this function. The Zero flag is set (Z) and A=0 if no errors occurred, otherwise the Zero flag is Reset (NZ) and A is non-zero. NOTE: This routine assumes that STHDRV and STMODE have been called first. (Uses AF)

10 - (RETDST) - Return the status of a drive. There are no entry parameters for this function. On exit, A contains the raw unmasked status byte of the drive or the last operation depending on the controller type, BC contains the binary number representing the FDC controller type (e.g. 765, 9266, 1772, etc), and HL contains the address of the status byte returned in register A. NOTE: This routine assumes that STHDRV has already been called. (Uses AF, BC and HL)

11 - (FMTTRK) - Format a complete track on one side of a Floppy Disk. It assumes that the Mode, Head/Drive, Track, and Sector have already been set. On entry, HL points to data required by the controller to format a track. This varies between controllers, so RETDST should be called to determine controller type before setting up data structures. On entry, D must also contain the number of Sectors per Track, and E must contain the number of bytes to use for Gap 3 in the floppy format. On exit, A=0 and the Zero flag is Set (Z) if the operation was satisfactorily completed, A <> 0 and the Zero flag cleared (NZ) if errors occurred. (Uses all primary registers)

HARD DISK SUBFUNCTIONS:

These functions are available to directly access Hard Drives. To date, only drives connected by a SCSI type interface have been used, but the interface is generic enough to allow mapping to other interfaces such as IDE. The functions are accessed by loading the desired function number in the C register, loading a 2 (SCSI driver) into the B register and calling or jumping to Jump number 31 in the Bios entry jump table. Since this interface is not as standardized as Floppy functions in order to handle SASI as well as SCSI devices, the interface has only basic functions with the precise operations specified by the User in the Command Descriptor Block passed with Function 2. This places a greater burden on User programs, but it allows more flexibility to take advantage of changing features in the newer SCSI drives.

An additional constraint placed on the Hard Disk interface is the restriction to 512 byte physical sectors at the interface. Since most hard drives now default to this sector size, no difficulties have been reported in the few years since B/P Bios has been released, so this constraint should pose no serious limitation.

0 - Set User Data Area Address for Direct SCSI IO, Return number of bytes available for the SCSI Command Descriptor Block. On entry, DE must point to a 512 byte User Data Area to Send/Receive. This is mandatory since 512 bytes are always returned from a direct access due to the wide variety of controller types handled. On exit, A contains the number of bytes available in the Command Descriptor Block which will usually be 10, but may be scaled back to 6 in limited applications. (Uses AF and HL)

1 - Set Physical Device bit and store Logical Unit Number (LUN) in SCSI Command Block (Byte 1, bits 7-5) from byte in A. On entry, register A contains a bit-mapped byte indicating the SCSI Physical Unit number (bits 0-2) and Logical Unit Number (bits 5-7). On exit, A returns a "1" bit in the proper position for the SCSI physical drive unit with Bit 7 being the host computer, Bit 0 indicating Unit 0, etc. (Uses AF)

2 -Direct SCSI driver. This routine performs the function described by the command in the SCSI Command Descriptor Block addressed by DE. On entry, register A must also contain a flag signifying whether or not user data is to be written by this command (A=0 if No data to be written, FF if the address set with Function 0 contains user data to write). At the end of the function, 512 bytes are always transferred from the Bios IO Buffer to the User's Space set by Fcn 0. This may be inefficient, but was the only way we could accommodate the wide variety of different SASI/SCSI controllers within reasonable code constraints. Additionally, Register A contains 0 if the function performed with no errors, and 02H if a check condition was encountered. Also, Register L returns the unmasked Status byte (0FFH indicating a timeout error), and H returns the first Message byte received. (Uses all primary registers)

NOTE: This routine assumes the Command Block is properly configured for the type of Hard Disk Controller set in B/P Bios, and that the selected disk is properly described in the Bios Unit definitions (if necessary). Errors in phasing result in program exit and Warm Boot. It assumes the user has executed Functions 0 and 1 to set the data transfer source/destination address and logical/physical drive addresses.

In the final part of this series, we will describe some of the utilities developed and modified to support banked systems and address our ongoing efforts with the banked ZSDos2, Command Processor and future directions.

Real Computing

By Rick Rodman

32-Bit Systems

All Readers

Samba, Linux,Minix

Paragons of the Industry

"You have object-oriented, and you have true object-oriented," said Bill Gates, "and what we have is one level above that." Yes sir, with profound thinking like that showing the way, it's no wonder PCs are so easy to configure and expand. You complainers who can't get an internal modem to work under Windows 95 should be ashamed of yourselves.

With 95 and NT and Office sprawling over multi-gigabyte hard disks and squandering processor resources that exceeded those of the entire world a few years ago, and hardware that, when it works, is too complex for anyone to tinker with, many have felt that the days of the computer hobby are long gone. Well, I'm here to tell you that it ain't so. We're entering a new decade of vast riches available to all!

Samba

Available now is Samba, a package which is supposed to allow Unix computers to attach to PC resources in SMB LANs. SMB stands for Server Message Block, and refers to a Microsoft-designed protocol which runs atop NetBIOS (aka NetBEUI). (NetBIOS is an API, NetBEUI is a protocol.) NetBIOS itself is unroutable, because it broadcasts, so lately Microsoft usually implements SMB over NetBIOS over either Novell's SPX or TCP/IP. In the latter case, NetBIOS machine names are cataloged in what is called a WINS Server. WINS stands for Windows Internet Name Service, but it really has nothing to do with Windows.

As is usual with Microsoft, marketing gets to choose all product names; SMB networking has been called MS-NET, 3-Server-3, OpenNet, and many other names, and supported by software packages called LAN Server, LAN Manager, Windows NT Server, and so on. There are two flavors of authentication, usually called "workgroups" and "domains". "Workgroups" basically means client-based authentication, whereas "domains" use a very crude server-based authentication. Microsoft's various networking versions of Windows, including Windows 95, all support both methods. There is also a DOS client, which works fairly well, albeit slowly.

In fact, as you might guess from such a clumsily structured design, performance is, and has always been, poor - much worse than Novell's Netware. Yet NT appears to be outselling Netware these days; which goes to show you, mediocre products, low or giveaway prices, heavy-handed marketing, and a dollop of monopolistic underhandedness, is the recipe for market share. Technological superiority has little to do with it.

Samba is downloadable from various addresses, in both source and binary form. There's also 'smbfs', which is included in some Linux distributions, and 'smblib', a library for adding SMB functionality to your own applications. The main Samba site is: *http://lake.canberaa.edu.au/pub/samba/samba.html.*

For KPCF and most smaller establishments, ftp meets most file-transfer needs, and TCP/IP defines functionality for sharing printers as well. NFS can be used if mounted drives are required.

However, for those of you using Linux in a large office where NT and/or Windows 95 machines are the norm, Samba could be very valuable. You'll be able to set up shares on your machine that PCs can access, access shares on other folks' machines or their servers, and let other users print on your printer or print on their printers. If you can get WINE working too, you can even run their software. And, of course, a Linux machine can do much more than a 95 or NT machine can out of the box. Imagine that - Linux, corporate network citizen par excellence.

Linux Developer's Resource

New Linux releases are coming out all the time, like mammoth airdrops. My latest collection is InfoMagic's "Linux Developer's Resource", which contains 6 (SIX!) CD-ROMs, including several releases, including Slackware 3.0 and Red Hat, both based on Linux kernel 1.21.

Slackware 3.0 has a *greatly* improved installation program and installs very smoothly. Either a 1.2 or a 1.4 floppy can be used. Red Hat requires a 3.5/1.44MB floppy drive. Its installation is the best I've seen under *any* Linux implementation, and includes setting up X Windows and everything. It's not quite as flexible as Slackware's in dealing with weird hardware configurations, however.

Also, I got around to trying the "UMSDOS" method of using Linux. Basically, you create a subdirectory on a PC running DOS or OS/2, and unzip some files in there. Well, this is definitely the fastest way to install Linux - you can be up and running in minutes, rather than the hours needed for regular installs. The hardware used by your PC must be supported by Linux, of course. Unfortunately, the version sup-

plied seems to be different from the rest of the CD-ROM, so you can't install any additional packages without strange errors like "bash: /usr/bin/make: No such file or directory" - a lie, since the file exists; it's just the wrong version. Also, UMSDOS is not quite as flexible as you might think; you can't conveniently read DOS files in the same partition. But for speed of installation, it can't be beat.

Linux has really come a long way. If you had difficulties with Linux before, you ought to give it another try. Now that the popularity of the Internet is finally bringing about standardization of networking services and protocols, Linux seems poised to enter the mainstream of the computing world as a standard computing platform. Getting acquainted with Linux could be, not just fun, but a foresighted career move.

Minix 2.0 and ELKS

Minix 2.0 has been released, and is available for download at: *ftp://ftp.cs.vu.nl.* There are versions for the 8086 (XTs and 286s) and for the 386. Source code is available. Since apparently no book will be published, it has been rumored that there are some changes with respect to copyright restrictions on Minix. Minix 2.0 includes TCP/IP and supposedly supports new kinds of hardware, including Adaptec SCSI boards. I tested this briefly and was unable to get past the 'mkfs' command, which refused to recognize my hard disk controller (on a generic 286 AT machine).

ELKS (Embeddable Linux Kernel Subset) is a variation of Linux which is intended, at least initially, at a floppy-only 8088 configuration. It is available for download at: *ftp://ftp.ecs.solon.ac.uk/pub/elks.* This is a refreshingly simple system which fits on a couple of floppies, including source. I had no difficulty booting and running it. The designers are working hard to keep total system size as small as possible, so it will work in a floppy-only 640K system. Their goal is to make an embeddable kernel, usable in small controllers.

Minix on a Rainbow?

Jeff Weiner and I have been studying DEC Rainbow documentation with an eye to possibly getting Minix 1.5 or 2.0 running on it. The Rainbow is, of course, a much more sophisticated machine than your regular XT. It includes both a Z-80 and an 8088. Its floppy drive is controlled by the Z-80, but the hard disk by the 8088.

ELKS is another possibility, and Minix 2.0 is out, but Minix 1.5 is stable and well-documented, and we'll be able to tell when it's working. Anyone who wants in on the fun (?), please contact me (rickerr@erols.com) or Jeff (j_weiner@juno.com) by e-mail, or send a message to the TCJ mailing list and we'll get in touch with you.

There's a Rainbow set up at KPCF which is used for various CP/M work. It's really good for that, having utilities to read various disk formats (all single-sided, of course). Also, there's an optional graphics board using a NEC 7220 GDC which is supported by GSX. The Rainbow was one of the nicest CP/M machines made, and they're available very inexpensively at flea markets and hamfests.

In fact, there's a web site with lots of software for the Rainbow at: *ftp://ftp.update.uu.se.* Using a program called Teledisk (from our friends at Sydex), you can create images of Rainbow software to use on your machine. Using this technique, I made a disk set for Concurrent CP/M.

Other available 32-bit OSs

A couple of other 32-bit OSs you may have heard about are Microsoft Windows NT 4.0 and IBM OS/2 Warp 4. NT 4 has the Win95 "shell", which I'm not too crazy about, and includes a Web browser and server. It's a little buggy, but a couple of service packs have come out already. One problem that drives me nuts is its *hesitations* - 40 seconds or more whenever you run a 16-bit program, or sometimes when you exit programs, or sometimes for no apparent reason at all.

OS/2 Warp 4 is very stable and polished, and much easier to install than any previous version of OS/2. Its user interface is definitely different, but very consistent, and easier to use once you get used to it. And Warp 4 has a responsive feel to it - when you click an icon, things happen; there's no several-second wait as under NT. It runs much faster on a 486 than NT does on a Pentium.

Of course none of the OSs mentioned here have more than a tiny sliver of market share compared with Windows 95, but you can read about Windows 95 in other magazines.

The CP/M Web Page

While the folks at Caldera pursue their lawsuit against Microsoft (and they should prevail, given that Microsoft has already admitted wrongdoing), they have graciously made available much software from the Digital Research archives at a website maintained by a volunteer: *http://cdl.uta.edu/cpm.* You'll find source and binaries for CP/M 2.2 and CP/M-68K, binaries for CP/M Plus, MP/M 2 and CP/Net, and various assemblers and utilities. These are not disassemblies, these are the real, original, Digital Research source code. CP/M 2.2 are in 8080 assembly, with utilities in PL/M. CP/M-68K is written in C, for the Alcyon cross-compiler on a VAX. (CP/M-86 and PL/I are not there, since they are still being sold commercially.) You can download it and do whatever you like for non-commercial purposes. Great stuff!

The wealth of information and source code available on CD-ROM and on the Internet is truly mind-boggling. But always remember, it was hobbyists and experimenters, sharing with the community - not crass commercial monopolistic proprietary beasts like Microsoft - who built the Internet and made it what good it is. Corporations created only Net equivalents of telemarketers and junk mail and commercials.

Next time

More fun with operating system hacking at KPCF, including CP/M porting, Unix System V, portable networking, and more on the Real-Time Control System. HTTP/HTML as the universal network interface: your toaster's web page! All this and more!

The Unofficial CP/M Web Site

Special Feature

CP/M Source Code

Web Site

by Tim Olmstead

Over the years, one of the things I have most desired is to get the source code to the programs I use. This gave me a feeling of power. I can make changes if I want to, and make the program do something that its' designers hadn't envisioned, or simply fix problems. With most commercial programs, however, you will never get to look at the source code.

I knew of efforts to disassemble CP/M-80 as early as 1980, and everyone I knew had the source. But, this was certainly not something we could talk about openly. Digital Research kind of frowned on people disassembling their code, so we had to keep a low profile.

When, in mid 1996, Caldera bought out the rights to Digital Research software, I wondered if we would now get a look at the "real" source code. It seemed that we might. Caldera was releasing the source code for DR-DOS, renamed OpenDos, and indications were that they would also release the CP/M code.

Time passed, and we seemed no nearer to getting the "blessed" code than we were in 1980. I found this to be quite frustrating, and it caused me to do some thinking. The conclusion I came to was this. When one company buys out another company it is because they expect to get something they want. Usually, once they get what they want, anything else that came along for the ride, just gets discarded, or further sold off.

To make the CP/M software available, it would require that Caldera use resources; assign people to do it. Before most companies would do this, they would have to know how they were going to make some money from it. I suspect that their board of directors, and stockholders, might require it.

I worried that CP/M would at best get discarded, and hit upon an idea. How about if somebody were to do something user supported? This would only require that someone approve the idea, and forward the material received from Novell to whomever was coordinating the effort.

I had no idea at the time how little that would be. Much material has been lost over the years. There is very little actual source code left, and no manual source. We have source for CP/M 2.2, CP/M 68K, and CP/M 3.0, but not MP/M x.x, CP/M-86, or any of the compilers, assemblers, etc.

So, a couple of months after the announcement of the acquisition by Caldera, I began trying to make contact with someone inside Caldera so I could make my proposal. This took over six months of effort, and finally bore fruit by going through the webmaster of the "unofficial OpenDos" web page. Contact was finally made.

During the time when I was attempting to make contact, I reviewed the problems associated with the distribution of operating systems software via the Internet. Files were the least of the problems. They can easily be copied over to a PC using 22DISK, and PKZIP'd for distribution. The real problem is manuals, and boot disks. I decided to defer on the boot disk issue as Don Maslin provides a tremendous service in archiving boot disks. Anything I did here would only be redundant.

So, I turned my attention to the worst problem; that of how to distribute manuals for CP/M over the Internet. I consider this to be of equal importance to having the binary. A user will need the manuals in order create a usable system from any of the CP/M operating systems.

The only solution was to get the manuals into some form that could be transmitted electronically. This meant scanning, and OCRing them, to get them into some text editor format. I purchased a flatbed scanner, and some OCR software, and went to work. I quickly discovered that OCR has come a long way over the years, but, the kindest thing I could say was "It ain't there yet". The manuals took quite a bit of effort to get presentable.

Since this was to be a user supported effort, I solicited contributions from those who likely still had copies of Digital Research software laying around. I discovered that many of the users on comp.os.cpm still had this software, but had sentimental attachments to it and wouldn't part with it. Some I had to buy to get, while some people have sent me virtual mountains of stuff. I now have a queue of manuals thick enough to keep me busy for a year or two.

Finally, Caldera gave their approval to build the site. I contacted a friend and got onto a server at a local university; The University of Texas at Arlington. This is a very strong electrical engineering school, and the material made a good fit.

With final approval from Caldera, the "unofficial CP/M web site" went on-line. The name caused some confusion

Continued on Page 20

Simplex III

Part 4

by Dave Brooks

Special Feature

Advanced

TTL Computer

Simplex-III Debug Support

Previous articles in this series have described the architecture and programming of Simplex-III, a home designed CPU from the late 1970's. This article describes the facilities provided for debugging hardware and software.

Back in the '60s and '70s, the fancy debug software we have now was unknown, except perhaps on mainframes. Software was tested by stepping the hardware through the code. This required that every programmer-accessible register could be inspected (and preferably, modified) using monitor hardware. Such facilities were also necessary to debug the hardware. To test code in this way, it is necessary to be able to halt after each instruction. A logical extension, the ability to halt at micro-steps within an instruction cycle, permits the same mechanisms to be used to verify the hardware.

Table 1: Signals displayed on the front panel

MP[0:5]	Microprogram Pointer (ie box address)
IDLE	Machine is NOT running
LSB	Processing least-significant byte of data
INTLVL	Running at interrupt level
LC[0:2]	Microcode-box repeat counter
SPAR[0:3]	Scratchpad address register
R[0:7]	Anti-race register
I[0:7]	Current-instruction register
ADDR[0:15]	Store address register
L[0:2]	Effective accumulator length
Z	Zero condition bit
N	Negative condition bit
CA	Carry-out from last instruction
CB	Internal carry between bytes
SIG	Signal LED, controlled by program

Accordingly, Simplex-III includes a control/display panel, which can display any internal CPU register, and could halt execution on one of:

1. The next byte-step (ie one internal clock cycle)
2. Microcode box (ie process a multibyte operand, then stop)
3. End of the current instruction
4. Never (ie run free)

Normally, the "stop never" can only be selected when executing instructions, however an engineering-test switch permits the microcode read/write routines to be repeated indefinitely, as an aid to testing.

Simplex-III was designed to use dynamic RAMs. The architecture includes an automatic refresh function, which runs a refresh cycle on every bus cycle not required by the CPU (including when the CPU was halted). This feature is also used by the monitoring functions.

To reduce the number of lines needed to monitor the internal bits, 3 address bits are used as a scan counter. These bits are always cycling, due to the DRAM refresh function. Bits to be monitored are grouped in sets of 8, and multiplexed using the address bits. The multiplexer outputs are carried to the display panel, which includes a LED array, scanned by the same address lines. A total of 55 LED's are fitted, arranged in a 7 x 8 matrix. These signals are listed in Table 1.

The LED arrangement is shown in Table 2, the MON-X lines being the outputs of the various 8-bit multiplexers.

Table 2: Multiplexed front-panel LED array

ADDR11\	L	L	L	L	H	H	H	H
ADDR12\	L	L	H	H	L	L	H	H
ADDR13\	L	H	L	H	L	H	L	H
MON-A	MP0	MP1	MP2	MP3	MP4	IDLE	LSB	INTLVL
MON-B	LC0	LC1	LC2	–	SPAR0	SPAR1	SPAR2	SPAR3
MON-C	R0	R1	R2	R3	R4	R5	R6	R7
MON-D	I0	I1	I2	I3	I4	I5	I6	I7
MON-E	ADDR0	ADDR1	ADDR2	ADDR3	ADDR4	ADDR5	ADDR6	ADDR7
MON-F	ADDR8	ADDR9	ADDR10	ADDR11	ADDR12	ADDR13	ADDR14	ADDR15
MON-G	L0	L1	L2	Z	N	CA	CB	SIG

Panel Operations

Microcode operation is controlled by two panel switches: Function and Halt. Function controls a multi-way branch taken by the microcode, to execute instructions, or to read and write scratchpad or memory locations. Halt determines when the microprogram is to stop, and return control to the front panel. The required function is then executed when the "Start" button is pressed. The "Function" operations are:

Restart

IPL

Examine register

Load register

Examine memory

Load memory

Execute instruction

The register-scratchpad and main memory are examined and altered by running short sequences of microcode, which move data between the display and the stored locations. When manually accessing the scratchpad, the required address is set up on panel switches. The Function switch is set to Examine or Load memory or scratchpad, and data (if required) is set up on panel switches. When the Start button is pressed, data is copied from the location to the R register, which is displayed on the panel. For write operations, data is copied from the panel switches to the location. For memory operations, the Store Address register auto-decrements after every Examine or Store operation, so that a sequence of locations may be read or written rapidly. The decrementing action is consistent with Simplex-III's big-endian organisation.

The "Restart" function simply sets the S and SA registers to -1. Since S is increased by 2 before the next instruction is fetched, the first instruction will be fetched from memory locations 1 and 0.

The "IPL" setting loads the value 1F into SA, and proceeds to copy 32 bytes from the IPL ROM into memory, filling downward to location 0. S is then set to -1 (as for Restart), in readiness to execute the just-loaded instructions. Typically, these would bootstrap in a larger program using the paper-tape reader.

When manually accessing the scratchpad, the "Sense" switch determines whether the base or interrupt-level registers are accessed. Effectively, it acts as a 5th address bit. In normal running, "Sense" may be tested by a conditional branch instruction, and provides a low-level data input to programs.

Next

The following articles will descend into detailed hardware operations. The microcode will be described, followed by a quick tour of the logic circuits, including a simple peripheral card. Part 5 starts on page 21.

Continued from Page 18

in the beginning. This is what Caldera wanted me to call it. They planned to put up an official web page, so I was to call mine unofficial. They did put up some material, and a pointer to my site.

Given the origin of this site, my goal is this; to put together a collection that is as complete as possible of all Digital Research produced software. There have been a number of things show up that I had no idea that Digital Research had done. I say "great", I'll add them to the collection.

Many people have offered me other software that runs on CP/M, such as Microsoft Basic. These programs, while they would make a good addition to the collection, can't be posted because I don't have the distribution rights to them. I have attempted to contact several of the copyright holders for these packages, such as Borland, and Microsoft, and usually never even get an answer to my email. Oh well. We will proceed with the Digital Research software.

There is one dark spot on the project though. I have been specifically prohibited from posting anything in the Concurrent product line. This includes MP/M-86, Concurrent CP/M-86, and Concurrent DOS. Caldera has sold the rights to this family of products. There are, I believe, three companies that have been actively involved in enhancing these products over the years, and have significant interest in continuing their efforts. They are the ones who have bought the rights to this family of products.

I have noticed quite a bit of confusion as to the exact status of the Digital Research software. I have heard many comments like this ; "Since we can now get CP/M for free, and it is in the public domain, we can do whatever we want with it". This is not true. CP/M still remains commercial software. There is a license agreement on the unofficial CP/M web site that all users should read, and understand. I know this is tough as it was written by lawyers, not humans, but try. What it says is, "You can have the software for free, with source code, for PERSONAL, NON-PROFIT use". For commercial use, Caldera will still sell licensees. I suspect that these will be favorably priced. The point is that CP/M is not freeware, nor is it public domain. It is still commercial software.

That said, let's keep checking those closets. I want to make the collection as complete as possible. I just got in a new OCR package, and I'm anxious to see if it will do any better job than what I've been using. Thanks to all that have made contributions to help make this a great site. The "unofficial CP/M web site" is unique. To my knowledge it is the only site that has been authorized to distribute Digital Research software.

The last time I checked the Caldera site, I couldn't find any CP/M files. That's OK. The user response to the unofficial CP/M web site has been tremendous. I have had many email messages, mostly positive.

Simplex III

Part 5

by Dave Brooks

Special Feature

Advanced

TTL Computer

Simplex-III Internal Details

This is the 5th in a series of articles which describe a home-designed CPU, built in the late 1970's from TTL logic parts. Previous articles have described the development of the project, and have examined the machine from a programmer's viewpoint. The final 3 articles look at the construction and the low-level circuitry.

Microprogram

The microprogram is described in pseudo-code, in Table 2. Each "box" or ROM address is labelled as "Boxnn", and also defines "Rpt", the number of times the box is repeated. A repeat-value of "L" denotes the content of the L register if the operand register is A, and 2 otherwise.

Table 3 shows the signals emitted from the microcode ROM array. The ROM contents are listed in Table 1.

Implementation

While it is not expected that anyone would want to build a copy of Simplex-III, some of the practical details may be of interest.

Mechanical Construction

The CPU proper is built on 4 sheets of stripboard, each about 150 x 150mm, having a 60-way edge connector. All connections are hand-wired, as it was not then practical to do one-off printed boards (also, the design went through several iterations and refinements). The first 3 cards are also linked by an additional connector, plugged in on the top edges (60 pins was not quite enough).

The boards are fitted in a card-cage, with the LED display board at one end. This end is placed behind a transparent panel, for viewing. The 4 boards are assigned as follows:

1. LED display, interrupt logic, byte counter and scratchpad addressing.
2. Main data paths, or "mill".
3. Microprogram ROMs, function switch interfacing & support.
4. Clock logic, including single stepping.

Additional slots are provided behind the CPU group, for memory and IO cards.

The main panel controls are located to the left of the card-cage, with the power supply and cooling fan behind.

Circuit Description

Simplex-III was built almost 20 years ago, and many parts are now obsolete (some were obsolete even then). The original schematics were hand-drawn. The present schematics have been digitised from the original drawings. Many "shorthand" devices have been employed, eg letting one component stand for many in a bus. Hence the schematics are not electrically precise: their object is to show how things worked, rather than to be suitable for compilation into netlists. In particular, component reference designators are meaningless in terms of how gates are grouped: they are purely for reference from the text.

Card 1 SCHEMATICS:

The register pair at E1/E2 is the L (accumulator length); the pair at C1/C2 are the condition codes. These registers are duplicated for interrupt and base levels. The appropriate level is selected by the REGSELA\ and REGSELB\ signals. The condition codes are selected by the multiplexer A for a conditional jump.

The decoder at F5 drives the LED array. The monostable E4 is a safety device: the LEDs are driven with short heavy current pulses, which would cause damage were the scanning function to cease. Should this happen, the monostable will time out, and inhibit the decoder outputs.

The 8-bit register at D6 is the current-instruction ("I") register.

The counter at C3 is the box-repeat counter. The number of repeats is controlled by the multiplexer at B3. This multiplexer inverts its outputs, so that the C3 counter counts upward until it reaches a count of 15. This asserts the TC output, and advances the microcode.

The group at C5, B5, D4, E6 source the scratchpad address, which is counted at D4.

Next issue

The next article continues the hardware tour, with the main data paths and the microprogram logic.

TABLE 1: Microcode ROM content

Addr. Pin	Chip C1 12345679	Chip D1 12345679
00	HLLHLHLL	LHHLLLHH
01	LHLHHLLL	LLLLLLLH
02	HLLHLHLL	LHLLHLLL
03	LLLLHLLH	LLLHHHLL
04	LHLHHLLH	LLHHHHHL
05		
06	LHLHHLLH	HLLHLLHL
07	LHLHHLLL	LHLLLLHH
08	LLLLHLLL	HLLHLLLL
09	LLHHHLLH	LHHLLLLL
0A	LHLHHLLH	LLHLLLLH
0B		
0C	LHLHHLLH	LLLHLLLH
0D		
0E		
0F		
10	HLLHLHHH	LLHLLHLH
11	LLLLHLLH	HLLLLHLL
12	LHLHHLHH	LLHLLHLH
13	HLLHHLHH	LLHLLHLH
14	LHLLHLHH	LLHLLHLH
15	LHLLHLHH	LLLLLHLH
16	HLLHLHHH	HLLLLHLH
17	LHHLLLHH	LLHLLHLH
18	LHHLLLHH	LLLLLHLH
19	LLLLHLLL	LLHLLHLL
1A	HLLHLLHH	HLLHLLLH
1B	LLLLLLLH	LLLHLLLL
1C	HHLLLHHH	LLHLLHLL
1D	HLLHLHLH	LLHLLLLL
1E	LHLHHLHH	LLHLLHLL
1F	HLLHLHHH	LLHLLHLL

TABLE 2: Microcode logic

```
In the following table, all numbers are in
hexadecimal. The instruction fields are defined as:
      opc     = I[0:3]
      Ro      = I[4:5]
      Rx      = I[6:7]
C = Condition-code bits (N, Z, CA)
S = 0 (locn. of "S" register in scratchpad)
The Mode Switch defines the values KV and M as
follows:
      Mode                        KV      M
      Execute instruction          2      1
      Examine SPAD                 0      3
      Load SPAD                    0      4
      Examine memory               1      C
      Load memory                  1      6
      Restart                      0      9
      IPL                         1F      7
Cold_start:
Box00: Rpt=2 //Preindex "S", goto microroutine
      SPAD[S++] = SA = SPAD[S] + KV
      goto Box[M]
Box01: Rpt=2                  //Fetch instruction
      I = R = mem[SA-]        //LSB in I, MSB in R
                              //  when done
      if (opc >= 0C) goto Box[10 + opc]
            else goto Box02
Box02: Rpt=2                  //Compute operand address
      SA = R + SPAD[Rx]
      goto Box[10 + opc]
Box03: Rpt=1                  //Examine SPAD
      R = SPAD[switches]
      goto Box00
Box04: Rpt=1                  //Load SPAD
      SPAD[switches] = R = Panel_keys
      goto Box00
Box0C: Rpt=1                  //Examine memory
      R = mem[SA-]
      goto Box00
Box06: Rpt=1                  //Load memory
      mem[SA-] = R = Panel_keys
      goto Box00
Box07: Rpt=2                  //IPL: set up SA
      SA = 1F
      goto Box08
Box08: Rpt=(until SA=0)  //IPL: copy to memory
      mem[SA-] = R = IPL[SA]
      goto Box09
Box09: Rpt=2 //Reset S to start execution
      SA = S = -1
      Set_Interrupt_level
      goto Box00
Box10: Rpt=L //ADD instruction
      SPAD[Ro++] = C = SPAD[Ro] + mem[SA-]
      goto Box00
Box11: Rpt=L //STORE instruction
      mem[SA-] = SPAD[Ro++]
      goto Box00
Box12: Rpt=L //LOAD instruction
      SPAD[Ro++] = C = mem[SA-]
      goto Box00
Box13: Rpt=L //XOR instruction
      SPAD[Ro++] = C = SPAD[Ro] xor mem[SA-]
      goto Box00
Box14: Rpt=L //AND instruction
      SPAD[Ro++] = C = SPAD[Ro] and mem[SA-]
      goto Box00
Box15: Rpt=L //LOGICAL COMPARE (TEST) instruction
      C = SPAD[Ro++] and mem[SA-]
      goto Box00
Box16: Rpt=L //ADD TO STORE instruction
      mem[SA-] = C = SPAD[Ro++] + mem[SA]
      goto Box00
Box17: Rpt=L //SUBTRACT instruction
      SPAD[Ro++] = C = SPAD[Ro] - mem[SA-]
      goto Box00
Box18: Rpt=L //COMPARE instruction
      C = SPAD[Ro++] - mem[SA-]
      goto Box00
Box19: Rpt=2 //Jump Indirect (Link)
      if (Ro != A) SPAD[Ro++] = SPAD[S++]
            //Saves link
      goto Box0A
Box0A: Rpt=2                  //Indirect jump
      SPAD[S++] = mem[SA-]
      goto Box00
Box1A: Rpt=1 //COUNT IN STORE instruction
      C = mem[SA-]++
      goto Box00
Box1B: Rpt=1                  //Set flags
      if (I[4] == 0) L = I[5:7]
      else {
            Intlevel = I[5]
            Signal[J,K] = I[6:7]
            }
      goto Box00
Box1C: Rpt=L                  //SHIFT LEFT instruction
      SPAD[Ro] = C = SPAD[Ro] + SPAD[Ro]
      goto Box00
Box1D: Rpt=2 //BRANCH CONDITIONAL instruction
      if (condition) {
            if I[7] SPAD[S] += R
            else    SPAD[S] -= R
            }
      goto Box00
Box1E: Rpt=L //LOAD IMMEDIATE instruction
      SPAD[RO++] = C = R
      goto Box00
Box1F: Rpt=2 //MOVE/INCREMENT REGISTER instruction
      SPAD[Rx] = C = SPAD[Ro] + R
      goto Box00
```

TABLE 3: Microcode ROM outputs & their functions

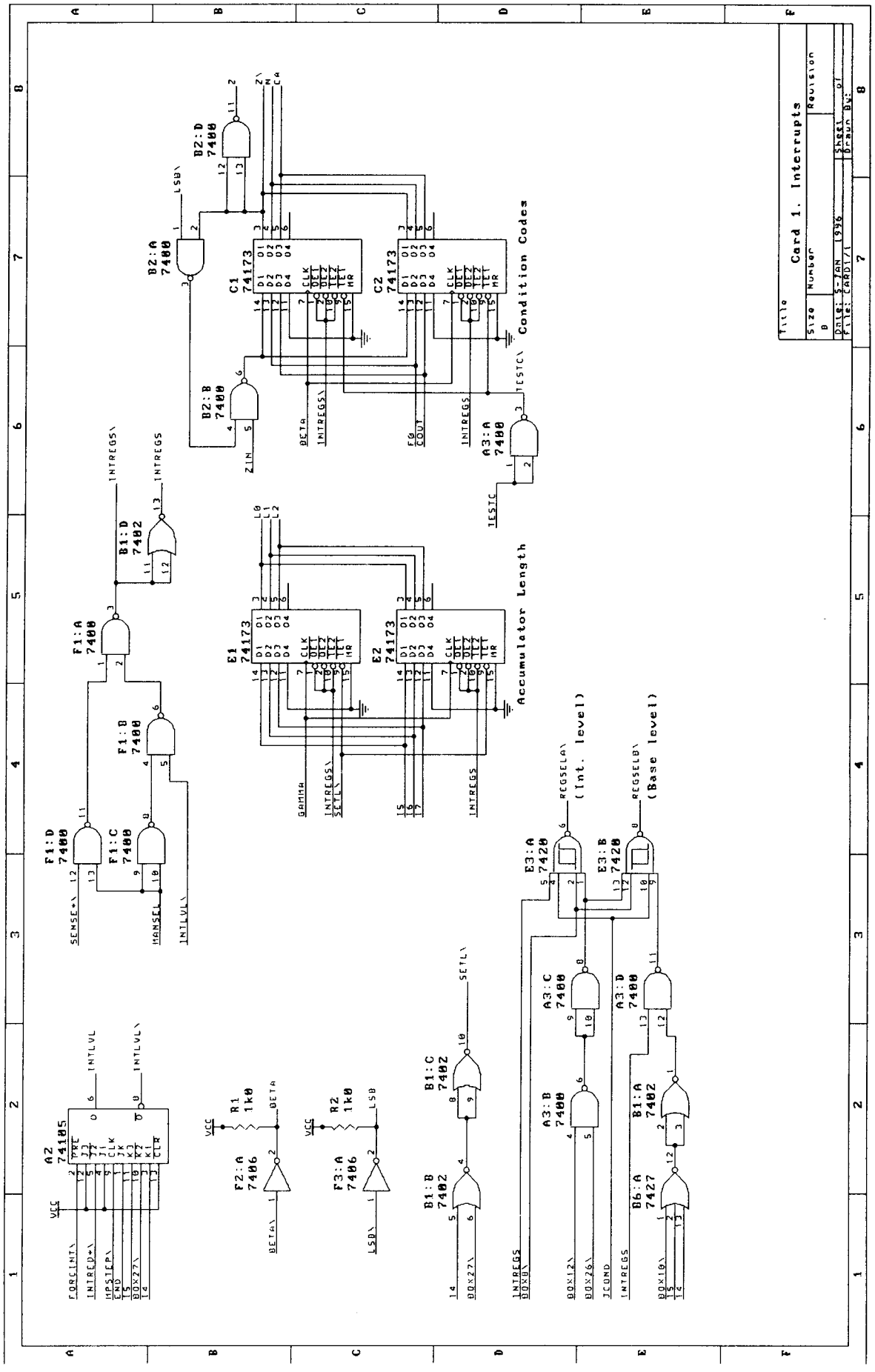
S[0:3]\	Determine the function executed by the ALU
M	Selects the arithmetic or logical ALU functions
CIN\	Carry-in to first byte of the box
TSTC	Condition bits are to set from this operation
END	This box is the last of an instruction
OUTSEL	Data to be driven to the IO bus
LOAD SA	Load result into Store Address reg.
LOAD SPAD	Load result into scratchpad register
BYTE	This is a 1-byte operation (no repeat)
SPD[0:1]	Selects scratchpad address source
BUS[0:1]	Selects source for tristate bus

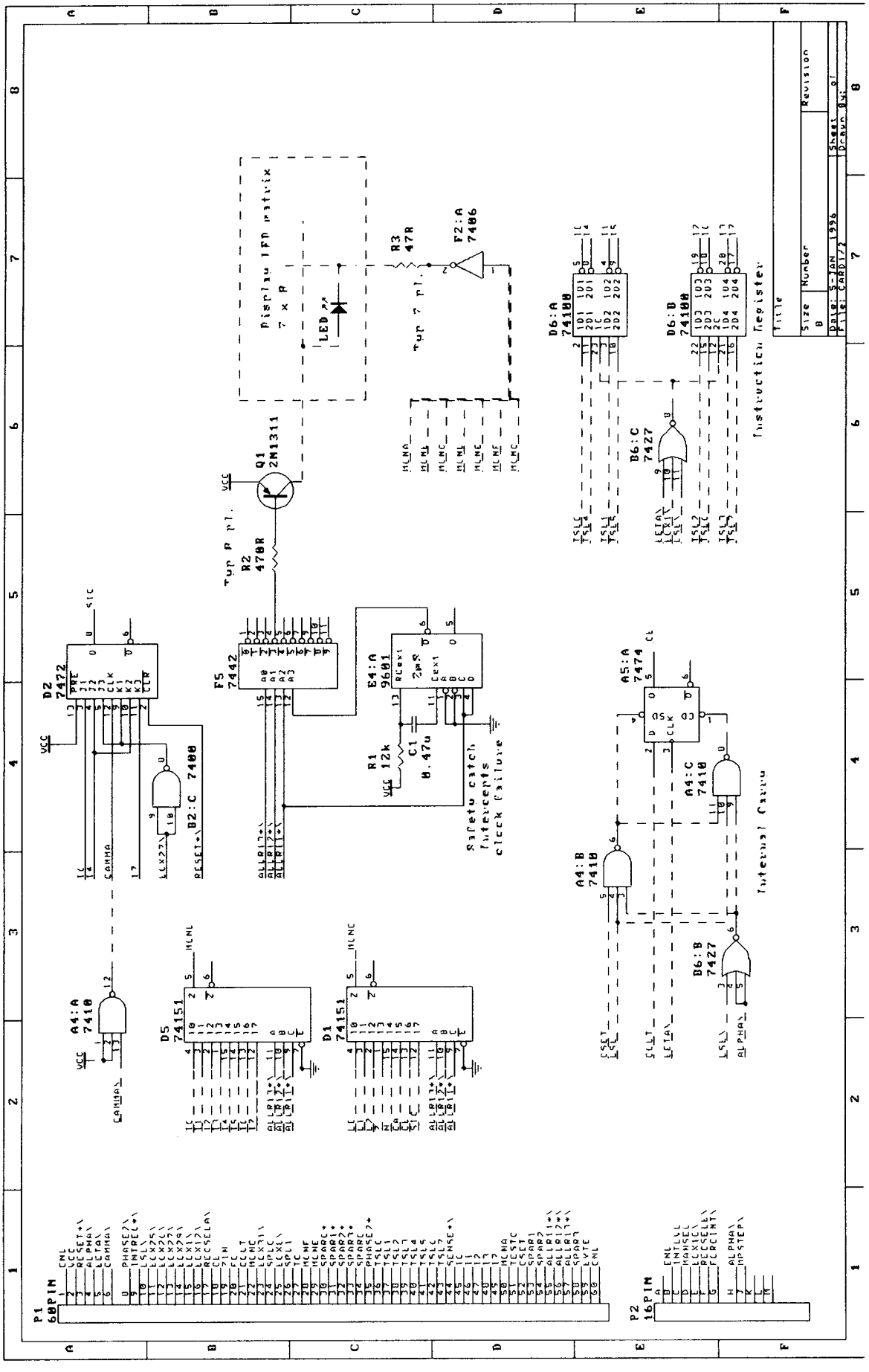
ROM Loc. C1

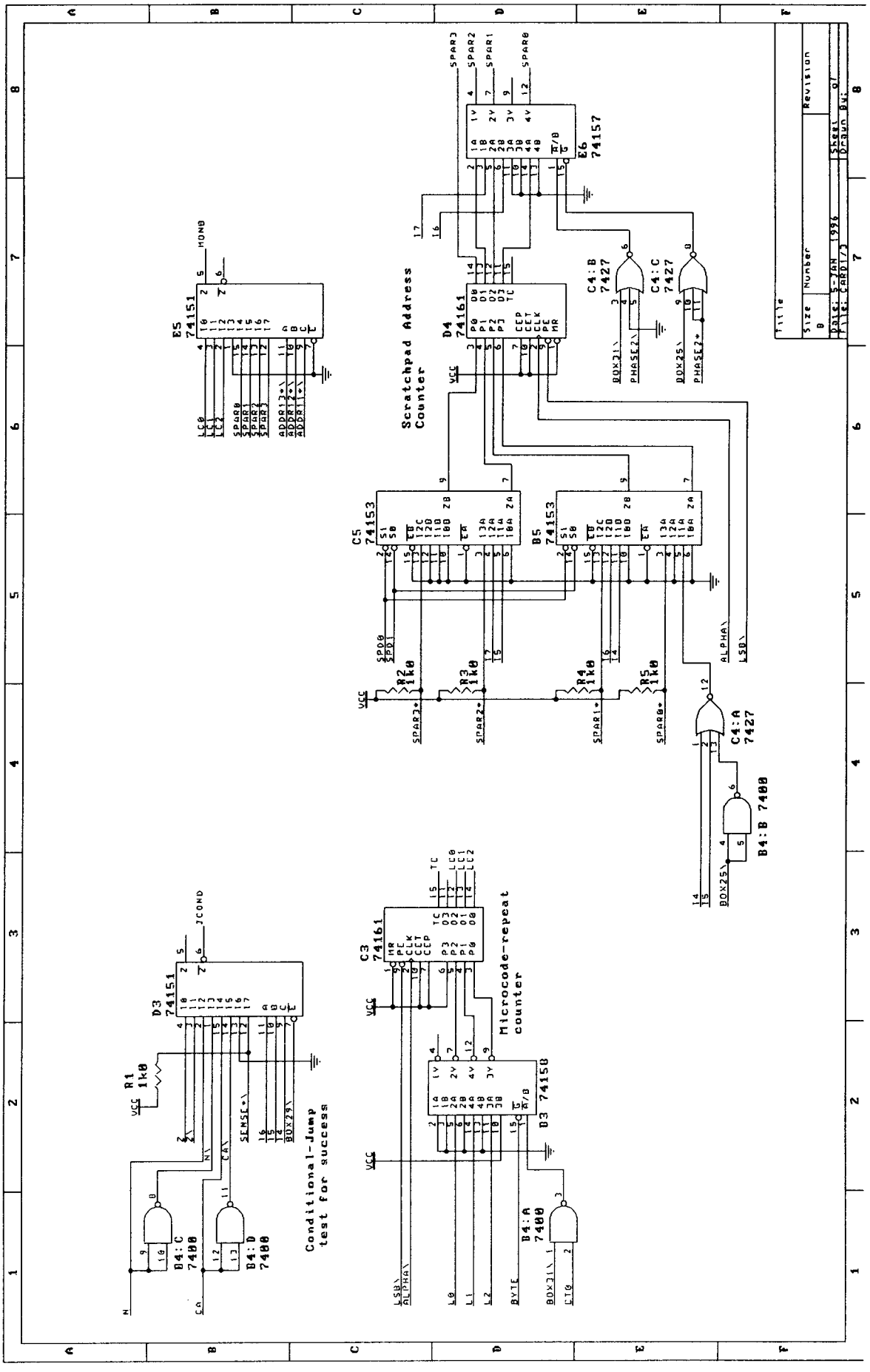
1 S0\	2 S1\	3 S2\	4 S3\	5 M	6 CIN\	7 TSTC	9 END
L	L	L	L	H	L	- SPAD	
L	H	L	H	H	L	- TSB	
H	L	L	H	L	H	- SPAD + TSB	
L	L	H	H	H	L	- -1	
H	L	L	H	H	L	- SPAD xor TSB	
L	H	L	L	H	L	- SPAD and TSB	
L	H	H	L	L	L	- SPAD - TSB	
H	H	L	L	L	H	- SPAD + SPAD (= sl 1)	
H	H	L	L	H	L	- 0	

ROM Loc. D1

1 OUT SEL	2 LOAD SA	3 LOAD SPAD	4 BYTE	5 SPD0	6 SPD1	7 BUS0	9 BUS1
				L	L	- S	
				L	H	- Rop	
				H	L	- Rx	
				H	H	- Keys	
						L	L - R
						L	H - Store
						H	L - Keys
						H	H - KVconst


Card 1. Interrupts
Condition Codes
Accumulator Length
(Int. level)
(Base level)


Instruction Register
Internal Carry
Safety catch
Interrupts
clock failure
Display LED matrix
7 x 8
LED
Turn ? pl.
Turn P pl.
Q1
2N1311
R1
12k
C1
0.47u
R2
470R
R3
47R
P1
60PIN
P2
16PIN
F5
7442
D2
7422
D5
74151
D1
74151
E4:A
9601
A4:A
7410
A4:B
7410
A4:C
7410
A5:A
7474
B6:B
7427
B6:C
7427
D6:A
74100
D6:B
74100
F2:A
7406
Size
B
Number
Revision
Date: 5-JAN 1996
File: CARD1/2
Sheet
Drawn By:
Title


Scratchpad Address Counter
Microcode-repeat counter
Conditional-Jump test for success
E5 74151
D4 74161
E6 74157
C5 74153
B5 74153
C3 74161
D3 74151
D3 74150
C4:B 7427
C4:C 7427
C4:A 7427
B4:B 7400
B4:C 7400
B4:D 7400
B4:A 7400
R1 1kΩ
R2 1kΩ
R3 1kΩ
R4 1kΩ
R5 1kΩ
Title
Number
Revision
Size
B
Date: 5-JAN 1996
Sheet
File: CARD1.3
Drawn By:

High-Density FDC for YASBEC

Special Feature

YASBEC Mod

New Disk Controller

by Harold F. Bower

Paul Chidley's excellent YASBEC (Yet Another Single-Board Eight-bit Computer) has proven to be remarkably versatile, and suffers few weaknesses. The most desirable feature was addressed by Jim Thale in 1994 when he produced the YASMIO expansion that added a "High-Density" floppy diskette controller (which can also handle 8" drives) as well as additional serial and parallel ports. Having worked closely with Jim to build the prototypes as well as write the drivers for the Banked and Portable (B/P) Bios, it was natural to draw on this experience when I built a YASBEC-based laptop a year ago. The circuit modifications in this article might also be easily adaptable to other computers.

Since I could not afford the space (or power) for a YASMIO in the laptop, a circuit modification to the YASBEC was the course of action. Borrowing on the circuitry from the YASMIO, I used the National Semiconductor DP8473 as the core since it does nearly everything required in one chip. While the part has been discontinued by National, it is still available from Digi-Key as of this writing. In addition to this chip, only a 24 MHz Monolithic oscillator, a few resistors, capacitors, a piece of perfboard and some pins are required for the entire modification. As a further benefit, several parts may be removed from the YASBEC since the 8473 includes the drivers on-board!

One of the main constraints going in to this project was to make minimal trace cuts and other modifications to the basic YASBEC. Since the fundamental goal is to replace the Western Digital 1772 controller (U14), any signals appearing on that chip are assumed to be available by plugging into the socket when the chip is removed. This furnishes us with power, 8 data lines, Address lines A0-A2, and Chip select for addresses 68-6FH. Also available are several lines needed for the Floppy cable connector. Missing from the needed signals are Read and Write (active low), and a Reset signal of the proper polarity. These three signals are, however, available from a socket adjacent to the 1772 which was designed for a 8231A Math Coprocessor (U13). In addition, more power and ground pins are available to provide better noise performance. Six additional signals are still needed, but they can be handled by a short length of ribbon cable and a home-made plug made from in-line machined-pin socket strips.

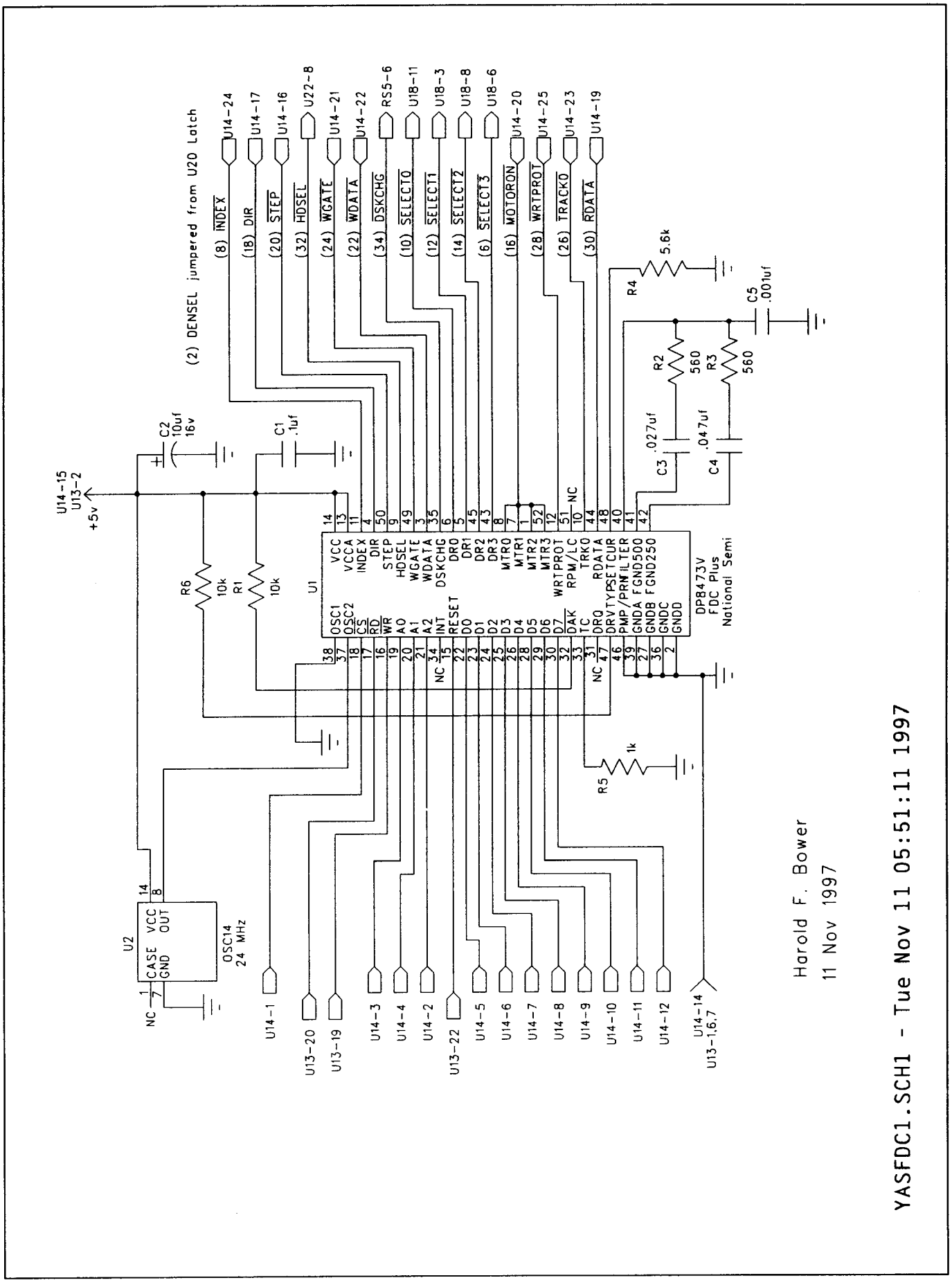
With the sockets for U13 and U14 available, we create a footprint for the modification. The prototype, as well as a duplicate to prove the design for this article are based on a piece of "pad-per-hole" perfboard measuring 1.5" x 3.0". Connections between this board and the YASBEC sockets are via pins which plug into the sockets and into pad holes on the perfboard. Since the two sockets are not exact multiples of 0.1" apart, some mis-alignment exists, but not to an excessive degree.

Before you begin this project, be sure you have any needed files on a bootable hard disk. With "Murphy" alive and well, you should prepare a second hard drive as well, since you will not be able to boot from floppy diskettes from the time you begin this project until you have the modification working and modify the Boot ROM to recognize the new circuitry. For the bold of heart, charge on...

The first step is to remove all unnecessary parts from the YASBEC. Remove the 8231A (U13) and FD1772 (U14) from their sockets. Carefully unsolder and remove the 8 MHz Monolithic oscillator (U24), the 74HC74 surface-mount IC from the bottom of the board (U23), and the two surface-mount drivers; 74AC04 (U22) and 74AC00 (U18) from the top of the board next to the 34-pin DIP header. Care in removal of these latter two chips is essential since you will need to solder wires to many of the pads for these ICs.

Next, solder short "U"-shaped pieces of wire-wrap wire across some of the pads where U22 formerly resided to connect signals from the DIP header directly to pins on the U14 socket. Five such jumpers are required to connect pins 1-2 (MOTOR_ON), 3-4 (STEP), 5-6 (WRITE_GATE), 10-11 (WRITE_DATA) and 12-13 (DIR_SEL). On the bottom of the board, solder a short jumper wire between pin 26 of U14 to Pin 2 of the 34-pin DIP floppy cable connector. This lead will be used to command floppy drives to switch between "High-Density" and normal MFM by latching a "1" or "0" in bit 6 of the 74HC273 at U20. This is the only bit of that chip used in the modified design, since the other six bits which were needed by the original 1772 are no longer needed. Note that wiring a TTL/CMOS level signal such as this directly to the floppy cable violates the normal open-collector/open-drain drive, but works in practice if the cable lengths are short, and you are careful not to short the pin or tie it directly to +5 volts. Set the YASBEC aside for the time being. We will return to it later to wire a small cable to pick up six more signals to/from the floppy connector.

Gather all parts needed to construct the small adapter board. Component selection is relatively non-critical except for those used in the analog data filters. Contrary to the filter


(2) DENSEL jumpered from U20 Latch
(8) INDEX U14-24
(18) DIR U14-17
(20) STEP U14-16
(32) HDSEL U22-8
(24) WGATE U14-21
(22) WDATA U14-22
(34) DSKCHG RS5-6
(10) SELECT0 U18-11
(12) SELECT1 U18-3
(14) SELECT2 U18-8
(6) SELECT3 U18-6
(16) MOTORON U14-20
(28) WRTPROT U14-25
(26) TRACK0 U14-23
(30) RDATA U14-19
U14-15
U13-2
+5v
C2 10uf 16v
C1 .1uf
R6 10k
R1 10k
U1
VCC VCCA INDEX DIR STEP HDSEL WGATE WDATA DSKCHG DR0 DR1 DR2 DR3 MTR0 MTR1 MTR2 MTR3 WRTPROT RPM/LC TRK0 RDATA DRVTYPE SETCUR PMP/PRM FILTER GNDA FGND500 GNDB FGND250 GNDC GNDD
OSC1 OSC2 CS RD WR A0 A1 A2 INT RESET D0 D1 D2 D3 D4 D5 D6 D7 DAK TC DRQ
DP8473V FDC Plus National Semi
R4 5.6k
C5 .001uf
R2 560
R3 560
C3 .027uf
C4 .047uf
R5 1k
U2
CASE VCC OUT GND
OSC14 24 MHz
NC
U14-1
U13-20
U13-19
U14-3
U14-4
U14-2
U13-22
U14-5
U14-6
U14-7
U14-8
U14-9
U14-10
U14-11
U14-12
U14-14
U13-1,6,7
Harold F. Bower
11 Nov 1997
YASFDC1.SCH1 - Tue Nov 11 05:51:11 1997

design used in the YASMIO which accommodated rates through 1000 kbps, this circuit includes only the 250 and 500 kbps capability. I have had best results using either a mylar or Silver Mica capacitor for the .001 microfarad unit, and mylar for the .027 and .047 microfarad items. Careful selection of these capacitors and the two 560 ohm resistors will provide the most reliable data recovery.

Finally, wire the adapter according to the schematic. The six leads shown on the schematic extended slightly to the right are the leads which will be wired to the extender cable and should be terminated on the adapter in six pin female header. Perhaps the easiest way to begin is to place the male pins in the respective sockets on the YASBEC, then place the perfboard on the pins in the desired position. Tack-solder a few pins in each row to retain the positioning and carefully pry the board with pin strips out of the YASBEC sockets. Next, position the 52-pin PLCC socket for the DP8473 and solder the unused pins to hold the socket in place. Finally, wire the remaining components using wire-wrap wire for connections.

When the board is fully wired, return to the YASBEC. Five of the six external connections will be made to the pads remaining after removal of U18 and U22, but one (DSKCHG) must be made from the bottom of the board. This may be made by drilling a small hole near pin 32 on the floppy connector. Hold the YASBEC up to a bright light and locate a free spot with no trace on either side of the PCB. Mark the spot and drill with a 1/16" drill or smaller. Feed a small stranded wire through the hole and solder it to the resistor lead which is wired to a trace running only to pin 34 of the Floppy Drive connector. In the original circuit, this was the /READY signal and was simply pulled high. The new circuit uses it for the /DSKCHG signal which is typically unused by the software, but available. Next connect small flexible stranded wires to the following pads for removed ICs: U22 Pin 8 (SIDE), U18 Pins 3, 6, 11, and 8 (/SELECT0-/SELECT3). A piece of flexible ribbon cable may be used if much care is used to prevent stress from pulling the pad traces from the PCB.

After the wires are installed, install the adapter board by plugging it into the sockets, and route the wires to the header on the adapter. Insert a 6-pin male strip into the header, and solder the wires to the respective pins. This method allows the board to be removed if necessary rather than fixing it into position by soldering it in place.

Now comes the hard part..the software! If you heeded the introductory warnings about preparing a bootable hard drive, you should be able to still use the computer by booting from the hard disk, but the floppy disks will be useless until a modified Bios is prepared. Anyone proceeding this far should have the knowledge to perform this, so we will concentrate on the requirements for booting from Floppy Disks, or the code that must be burned into ROM. Rather than specifically addressing the YASBEC ROM, we will generically cover code fragments suitable for integrating in any form.

It might have been simpler to extend the DENS signal from the 8473 directly to pin 2, but that would have required all diskettes to conform explicitly to the IBM-PC conventions for Hi/Lo density and speed. Maintaining this signal under your control by latching it with the 'LS273 provides the option of handling older drives which may not be so "standardized". The software fragments listed here do correspond to 3.5" and 5.25" drives strapped just as they came from a clone. The primary routine is the one which attempts to identify the drive and media type. As listed, it can take many seconds to sort out the correct format due to the many combinations. If you restrict the choice of drive to a particular type, then you can eliminate the other alternatives and significantly reduce the time needed to identify the drive/format.

Assuming you have a bootable machine (recall the hard drive warning?), you might want to modify a BIOS based on these fragments and test the hardware adapter, then integrate the fragments into the boot ROM. In this way, you can become familiar with the nuances of the circuit and its capabilities. A replacement Floppy Module for B/P Bios is available to B/P Bios owners for the asking (E-Mail HalBower@worldnet.att.net).

MULTITASKING 8051 CAMELFORTH

Brad Rodriguez,
T-Recursive Technology

Special Feature
CamelForth
8051 Multitasking

This article describes a multitasking extension to the 8051 CamelForth system (described in TCJ #71 and #72). The techniques described here are not limited to Forth; they should be useful to any 8051 assembly-language programmer.

CONTEXT SWITCHING

When several tasks are sharing a processor, the information which is specific to any one task is called that task's "context." This includes the current instruction pointer, CPU registers, subroutine return stack, and any other private data. CamelForth uses the 8051 registers as follows:

The registers labelled "scratch" may be used within Forth words, but have no guaranteed value between Forth words (except that the register select bits in PSW must always be 00 in 8051 CamelForth).

The 8051 stack pointer SP points to the last byte actually stored in the Return stack. This stack grows from internal RAM location (register) 09 upward to 7F.

In CamelForth, each task's private data is contained within a 768-byte "task area," organized as in Figure 2.

All of this data is addressed relative to a pointer, UAREA, whose high byte is stored in register 8. (The Task Area is page-aligned, so the low byte of this pointer is always 00.) To make another Task Area current, only this pointer need be changed. Note that the start of the Task Area is 100h bytes *before* UAREA.

So, to switch the from one task's context to another, the following steps must be taken:

1. Save the "important" registers and the Return stack in external RAM, with the task's private data. This is is purpose of the "Task Save Area" in Figure 2.
2. Switch the UAREA pointer to the new task's User Variables area.
3. Restore the previously-saved registers and Return stack from the new task's Task Save Area.
4. Resume execution.

The code to do this is shown in Listing 1, as a Forth word SWITCH. This word expects the address of the new task's Task Save Area on top of stack (i.e., in DPTR). Obviously, copying the Return stack out to the Task Save Area is the time-consuming step, so SWITCH includes a tight loop to copy registers N to 1 to external RAM, using R0 as the loop counter. The number of the highest register to be saved is contained in the Return stack pointer SP; this number is saved in the first byte of the Task Save Area.

Since SWITCH is called as a subroutine, the program counter will already be pushed on the return stack (and thus saved by the copy loop). Register R0 (the Data stack pointer) is saved by copying it to R1 before the loop. DPTR doesn't need to be saved, since its contents can be discarded after the task is switched. On the other hand, R2-R5 are saved needlessly, since they don't need to be preserved across Forth words. But including them simplifies the loop, and will come in handy later.

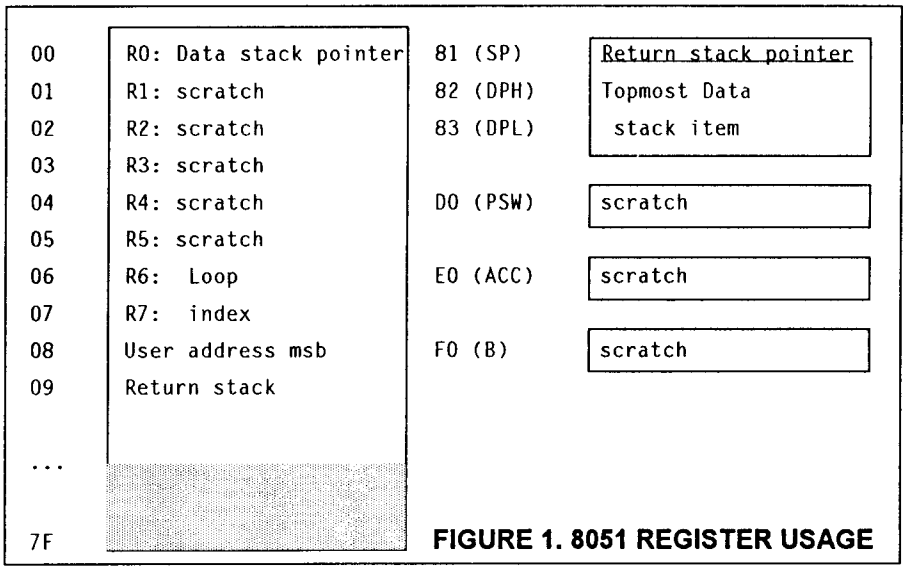


FIGURE 1. 8051 REGISTER USAGE

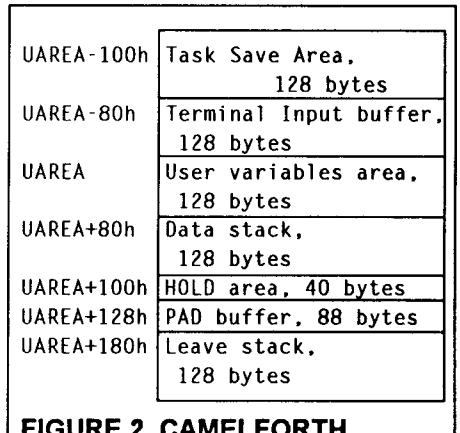


FIGURE 2. CAMELFORTH TASK AREA

After SWITCH copies the current task's context out to RAM, it fetches the new task's context (specified by the address parameter passed to SWITCH). The process is simply the reverse of what was done to save the context. This assumes that the new Task Save Area contains the data saved when *that* task did a SWITCH. What about a brand-new task?

The Forth word INITTASK sets up the Task Save Area for a new task. It saves 10 bytes in the Task Save Area: two bytes of return stack, and the eight registers 1 through 8. For register 8, the high byte of the User Variables area address is stored. For register 1, the low byte of the Parameter stack pointer, 0FD, is stored (this value will cause an empty stack once the task starts running). The "saved" return stack contains only a return address, which is where execution will begin when the task is started.

So, to create and launch a new task:

1. Reserve 768 bytes of storage for the Task Area. Call this area `taskname`. This name can be defined as a Forth CONSTANT, or by using CREATE if care is taken to keep the area page-aligned.
2. Initialize the task to perform a Forth word, with the command
   ```
   ' word taskname INITTASK
   ```
3. Launch the new task with the command
   ```
   taskname SWITCH
   ```

The "main" CamelForth task will be suspended, and the new task will run. Beware: if the new task never does a SWITCH back to the main task, it will retain control forever!

Listing 2 shows an example of how to define a task area in high RAM, by offseting it below the main task's user area (thus ensuring page alignment). It also shows how to write a task that will do something and then return to the main task: after loading this program, every time you type `TASK1 SWITCH`, the 8051 will emit a bell character.

ROUND-ROBIN TASK SWITCHING

For true multitasking, each task must run for a short time, and then hand off control to another task...making sure that all of the defined tasks get a turn. One way to do this is to have each task switch to the "next" task in the list. This can be done, clumsily, with SWITCH statements, as long as the list never changes (since task addresses are hard-coded in the program). A better way is to maintain a dynamic linked list of all tasks in RAM.

Listing 3 shows high-level words to manage a linked list of tasks. The first cell of the user variables (at offset 0 from UAREA) is reserved in CamelForth for a task link. MYTASK always returns the address of the running task area, by offseting -100h from UAREA.

DETACH empties the task list, by making the running task link back to itself. (This causes an inadvertent PAUSE to simply save and restore to the running task's context.) ATTACH adds a task to the list, inserting it immediately after the running task. Finally, PAUSE just causes a switch to the next task in the list. The secret to PAUSE is that it fetches a User variable, TASKLINK. User variables are always addressed relative to the current UAREA pointer, so whenever a task runs PAUSE, it fetches *its* link to the "next" task. In this manner, all of the tasks are linked together in a circular list. (See Figure 3.)

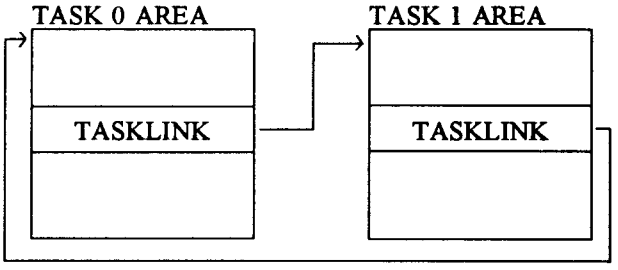


FIGURE 3. LINKED LIST OF TWO TASKS

Listing 3 also includes the "bell" demo, this time written with PAUSE. Every time PAUSE is typed at the keyboard, a bell character is output.

SWITCH can also be used to implement more complex task schedulers, by defining a "dispatcher" task (or a "dispatcher" Forth word). The dispatcher might, for example, select the task having highest-priority according to some rule. This is left as an exercise for the student.

PRE-EMPTIVE TASK SWITCHING

SWITCH and PAUSE require a task to voluntarily give up the CPU to other tasks. This "cooperative" multitasking is simple and efficient, and avoids many of the synchronization problems that can occur when many tasks share a resource. But poorly written tasks can "hog" the CPU, and with even the best-written tasks, it's difficult to parcel out CPU time evenly.

A pre-emptive multitasker uses some external event — typically a timer interrupt — to force a task switch. This ensures that tasks are switched regularly. But since a task switch can occur at any time — not just between Forth words — we must take pains to save *all* of the working registers that are used by CamelForth.

SWITCH saves all of the working registers except R1-R3, DPH, DPL, ACC, B, and PSW. The simplest solution is for the interrupt to push these onto the return stack, and then call SWITCH with the address of the next task. When the round-robin returns to this task, SWITCH will return to the interrupt service routine, which will then pop these eight registers and return to wherever this task was suspended. A subroutine PREEMPT to do this is given in Listing 4.

Note that PREEMPT fetches the same task link as does PAUSE. We can't use `ACALL PAUSE`, since the phrase `TASKLINK @` in PAUSE may destroy some registers that we haven't saved. (We could learn this by examining the kernel listing for USER and @, but it's bad form to build such hidden dependencies into kernel words.)

Also, PREEMPT saves all of the "extra" registers on the Return stack, except DPH and DPL. This is because SWITCH, before returning to PREEMPT, pops DPH and DPL from the *Data* stack.

Listing 4 also shows how to link PREEMPT to a timer interrupt, and initialize the timer. Listing 5 is the corresponding high-level test code. Once TASK1 is ATTACHed, the variable TICKS should increment every 65.536 msec (with a 12 MHz oscillator), and yet there should be no visible effect on normal Forth operation. Note the use of PAUSE to return control immediately to the "main" task, after the interrupt is processed.

A disadvantage of this approach is that the RETI instruction, required by the 8051's interrupt processing hardware, is not executed until the preempted task is resumed. For a simple example this is tolerable. To avoid this problem, PREEMPT should call a second copy of SWITCH, identical except that it ends with a RETI. Then PREEMPT should end with a RET.

Also, the preemptive multitasker could be made slightly more efficient if SWITCH saved all working registers (i.e., if PREEMPT were merged into the SWITCH routine). But this adds unnecessary overhead to the "cooperative" SWITCH. The code presented here clearly shows the extra context-switching overhead of a preemptive multitasker.

LISTING 1

```
; ================================================
; CamelForth Multitasker for the Intel 8051
; (c) 1996 Bradford J. Rodriguez
; Permission is granted to freely copy, modify,
; and distribute this program for personal or
; educational use.  Commercial inquiries should
; be directed to the author at 115 First St.,
; #105, Collingwood, Ontario L9Y 4W3 Canada
; ================================================

        .equ dr1,h'01    ; r1 as direct register

; The key word of the multitasker is SWITCH.
; It saves the working registers AND the return
; stack of the currently executing task to a
; storage area in external RAM.  Then it gets
; the saved registers and return stack of the
; new task, restores them, and continues
; execution wherever the new task left off.
;
; Registers as they are saved:
; 01 (R1): saved Parameter Stack pointer.
; 02 (R2): future use
; 03 (R3):     "
; 04 (R4):     "
; 05 (R5):     "
; 06 (R6): loop index
; 07 (R7):     "
; 08:      P2, User Area pointer high
; 09...N:  return stack (N is given by SP)
;
; DPTR is not saved, since it is consumed by
; SWITCH.  (It is the address of the new task's
; save area, UAREA-100h.)
;
; Note that these are stored backwards in
; external RAM, starting at address UAREA-100h.
; Thus the save area of a newly created task
; should look like:
; SP:     0Ah
; 0A,09:  init'l Program Counter, hi byte first
; 08:     task's User Pointer high (stack page)
; 07,06:  xxx
; 05,04:  xxx
; 03,02:  xxx
; 01:     0FDh, initial stack pointer
; The initial stack pointer must be FDh because
; of the poptos at the end of SWITCH.

; SWITCH     a -       switch to new task
        .drw link
        .set link,*+1
        .db  0,6,"SWITCH"
SWITCH:
      mov r2,dph       ; stash new task adrs
      mov r3,dpl
      mov dph,UP       ; save me at UAREA-100h
      dec dph
      mov dpl,#h'0
      mov dr1,r0       ; save my Pstack pointer
; This loop copies internal RAM, from location
; (SP) down to 01, to external RAM.  6+7n cycles.
; The length is saved as the first byte.
      mov a,sp         ; sp=high address,
      movx @dptr,a     ;   =length.
      inc dptr
      mov r0,a         ; 00 won't be moved
saveregs:
      mov a,@r0       ; 1 cycle
      movx @dptr,a     ; 2 cycles
      inc dptr         ; 2 cycles
      djnz r0,saveregs ; 2 cycles

      mov dph,r2       ; now get new task
      mov dpl,r3
; This loop copies external RAM to internal RAM,
; and restores SP accordingly.  6+7n cycles.
      movx a,@dptr     ; get high address
      inc dptr
      mov sp,a         ; restore Rstack pointer
      mov r0,a
getregs:
      movx a,@dptr
      inc dptr
      mov @r0,a
      djnz r0,getregs

; The top of this restored return stack contains
; a return address in the new task.  DPTR no
; longer contains its top-of-stack; so pop the
; new top of stack from RAM.
      mov r0,dr1       ; restore Pstack pointer
      mov p2,UP        ; set new stack page
      ljmp poptos      ; pop TOS and return

; ------------------------
; INITTASK   xt a -     initialize a task area
; Given the xt (code address) of a Forth word to
; execute, and the address of a task's save area,
; fill in that save area so the given word will
; execute when that task is started.
      .drw link
      .set link,*+1
      .db  0,8,"INITTASK"
INITTASK:
      mov a,#h'0a    ; length
      movx @dptr,a
      inc dptr
      movx a,@r0       ; low byte of xt
      inc r0
      mov r2,a
      movx a,@r0       ; high byte of xt
      inc r0
      movx @dptr,a     ; store high byte first
      inc dptr
      mov a,r2
      movx @dptr,a
      inc dptr
      mov a,dph        ; UAREA=SaveArea+100h, so
      inc a            ; DPH+1 = UAREA high byte
      movx @dptr,a
      inc dptr
      inc dptr         ; skip 6 don't-cares
      inc dptr
      inc dptr
      inc dptr
      inc dptr
      inc dptr
      mov a,#h'fd      ; initial Pstack pointer
      movx @dptr,a
      ljmp poptos
```

LISTING 2

```
( MULTITASKER TEST)

HEX U0 100 - CONSTANT TASK0  ( start of "main" Task
Area)
TASK0  300 - CONSTANT TASK1  ( new Task Area, 768
bytes )
                             (  lower)

: TEST1   BEGIN 7 EMIT  TASK0 SWITCH AGAIN ;
' TEST1 TASK1 INITTASK
```

LISTING 3

```
( ROUND-ROBIN TASK LIST)

HEX -100 USER MYTASK  ( start of current Task Area)
       0 USER TASKLINK ( link to next task in list)

( Initialize the task list to "empty".)
: DETACH
   MYTASK TASKLINK ! ;

( Insert a new task into the linked list,
immediately)
( after the current task.)
: ATTACH ( a - )
   TASKLINK @      ( my previous successor)
   OVER TASKLINK ! ( new task becomes my successor)
   SWAP 100 + ! ;  ( prev. successor becomes new)
                   (  task's successor)

( Switch to the next task in the list.)
: PAUSE   TASKLINK @ SWITCH ;

( EXAMPLE)
DETACH
MYTASK CONSTANT TASK0
TASK0 300 - CONSTANT TASK1
: TEST1   BEGIN 7 EMIT  PAUSE AGAIN ;
' TEST1 TASK1 INITTASK
TASK1 ATTACH
```

LISTING 4

```
; ---------------------
; PREEMPT                       force a task switch
; If entered from an interrupt, this will cause
; a switch to the next task in the round robin.
; The task link must be in the first cell of the
; user area (user variable U0).  Note that this
; is an assembler subroutine and must not be
; called as a Forth word.
PREEMPT:
      push psw      ; save regs used by SWITCH
      push acc
      push b
      push dr1
      push dr2
      push dr3
      lcall pushtos ; DPTR saved on Data stack!
      mov dph,UP    ; fetch task link...
      mov dpl,#0
      movx a,@dptr
      mov r2,a
      inc dptr
      movx a,@dptr
      mov dpl,r2
      mov dph,a     ; ...to DPTR
      acall SWITCH  ; switch to next task
; Execution will resume here when the round-robin
; comes back to this task.  Note that the last
; action of SWITCH is to restore DPH:DPL from the
; Data stack, with "poptos".
      pop dr3       ; restore regs
      pop dr2
      pop dr1
      pop b
      pop acc
      pop psw
      reti
; ---------------------
; Sample timer 0 interrupt, entered when timer 0
; rolls over from FFFF to 0000.  The interrupt
; flag is automatically cleared when the
; interrupt service routine is entered.
CLOCK:
      sjmp PREEMPT

; CLOCKON starts timer 0 & enables the interrupt
      .drw link
      .set link,*+1
      .db  0,7,"CLOCKON"
CLOCKON:
      mov tmod,#h'21  ; T1 mode 2, T0 mode 1
      mov th0,#h'0
      mov tl0,#h'0
      setb tcon.4      ; enable timer 0
      mov ie,#h'82     ; enable timer 0 irpt
      ret

; CLOCKOFF stops timer 0 & disables the interrupt
      .drw link
      .set link,*+1
      .db  0,8,"CLOCKOFF"
CLOCKOFF:
      clr tcon.4    ; disable timer 0
      clr ie.1       ; enable timer 0 irpt
      ret
```

LISTING 5

```
( PREEMPTIVE MULTITASKING TEST)
( requires round-robin support words)

DETACH                ( reset task list)
VARIABLE TICKS
: TEST2   BEGIN 1 TICKS +!  PAUSE AGAIN ;
' TEST2 TASK1 INITTASK
TASK1 ATTACH          ( add TASK1 to task list)
```

Small System Support

Regular Feature

68xx/68xxx Support

C & Assembly

by Ronald W. Anderson

Since we have worked our way through all of the C programming lessons that I had prepared, I thought this time we would talk more generally about writing a program to solve a problem. This column has come about because a reader has written me a couple of letters asking me to help him with two BASIC programs. I told him that his problem was not BASIC, but rather one of looking at a problem and figuring out a way to solve it. A procedure that solves a problem is called an algorithm. I thought I would present his request and my solution.

The writer is interested in analyzing the Lottery numbers for his state. The request was essentially:

Given a list of a number of winning lottery numbers (i.e. historical data), write a program that would report the number of times a given number (in the range of 1 to 39 for his state) had appeared. If a number appears one day (you might read "drawing" where I have used day here) and not the next, that fact is recorded as 1 hit. If a number appears 2 days consecutively but not on the third day, that is recorded as 2 hits (or a double hit), etc. The output is to be a table printed out on a printer. I organized the data per the writer's illustration, list the numbers 1 to 39 in the left hand column. Next column is to contain a count of the single hits in the data. Third column is to be a count of the double hits in the data,etc. with columns for up to 5 hits. The data file for a number of days is reproduced below.

```
2, 7, 12, 27, 29
1, 8, 21, 31, 38
2, 10, 36, 38, 39
2, 5, 19, 30, 32
8, 18, 25, 32, 33
1, 12, 24, 28, 30
1, 9, 10, 21, 37
3, 6, 15, 19, 38
4, 7, 27, 32, 38
1, 11, 29, 32, 37
1, 15, 21, 23, 26
1, 5, 26, 28, 33
6, 20, 25, 26, 35
1, 17, 18, 27, 32
9, 13, 21, 28, 39
1, 4, 23, 28, 30
1, 3, 6, 15, 20
5, 8, 19, 22, 34
4, 5, 26, 28, 38
```

The program below is in Quick BASIC. This BASIC has provisionfor integer variables, unlike some of the older BASICinterpreters. I've chosen not to use them in order to keep thisa little more like some of the older BASICs. First we'lldiminsion some arrays. Then we will get into the three phases ofthe program. I decided to read the data file into an array so Icould manipulate the values if required. I hadn't thought thewhole problem through at the time, but it turned out that aneasy approach was to read through the data 39 times, once foreach possible number. Reading through an array a number of timesis faster than closing and re-opening a data file a number of times.

```
10  DIM LD(5, 100): REM NUMBERS, DAY
20  DIM HT(39, 5):  REM NUMBER, HITS
30  REM WE WILL USE A VARIABLE HC = THE HIT COUNT
40  OPEN "LOTTO.DAT" FOR INPUT AS #1
50  D=1
60  INPUT #1, LD(1, D ), LD(2, D ),
        LD(3, D ), LD(4, D ), LD(5, D )
70  D = D + 1
80  IF EOF(1) THEN 100
90  GOTO 60
100 CLOSE #1
110 L = D : REM LAST DAY NUMBER +1
```

I didn't use a FOR D=1 to 100 loop because some BASIC interpreters won't guarantee the value of D if you escape the FOR-NEXT loop with a GOTO, i.e. get out of the loop early because of detecting End of File in the present case. This program as written will crash if the input file has more than 100 datalines. A test for D = 100 ought to be inserted for safety. Of course if you know you are going to have 150 data items you can simply increase the dimensions of the arrays in the program.

This next section is simply to see that we got the right numbers into our array. If you do something like this you can just as well delete it when you are done testing the first part of the program.

```
130 REM TEST INPUT DATA
140     FOR D = 1 TO 2
150         PRINT "DAY "; D ; " : ";
160         FOR N = 1 TO 5
170             PRINT LD(N , D ); " ";
180         NEXT N
190         PRINT
200     NEXT D
210     PRINT : PRINT : PRINT
```

Now we read the data file once for each possible number 1 to 39. In the code below the variable F is a "flag" that says a hit was found for the day. If it was found it was counted by incrementing HC. If there was no hit but we had accumulated a hit count from previous day's data, we increment the output array for the present number (K) and the number of hits that havebeen accumulated. Thus if we are looking at K=1 and have HC=2 F=0, we increment HT(1,2) and set HC to 0.

```
280 FOR K = 1 TO 39 : REM NUMBER
285  HC = 0
290  FOR D = 1 TO L : REM DAY OR DRAWING NUMBER
300    F = 0        : REM FLAG FOR A HIT
310    FOR N = 1 TO 5
320    IF LD(N , D) = K THEN HC = HC + 1: F = F  + 1
330    NEXT N
340    IF F = 0 THEN IF HC <> 0
        THEN HT(K,HC) = HT(K,HC) + 1
350    IF F = 0 THEN HC = 0 : REM SAVE COUNT
            AND RESET
360   NEXT D
370  NEXT K
380 REM NOW OUTPUT A TABLE
```

The remainder is simply to print the table. If we want to output to a printer in the present case of Qbasic we change PRINT to LPRINT for the table.

```
400 FOR K = 1 TO 39
410  PRINT K ; ":";
420  FOR N = 1 TO 5
430   PRINT HT(K , N ); " ";
440  NEXT N
450  PRINT
460  IF K = 20 THEN INPUT A$: REM PAUSE THE SCREEN
470 NEXT K
480 END
```

As I have been preparing this for this column, I found a simplification that indicates I hadn't thought it through very thoroughly the first time. I could therefore provide a good example of a poor program but I think I won't... Well, OK. I had made HC an array of dimension 39. It then came to me that I was handling one number at a time and only used HC(1) the first time through the loop, HC(2) the second time, etc. Therefore, HC didn't need to be an array. I can conceive of a more complex algorithm that could use HC as an array and only make one pass through the data, but that might be the subject for another time. By the way, I did something sneaky. The LD array was dimensioned 100... BASIC always initializes variables and arrays to all zeros. In the center working loop of the program I used the limit L which is one larger than the number of days of the data. All the numbers in LD(L) will be zero. The Flag F will be 0 and the remaining hit count if any will be written to the HT array. This is an easy way to insure that we don't leave any dangling hit counts at the end of the loop.

Let me warn you, however, that most other languages don't initialize variables and arrays unless you specifically do so in your code. If you were to write this program in C, you would have to do that, either clear the whole LD array initially in a double loop setting all the elements to zero, or clearing LD(L) after the limit has been established.

The Lottery

Having presented a Lottery data analysis progam I feel compelled to add my thoughts on the whole subject. You might be wondering why anyone would want such data.

One group would argue that the number 1 came up more than any other in the data, so the random number generator must be"biased" toward 1. Those folks would bet on a larger number of 1s and reduce the bets on numbers that didn't show up very often.

A second group would reason that the law of averages dictates that all the numbers should show up equally. There have been a large number of 1s, so in the future there have to be less occurences of 1 so the longterm average will balance out.

The truth of the matter is (disallowing that one pingpong ball could be biased to select itself more often) that pingpong balls have no memory of the past. If a number has occurred more often than expected after 20 tries, it is probable that after 40 tries it will have occurred more often than expected by about the number of excess in the first 20 tries. That is, no matter what the previous record, the best guess concerning the future is still entirely random.

The person that wrote for help has assured me that he is using the data and program(s) to follow the winners and see how well he could have done had he played. A friend of mine calls the lottery the stupid tax. He says it works very well, because the dumber you are the more you pay!

In the present case of picking 5 out of 39 numbers, the probability of winning is very small. You have 5 chances in 39 of your first pick being in the winning numbers. There are four winners left so you have 4 chances in 38 of picking the second winning number etc. The probability of winning is 5/39 * 4/38 *3/37 * 2/36 * 1/35 or 0.0000017368 by my calculator. That is one chance in 575,757. That is "on the average" there would be one winner for every 575,757 tickets purchased. This is not a guarantee, just an average. The only way to guarantee a win would be to buy 575,757 tickets and bet on all the possible combinations.

The deterrent to this sure win strategy is the very real possibility of someone else picking the winning combination too. People who have this much spare change to throw away are not generally of a mind to take such a chance. If they were, they probably wouldn't have a fortune.

I was listening to the radio the other day on the way to work. The MC was interviewing someone who wrote a book on playing the Lottery. He guaranteed that by using his "system" you could cut your average losses to only 40%. While I suppose cutting your losses such that on the average you only lose 40 cents on a dollar ticket might be a bit appealing, his system still won't make you a winner.

As I write this, the Michigan Lottery has a payoff of $43,000,000. Our Lottery was changed a while ago to make the winning probability a lot smaller in order to make the jackpots a lot larger. We have 49 numbers, and pick 6. The probability of a win is 0.00000007151124, or 1 in 13,983,820. The state quotes the chances as 1 in 14,000,000, so this looks like the correct calculation. I am assuming in both cases that the order of the numbers doesn't matter.

(Later) There were three winners of the $43,000,000 so it was split as I mentioned seems to happen when it gets so large that people flock to the sales places to buy tickets. The line was 8 or 10 deep at the convenience store this afternoon when I went to buy something other than a lottery ticket. By the way, I have heard that the Michigan State Lottery Commission uses several sets of numbered ping-pong balls, and on a given day chooses the set randomly. Thus if someone does detect a bias in the data over a period of time, it is no guarantee that the set chosen for a given day will have that same bias.

I just heard the oxymoron of the century, "The Michigan Lottery Smart Play". (An oxymoron is an incongruous pair or more of words, like "terribly good" or "Millitary Inteligence").

Miscellaneous

We have a friend who retired from teaching high school English about six years ago. At the time she was interested in trying her skills as an author so she asked me for help in getting acomputer that wouldn't cost an arm and a leg. We found one at the local Service Merchandise Store. It is an XT clone with a 30Meg RLL hard drive and 640K of memory. I found an older CGA cardwith composite video output and donated it and an old monochrome monitor that would work with it. We found a used daisy wheelprinter for $200 and she was in business.

A couple of months ago I received a phone call from her. It seems her son in San Francisco had suggested that she get onCompuServe so she could exchange Email messages with him. He sent a 2400 baud internal modem and a disk containingCompuServe's software package and offer of ten hours of free service. I agreed to install the modem when it arrived and I recruited my son-in-law to be, Joe, to help her get registeredand to set up the computer so it would be easy to get onCompuServe.

The first snag was that the computer has only a 360K drive, butthe software arrived on a 5" high density drive. Well, Ithought, I'll just go home and copy the software to a few 360Kdisks that I pre-formatted on the old XT. Second snag was that one of the files was longer than 360K!

I took the system home to work on getting the software transferred and found a nice shareware program that would split the long file so it would fit the smaller capacity disks. We installed the software and I added the modem, which we checked out by dialing a local bulletin board, and then I got a better idea. I had a 286 system in my collection that would run 7 times faster than the XT and perhaps make it less tedious to talk to CompuServe since the computer could respond faster. Problemwas that I didn't have a display for the 286.

I looked around the junk pile at work and found a monochrome VGA monitor that somebody had discarded because it supposedly had aproblem. I had been told to throw it away. I dusted it off andtried it out, and it seemed to work fine, so I threw it away in my car and connected it to the 286 system. I then transferred a number of directories full of files from the XT to this AT and ran it for a week, several times overnight to give it a little test. All worked fine so I brought it back to our friend and installed it announcing that I had substituted a faster computer and a much nicer monitor. Joe got her duely signed up with CompuServe and we were all happy.

A week went by and I received a call that the hard drive didn't always start. There would be a Hard Drive Failure message on thescreen. It had only happened once to me, the evening that I installed the computer. Meanwhile I had picked up another 40Mbyte hard drive (IDE) so I thought I would exchange drives.System came back to our house and I spent a couple more evenings changing drives and moving files.

The system got re-installed and all was well for a couple of weeks. Then I received another call. "The monitor went blank while I was editing". We decided to go back to the XT and CGA/mono monitor. I had by then had great success with INTERLINK, the utility in DOS 6.0 and beyond to connect two computers via serial null modem cable so I thought I would bring the XT back and simply copy files via serial link. I couldn't get the serial link working (repaired since) so I brought both computers home again (fortunately only 1/2 mile or so) and once more transferred files via 360K disks (about a dozen of them)... I've reinstalled the original system with modem and CompuServe software and so far no complaints.

I decided to try to repair the monitor. I rarely throw any non-working computer system component of any possible value away without trying to fix it so I removed the case and ran it for awhile. One semiconductor device got very hot and heated its heatsink to (estimated) about 140 degrees F. I suspected this device as the culprit, but it had no numbers on it that made any sense. I blew on it with a small fan and delayed the shutdown for a very long time, but eventually it would cut out. When it was working the picture was "perfect" and when it quit the screenwent blank.

Due to lack of a schematic or a reference for the semiconductor device (a three terminal TO-220 package) I finally gave up. A new monochrome monitor can be had for about $90 and I had spent a week of evenings and half of Memorial Day on the project. I didn't think it would be worth any more of my time. I assumedthe device to be a transistor emitter follower with a problem,and connected a transistor thus. The terminal didn't work very well, though the current was much less and the transistor didn't get very warm. I took the monitor back to work and did what I was told to do with it in the first place (throw it away). I now have two old 286 systems that work fine. I bought a VGA switchbox so I can run one of them on my main monitor by switching itfrom my "good" computer temporarily.

We win some and we lose some. Don't expect success every timeyou find a $10 computer at a garage sale. Persistence sometimes wins out, but not always. I still remain a willing recipient of old computer components working or otherwise. I feel it is always worth a little time to look at a non-working component because sometimes the problem is both obvious and simple to fix (like a broken wire in a video connector).

If you pay $10 for a computer and manage to blow it up while trying to get it running, or to find out that it has a problem that will prevent it from working without investing in, say a hard drive, you can chalk it up to education. Besides, after you have collected two or three ten dollar computers you will surely have enough working parts to be able to make one or two that run.

A technician that works with me has a friend who makes a "garbagerun" in Detroit on garbage days. He watches the curb for anything that looks like a computer or monitor. Sometimes the free goodies actually work and sometimes they are good for parts. Among the items has been a working 286

based PS-2 system. Nothing super, but a working computer with a hard drive. The only problem with PS-2 systems is that they are completely unexpandable with plain vanilla AT bus parts. They require special microchannel bus cards and disk drives made specificallyfor the PS-2. I suspect IBM wanted to come up with their own hardware so they could corner the market. I think the scheme backfired when people found out what they could do to upgrade their PS-2 hardware (throw it away and buy a new one), and realized that purchasers of AT clones had all kinds of ways to go to upgrade. At any rate, it is fun collecting old computers and trying to make usable systems out of them.

Since this is still a bit shorter than usual and since I have another tale for you, I'll go on a little. The other day, afriend of mine brought me a casualty 286 processor board. It came out of a computer he upgraded for a friend. The computer had quit and refused to boot. On taking it apart he discoveredthat the backup battery (Ni Cad) had leaked and the alkalineelectrolyte had begun to attack the copper conductors. The stuff actually crept along the conductors under the solder mask. Fortunately most of the conductors in the vicinity of the batterywere power busses, and so they were fairly large. I found the corrosion not to be soluble in water. I had to scrape it off after removing the battery. In the process, I scraped all the way through what remained of a couple of conductors, but a careful check with a multi-meter showed that all the feed-throughs were still connected and a couple of short wirejumpers soldered across the breaks in the conductors made them continuous again from the power connector through to the buss connectors.

I opened one of my old computers and attached the power supply, moving the VGA board to the repaired motherboard. On power up, the monitor came up and told me I didn't have a bootable drive, so next the combination floppy and IDE drive card was moved tothe repaired board. This time the system booted and I could run the software that was on the hard drive connected to the IDE controller. I've put that board aside for future reference. NowI have three 286 based 12 MHz AT systems. Two are in cases andcomplete except for a monitor. The third is just this extra board. I'll find a replacement Nicad battery for it and it willbe available to help someone out as soon as I can scrounge a few more hard drives.

Programming

I found an offer in a catalog from a company called SurplusSoftware Inc. 489 N. 8th Street, Hood River OR 97031. I include their address in case you would like to write for a catalog. Basically they buy leftover "old versions" and software inpromotional packaging and sell it at some very good prices if you can find something you want. In their last catalog was a CD-ROM called Language/OS published by Knowledge Media Inc. The ad promotes the disk as containing complete source code for 100 computer languages, operating systems and tools. At $13.50 I couldn't resist. I found a lot of stuff for UNIX, but there were some other goodies for DOS. Many of the things offered were written several years ago and are set up for 8086/88 systems.

Among the goodies I found were a complete Fig FORTH package, a Modula-2 package, and a complete BASIC interpreter with sourcecode in C. I found to my delight that the package compiled usingTurbo C version 3.0.

After my initial delight I started to use the package BywaterBASIC version 1.1. I found a few bugs, the largest being that any negative floating point number printed out as an integer. Print 1.234 got me 1.234000, but print -1.234 got me -1. Having thecomplete source code, I first sent a letter off to the author asking if there might be some updates available. V 1.1 was dated1992, so it is fairly likely. Meanwhile, I began to look through it and insert print statements here and there to track theprogress of the variable through the print routine module. I discovered that it was getting fouled up by the routine that decided how many digits to print. It did an fmod(dval ,d) and compared the remainder with a constant, dropping out of a loop that incremented the number of digits after the decimal point starting at a value of zero. It occurred to me that perhaps fmod() would return a negative number which would be less than the constant in any case, but would work properly with positive arguments. I added an fabs() function to take the absolute valueof the value before the mod calculation, and suddenly I had negative numbers printing just fine.

Next problem was that the interpreter doesn't properly handle unary negation, i.e. I set a variable pi = 3.14159. print PI now works fine but another asignment statement a=-pi chokes on asyntax error. Obviously the expression handler doesn't like two operators in a row (= -). It bought a = 0 - pi, so obviously theauthor simply overlooked unary negative. (A unary operator operates on one argument. Operators that need two arguments suchas * or / are called binary operators (no relation to binary numbers other than the fact that the number 2 is involved inboth).

I have not found the problem there. The expression evaluator is fairly complex (as are most) and there seems to be no provision for the unary minus operation. I tried a few things to attempt to fix the problem but I will have to spend a lot more time learning how it works before I will be able to fix it properly. I think I'll wait for news on newer versions, at least for awhile before I spend a lot more time on it.

It occurred to me that maybe these problems were because of a difference between Turbo C and the compiler used on the original code, so I went back and found the original .exe file on the CD-ROM. It worked identically. I think you would find the story in the documentation interesting.

I found I could do a few nasty things in the way of errors thatwould make the interpreter go West, though it has extensive error trapping and reporting features. I'll have to spend some time tracking down how to trap a few more syntax errors. Loading a program that didn't have line numbers (a QBASIC program) went OK but an attempt to list it caused it to go into an infinite loop listing blank lines and it required a reset to get out.

Last Word on Bywater Basic — My letter to the author was returned as was my Email attempt. As nice as it might be to havea BASIC interpreter with complete source code, I decided that since I already have QBASIC, which works fine, I would find somemore interesting projects to do, so I abandoned further efforts on the project.

I/O Software and Operating Systems

by Dave Baldwin

Special Feature

I/O Software and

Operating Systems

The primary performance limit on I/O software, such as serial communication programs, is the amount of time available to perform the task. This is determined by the speed of the system and by the amount of time that is used by the system which is not available to your program. Interrupts, timer overhead, and background processing limit the amount of processing time available for your program.

Operating systems can have a serious impact on the performance of your programs. Some have a large amount of background processing activity that eat up system resources. Some use memory and I/O management hardware to prevent direct access by 'unauthorized' software. To determine whether your program can actually perform it's job, you need to look at the amount of time available and the kind of access allowed under a particular system.

There are five different levels of Operating Systems in my scheme, levels 0 thru 4, with each level having different programming requirements. Level 0 is the minimal single board computer with a ROM monitor or you might have to write the monitor yourself. Level 4 is Unix, Linux, and the other multi-user, multitasking operating systems where you are not allowed to write programs that do simple I/O.

Level 4 problems can be added to the lower levels when they include a network that allows others to access resources on your machine. Sharing files, printers, and modems with other machines turns a single user system into a kind of multi-user system. File tranfers to and from your disks and remote printing to your local printer can make it very difficult to predict the response time of your program. The interrupts and background processing required to maintain the network connection take a unpredictable amount of time and resources.

Level 0

Your basic single board computer usually comes without an 'operating system' and provides very little built-in I/O. It's expected that you bought it (or designed it) and plan to write your programs to control the hardware - so it's all going to be DIY. Yes, I know that you can buy SBC's that include DOS, Windows, Unix/Linux, etc., but I don't call such systems a 'basic single board computer' (they're really levels 2 to 4).

The advantage of the basic SBC is that you can do it all yourself. Timer interrupts don't have to do anything extra to keep track of 'system events' or related background processing. All other interrupts will be directly related to your task and, since you wrote them, you will be able to account for the time they take.

The disadvantage of such a system is that you HAVE to do it all yourself. You can buy Real Time Operating systems to help you manage your system. You have to take into account the overhead required by them, the learning curve, the lack of source code, and waiting for someone else to fix system bugs.

Level 1

CP/M and similar OS's like early versions of PC/MS-DOS are at Level 1. They provide a file system, console I/O, and printer output. Decent serial I/O is up to you. What they don't have is a lot of overhead. Maybe a simple periodic timer interrupt, though even that, on a slow system, can limit performance. Timer interrupts are used for disk time-out functions and some have video interrupts for screen control functions (like the Kaypro screen interrupt which can limit serial performance to 2400 baud if not dealt with). Everything else is DIY with very little interference (or help) from the OS. Recent release of source code for these OS's means some 'tweaking' of the code is now possible. You could build your own real-time system based on one of these 'free' source code releases, and thus be able to shift it to handle your needs better.

Level 2

Recent versions of PC/MS-DOS are at Level 2. Although they do provide many more drivers for different devices, you can still write and run your own I/O programs. In fact, some of the device drivers (like the COM driver) are so poor that you have to write your own if you want decent performance.

The biggest difference between Level 1 and Level 2 operating systems is the amount of background overhead. On 386+ PC systems, the SMARTDRV disk caching program is normally installed along with HIMEM.SYS, the memory management program. These two programs 'hook' most of the system interrupts and do their work in the background. The timer interrupt in particular is used by SMARTDRV to tell whether it is time to write data to disk.

This 'background' work can take a fair amount of CPU time and limit I/O performance. A simple example of these effects is serial transfers. My 386/33 with SMARTDRV and HIMEM.SYS loaded has problems above 19.2Kb. If I boot from a floppy without them, the same serial program will run at 57.6Kb without any errors or problems.

Level 3

Levels 0 thru 2 are fairly general purpose systems without a lot of limits. Levels 3 and 4 are designed to be 'desktop' or

'workstation' systems almost exclusively and they are intended to limit I/O access to 'approved' drivers to preserve the 'integrity' of the system. If you really have to do I/O under levels 3 or 4, you may have to buy the appropriate SDK's and development software. The learning curve can be fairly steep if all you've done is simple programming under the lower levels.

The Windows OS's are at Level 3. They are 'protected mode', single user, task-switching operating systems. In 'protected mode', the memory management hardware of the 386+ CPU's is used to limit access to memory and I/O addresses. This is what initiates the 'General Protection Fault' errors you see at times in Windows.

The Windows OS's also have a lot of overhead. In a way, Windows is the main program and all you do is write subroutines that use the Windows facilities (API's). So most of the time, it's Windows that is actually running, not your program. Windows is in control, not you.

Want to display something on the screen? You make up a list of what you want and where you want it and you give it to Windows to actually put it on the screen. If you try to access video memory directly, you'll get a GPF because video memory 'belongs' to Windows, not to your program.

Each program is 'allocated' memory to use when it runs. If it tries to access memory that is not allocated to it, you get a GPF. If a program needs more memory, it has to ask Windows for more.

Windows reserves all I/O access to itself. You have to 'install' a driver that requests and gets I/O access privileges to do I/O without problems. If you try to access I/O addresses directly, you will get a GPF. Yes, I know that under Win3.1, you can run any old DOS program you want to do anything you want. You can also cause yourself problems if your DOS program uses I/O that your Windows program also uses.

Win95 and Win NT get progressively more picky about DOS I/O. Apparently, it's almost useless to try to do DOS level I/O in an NT DOS box. I'm told that every time you try to access a different I/O device, your program stops and a 'box' pops up and asks you whether you want to allow your program to access this I/O device or address. Win95 is a mix of old DOS and NT concepts. Accessing I/O however has the same problems - it's best to use VB (Visual Basic) or such if you can live within their limits (usually slower 19.2 MAX on a 486/100 or above) and some restraints on data formats.

Level 4

This is the level for Multi-user, preemptive Multitasking operating systems like Unix and Linux. Like Windows they are 'protected mode' operating systems. Unlike Windows, they are designed to keep on running if a single program causes a 'protection fault'. They just cancel the offending program and keep everything else running.

Because they are designed to keep on running in spite of errors, simple I/O is not allowed at all. All I/O is done through OS driver extensions that have to be installed correctly and request the appropriate privileges. If it can't be done through the 'device drivers', it generally can't be done at all. Keep in mind you can write new drivers - if you have the skill and knowledge - or stay within the limits of the system drivers.

Another problem with Level 4 systems is that you are sharing the machine with other users and programs. The amount of time available to your program can vary a lot. With the Windows programs in Level 3, you can still be sure that your program is the 'main' program even if you do have to account for the overhead. On a Level 4 multi-user system that really does have multiple users, you can't be sure of that. It can get very difficult (almost impossible) to predict your response time.

Networks

Networks are almost always part of Level 4 systems. They can be part of the lower level systems as well. When they are, the character of the lower level systems can change dramatically.

When networks are used to share resources between single-user machines, the background processing time becomes dependent on the overhead required to maintain the network connection and the resource requests from the other machines. High speed networks using Ethernet cards, can have the card doing a lot of the processing. Simple serial networks use the CPU's time to do everything. If you're setup to share files with other machines on the network, your ultra-fast machine can suddenly (and unpredictably) become the slowest dog in town if everybody wants files from your machine at the same time.

Conclusion

There are development kits available that claim to be able to do just about anything you want. The adds seem to proclaim that they can overcome any restrictions. I don't believe it. You can never really overcome the limits of your operating system.

Without an operating system, you're limited only by your hardware and your own software. As you go up the levels of operating systems, the OS's take progressively more time away from your program. They also add the convenience of the additional built-in functions.

When you start a project, you need to figure how much capability you need. Sometimes simple hardware can out perform faster complex systems simply because of the lack of overhead. An 8085 or a Z80 CPU and a pair of 8250 UARTs is fast enough to run continuous, nonstop, full duplex serial I/O at 38.4Kb on two ports without an operating system. A standard 386/33 PC with DOS6 and a normal installation with SMARTDRV would be hard pressed to do that because of the background overhead. You would need to upgrade the UARTs to 16550's with FIFO's so that the serial ports could keep on running when the system is doing background processing.

It's not unusual to make systems that combine the different levels. Your 'standard' PC with a hard disk and a modem is the mosty common example. While the main 80x86 CPU runs DOS or another disk-based OS, there's a CPU in the hard disk that just takes care it, another in the keyboard, and another in the modem. These other CPU's are at level 0 and don't know anything about the higher level operations of the main CPU.

The Computer Corner

Regular Feature

Editorial Comment

Linux or DOS?

By Bill Kibler

Since I no longer manage TCJ, my computer systems usage has been changing. When I was managing everything, I had started setting up a full networked system, with the idea in mind that at some point I would need several workstations to handle TCJ's needs. I have completed the network and am reconsidering my needs and usage.

In the quest for more computing power, I started looking at Linux as an operating system. I have OpenLinux from Caldera installed now, and have tried Slackware and Yaggdrasil as well. At work I now use Unix almost exclusively, although I still have an NT workstation for e-mail and text formatting needs.

While using Caldera for Linux, I have loaded Caldera's OpenDos and looked at their CP/M source code. I have been playing with NT, OpenDos, and Linux in order to settle on an operating system. Since I no longer need all this horsepower, it is causing me to reconsider my computing needs.

The Choices

In reconsidering my needs, I have been pondering the concept of three system types in which way to go. The types are "stick it out", "go for broke", "use them all." Unlike most choices where it is some sort of hardware or operating system that controls the decision, this is based on usage and needs.

To explain needs usage, let's talk about the "stick it out" concept. Here the user or person is still using their first computer. By this I mean, you buy one computer complete with operating system, word processor, and most of the tools you need and never buy anything else. This could be a Kaypro, an early XT or AT, or maybe your a late bloomer and are using the latest full powered system. The main line here is not staying up with technology, but instead using the system for it's set of utilities and needing nothing else. You're happy, it works, and you have learned it all. Why start over again learning things that might not give you anything more than you already have?

I contrast the "stick it out" concept with the "go for broke" philosophy. Here the idea is to stay on top with the latest system and software. You are always upgrading not only hardware but software and utilities. Since you most likely are buying and selling, maybe even building your own systems, the cost is not really a factor. Most likely you end up testing, installing, and trying things constantly as you try to learn and use the latest hot item. It never ends and often it seems I spent more time installing than using when I have used this approach.

The last approach is "using them all", where you have a little of both going on. I have been moving in this direction, although lately starting to reconsider the merits of such a move. To use them all you must understand the old and the new. There is constant learning and adaptation and tons of problems to overcome. The advantages come from using some of your old tools and platforms since you are likely to be very good at using those tools. Disadvantages come from some platforms that are very poor when used with your favorite tools. A considerable amount of time are also spent on trying to make things work.

NT to Linux

Let me review my current installation at home. I have four systems, two in the office (the old TCJ workhorses), one in the house (the wife's and son's Windows 95 system), and one in the workshop for downloading files from the S-100 CP/M systems. I run an Ethernet backbone that ties all the units together for backing up and transferring data between the boxes. The office units are the ones I am playing with at present, one an NT box and the other an Caldera Standard OpenLinux Server. The server has a 4.3GB hard drive and was setup with Samba to allow backup from any of the other platforms.

I had some setup problems with the Server, but once I read the books and decided it was actually the way they said, all has worked very well. You can mount the drive and move files very easily from any of my systems and I have not had any failures. Since I work on Unix systems for a living, the Linux/Unix operating procedures are not a big issue. I know enough to start and stop the system and use it as a backup platform. All the tools that Linux provides are still somewhat foreign to me, since there are just too many of them to learn.

The NT workstation has RelectionX, an X server that allows the NT workstation to act as if it was a Unix Workstation. This allows me to run the graphic interfaces to linux on my NT station. Well, that is at least the idea. I have had considerable trouble setting it up and have yet to get it to work correctly. At my last job, I used this same package every day without the same problems I am facing. I have yet to determine the problem, but it is one of the reason I have been reconsidering my options.

What Changes

I have started re-thinking my computer usage for many reasons. One has been the amount of time spent trying to resolve the X problem. Another has been my reduced demand for computing and thus needing a big system. I am also starting to tire of being in a setup and learning mode all the time. So how do I choose one of the options I outlined above. Let's start by looking at some of the pros and cons.

One of the nice things I like about CP/M is it's small size. The other day I fired up an CP/M system to take to a show and got a big smile on my face, as I inserted the 360K floppy, hit reset, heard it load one then two tracks of data and saw the prompt. It is almost a tie as to which system takes longer to boot up, NT or Linux, but I think linux is faster. Both systems are intended to run non-stop until they crash that is, and then hopefully they will re-boot after the crash. Do not believe the media about how solid NT is, it crashes often and you will be amazed at how often it is necessary to reboot due to adding this or that package. Linux does crash, not often, but it can. Linux does not have the "must reboot" NT syndrome when loading programs, but some changes to the kernel, will require a reboot (almost never...).

For both PCDOS and CP/M, the commands and tools are for the most part very simple and easy to learn. The set of tools is limited, but enough to do what needs to be done. Linux is the extreme in the other direction, there are hundreds of tools, most with very cryptic command lines to master. At work I still must use the "man pages" (on line documentation) often to get the correct syntax as do most Unix users. NT and WIN95 both have so many utilities that often don't follow the help descriptions of how they work, that they seems more of a problem that they are worth. NT and WIN95 both rely on "Wizards" to do the work as no person could possibly figure them out.

As to installation on a new system, PCDOS is probably the most straight forward to install. CP/M on a new platform requires many programming skills, a lot of teeth mashing, and plenty of pure luck. I did it many times years ago and do not recommend it for the faint at heart. NT and WIN95 both are simply follow the prompts and hope for the best. You stand about an equal chance of having the installation work or be screwed up. I have found their configuration problems to be from loading multiple copies of drivers that later require hand removal to get things like modems to work. The Linux community is ever striving to improve their installation procedures. I have installed all the major versions of linux and have video tapes to help explain it as well. I still make mistakes doing it and often will need two or more shots to get it working correctly. I find the lack of exit and re-entry options in their setup scripts a major problem, make a mistake and start over from the beginning (I mean hit reset and reboot).

In reconsidering my option, I have looked at word processing and text editing. I have used WordStar since version one and two, so I feel very comfortable with it. I have tried all the others and find the current crop of windows based word processors to be horrible. Word is the worst editor of them all and I find them all difficult to use despite the windows interface. For text or program editing I like my SPE (Sage Professional Editor - now called Preditor) the best because of it's multiple window cut and paste options, and especially it's column options, missing on most Windows based editors. The fact that these programs are both PCDOS based and not windows based I think is a plus, but shows up as a problems when using them in DOS Boxes on NT or 95 (run like #$%^ in DOS Boxes).

Networking is a dream on Linux, since all Unix like systems are made to work remotely. NT and W95 have built in networking but configuration will be a trail and error adventure with uncertain results. Once installed, my Linux Server works great with the other units. I can do a DOS connection, but have not yet done it, although I know it is possible as I have set up systems at work. I remember the DOS setup as straight forward and somewhat simpler than NT/W95. Networking machines does not have to be with ethernet cards and complex installations. The Little Big Lan works fine using serial ports between PC's and there is work on a Linux port and it supports a few ethernet cards as well. There is also several public domain programs for tying DOS type machines together, and some of us are even trying to figure out how to make CP/M play on the network.

The internet has become a main driving force in computing and certainly influences my choices. Until recently you were limited to a Windows or Xwindows based web browser for surfing the net. Caldera now has WebSpyder, a DOS based graphics browser, and of course there are several versions of Lynx for DOS browsing without the graphics. I like Netscape over IE for many reasons, although up until lately Netscape was by far the better when printing was done. Netscape is available on both my NT/W95 platforms and on my Linux. Another option is using programs like Plume which is a Tcl/Tk web browser. I find the options becoming more and more platform independent and thus becoming a non-issue for choice of system.

What choice next

Well I still am researching my options and would like feedback from readers on their choice. I am testing using Linux as the do all platform since it can run DOS (and some windows) programs and Unix programs. I will let you know in an article about using Linux as the do all platform of choice. Some of the problems setting up this linux test has made me consider using DOS almost entirely, only using linux as the backup server since once setup it seems to work with little intervention on my part.

I did a search the other day on the internet and was impressed with the number of DOS programs out there. I really feel it is possible to do it all on an old DOS machine. Drop me a note if you agree. Till later keep hacking.

Bill Kibler

The TCJ Store

Regular Items

Back Issues ..... **See page 44**
All Back Issues of TCJ are available.

TCJ Reference Cards ..... **$3.00 + $1 S+H**
So far, all we have is the Z80 Instruction Set card from Issue #77. These are on heavier stock than the one sent with the issue.

The next two items are Group Purchase Items. TCJ doesn't have the resources to stock these for you, so we have to collect a minimum number of orders before we can provide these.

***GIDE kits** **$73**
Tilmann Reh's GIDE board was featured in several issues of TCJ. It is a 'Generic' IDE board for the Z80 that plugs into the Z80 socket (you plug the Z80 back into the GIDE board). This is still an experimenters kit. Sample code and docs including the articles from TCJ are provided, but you have to write your own BIOS routines.

SOLD OUT

CP/M CD-ROM ..... **$25 + $4 S+H**
This is the Walnut Creek CP/M CD-ROM (normally $39.95+S&H), with 19,000 files from Jay Sage, David McGlone, FOG (First Osborne Group), the Beehive BBS, the Enterprise BBS, ftp.demon.uk, and the SimTel20 CP/M collection from the Internet.

Special Items

We currently have two each of Tilmann Reh's **CPU280** boards and the **IDE** boards that go with them. The CPU280 was featured as the Centerfold in Issue #77 and the IDE interface was in Issue #56. These are bare boards and are not for the faint of heart. They are expensive and the parts are hard to get. But they're fast.

***CPU280 bare board** **$150**
Comes with docs and utility disk.
***IDE bare board** ..... **$ 65**
Comes with docs.
***CPU280 & IDE together** **$200**

CP/M Kaypro Catalog

TCJ has taken over Chuck Stafford's Kaypro business. Here's our current catalog.

Upgrades	
Advent TurboRom	
K4-83	$35.00
K10-83	$35.00
Kx-84	$35.00
MicroCornucopia Roms	
Pro 8	$35.00
884 Max	$35.00
884 Max (Lo)	$35.00
Character ROM	$35.00
Add-ons	
HandyMan	$75.00
Disk Drives	
Dual Density TEAC FD-55BV	$15.00
Quad Density TEAC FD-55FR	$15.00
Pair	$25.00
ST-225 20 MByte MFM HD	??
Disk Controllers	
WD-1002-05 HDO	$75.00
Tech Data	
Kaypro Technical Manual	$25.00
Microcornucopia Schematics with Theory of Operation	
K-II/4 83	$15.00
K-10/83	$15.00
All-84	$15.00
Any two	$25.00
All three	$30.00
Software	
Advent Harddisk Formatter	$25.00
TurboRom Applications Patches	$10.00
TurboRom Developers Diskette	$10.00
Kaypro 10/83 Tinker Kit	$10.00
Kaypro 2,4/84 Tinker Kit	$10.00
Kaypro CP/M 2.2H Autoload set 8 diskettes for K-10/84	$40.00
Other Stuff	
Keyboards	$30.00
Video - CRT and board	$40.00
Kaypro Carrying bags	$75.00
Kaypro machines	
K-II, K-2, K-4, K-10 available in various condition.	

TCJ can accept credit card orders by phone, fax, or mail or you can place an order by sending a check to:

The Computer Journal
PO Box 3900, Citrus Heights
CA 95611-3900
Phone: 800-424-8825 or 916-722-4970
Fax 916-722-7480 / BBS 916-722-5799

Include your shipping address with your check, and your Internet address if you have one. For more info, contact TCJ via E-mail at tcj@psyber.com

* In Europe and particularly Germany, contact Tilmann Reh for a current price and shipping. His email address is: "tilmann.reh@bigfoot.com".

His postal address is:
Tilmann Reh
Autometer GmbH
Kaenerbergstrasse 4
57076 Siegen (optional "-Weidenau")
GERMANY

Regular Feature
Contact Listing

SUPPORT GROUPS FOR THE CLASSICS

TCJ STAFF CONTACTS

TCJ Editor: Dave Baldwin, (916)722-4970, FAX (916)722-7480 or TCJ BBS (916) 722-5799 (use "computer", "journal", pswd "subscriber" as log on), Email: tcj@psyber.com.

TCJ Adviser: Bill D. Kibler, PO Box 535, Lincoln, CA 95648, (916)645-1670, E-mail: kibler@psyber.com.

32Bit Support: Rick Rodman, 1150 Kettle Pond Lane, Great Falls, VA 22066-1614. Real Computing BBS or Fax: +1-703-759-1169. E-mail: ricker@erols.com

Kaypro Support: Charles Stafford, on the road somewhere. Email: CIS 73664,2470 (73664.2470@compuserve.com). TCJ has taken over Chuck's Kaypro parts and upgrade business.

S-100 Support: Herb Johnson, 59 Main Blvd. Ewing, NJ 08618 (609)771-1503. Also sells used S-100 boards and systems. E-mail: hjohnson@pluto.njcc.com. Web page: http://pluto.njcc.com/~hjohnson/

6800/6809 Support: Ronald Anderson, 3540 Sturbridge Ct., Ann Arbor, MI 48105. Email: rwilande@concentric.com

Z-System Support: Jay Sage,1435 Centre St. Newton Centre, MA 02159-2469, (617)965-3552, BBS: (617)965-7046; E-mail: Sage@ll.mit.edu.

REGULAR CONTRIBUTORS

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

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

Tilmann Reh designed the GIDE board and has many programs for CP/M+ and is active with Z180/280 ECB bus/Modular/Embedded computers. Microcontrollers (8051). E-mail: tilmann.reh@bigfoot.com

Helmut Jungkunz, Munich, Germany, ZNODE #51, 8N1, 300-14.4, +49.89.961 45 75. Email: helmut.jungkunz@metronet.de.

USER GROUPS

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

SMUG, Sacramento Microcomputer Users Group, has disbanded after all these years.

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

NOVAOUG: The Northern Virginia Osborne Users Group, Newsletter $12, William E. Kost, 7007 Brocton Ct., Springfield, VA 22150. Info: 703-569-2213, BBS use CAPDUG's.

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

NATGUG, the National TRS-80 Users Group, Roger Storrs, Oakfield Lodge, Ram Hill, Coalpit Heath, Bristol, BS17 2TY, UK. Tel: +44 (0)1454 772920.

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

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

Adam International Media, Adam's House, 1829-1 County Road 130, Pearland, TX 77581-5040. Contact Terry Fowler for information. Web: "http:// WWW.Flash.Net/~coleco"

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

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

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

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

Vancouver Island Senior ADAMphiles, ADVISA newsletter by David Cobley, 17-885 Berwick Rd. Qualicum Beach, B.C., Canada V9K 1N7, (604)752-1984. Email: dcobley@qb.island.net

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

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

The San Diego Computer Society (SDCS) is a broad spectrum organization that covers interests in diverse areas of software and hardware. It is an umbrella organization to various Special Interest Groups (SIGs). Voice information recordings are available at 619-549-3787.

The **Dina-SIG** part of SDCS is primarily for Z-80 based computers from Altair to Zorba. The SIG sponsored BBS - the Elephant's Graveyard (619-454-8412) - is open to all callers who are interested in Z-80 and CP/M related machines and software. Contact Don Maslin, head of the Dina-SIG and the sysop of the BBS at 619-454-7392. Email: donm@cts.com.

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

chapters.

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

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

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

OTHER PUBLICATIONS

The Z-Letter - has ceased publication.

The Analytical Engine, by the Computer History Association of California, 4159-C El Camino Way, Palo Alto, CA 94306-4010. Home page: http://www.chac.org/chac/ E-mail: engine@chac.org

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

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

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

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

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

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

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

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

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

Update Magazine, PO Box 1095, Peru, IN 46970, Subs $18 per year, supports Sinclair, Timex, and Cambridge computers. Emil: fdavis@holli.com.

SUPPORT BUSINESSES

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

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

Elliam Associates, PO Box 2664, Atascadero CA 93423, (805)466-8440. Sells CP/M user group disks and Amstrad PCW products. Email:??

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

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

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

Peripheral Technology, 1250 E. Piedmont Rd., Marietta, GA 30062-2821, (404)973-2156. 6809/68000 single board system. 68K ISA bus compatible system.

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

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

MicroSolutions Computer Products, 132 W. Lincoln Hwy, DeKalb, IL 60115, (815)756-3411. UNIFORM Format-translation, CompatiCard and UniDos products have been discontinued. Web page: http://www.micro-solutions.com.

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

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

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

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

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

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

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

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

The Computer Journal Back Issues

Sales limited to supplies in stock.

Volume Number 1: Issues 1 to 9

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

Volume Number 2: Issues 10 to 19

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

Volume Number 3: Issues 20 to 25

- Designing an 8035 SBC
- Using Apple Graphics from CP/M
- Build an S-100 Floppy Disk Controller
- Extending Turbo Pascal: series
- Analog Data Acquisition & Control
- Programming the 8035 SBC
- Variability in the BDS C Standard Library
- The SCSI Interface: series
- Using Turbo Pascal ISAM Files
- The Ampro Little Board Column: series
- C Column: series
- The Z Column: series
- The SCSI Interface: Introduction to SCSI
- Editing the CP/M Operating System
- INDEXER: Turbo Pascal Program
- Introduction to Assembly Code for CP/M
- Ampro 186 Column
- ZTime-1: A Real Time Clock for the Ampro Z-80 Little Board

Volume Number 4: Issues 26 to 31

- Bus Systems: Selecting a System Bus
- Using the SB180 Real Time Clock
- The SCSI Interface: SCSI Adapter Software
- Inside Ampro Computers
- NEW-DOS: The CCP Commands (continued)
- ZSIG Corner
- Affordable C Compilers
- Concurrent Multitasking: DoubleDOS
- 68000 TinyGiant: Hawthorne's Low Cost 16-bit SBC and Operating System
- The Art of Source Code Generation
- Feedback Control System Analysis
- The C Column: Graphics Primitive Package
- Hitachi HD64180: New Life for 8-bit Systems
- ZSIG Corner: Command Line Generators and Aliases
- Writing a Tutor Program in Forth
- Disk Parameters: Modifying the CP/M Disk Parameter Block for Foreign Disk Formats
- Build an A/D Converter for the Ampro Little Board
- HD64180: Setting the Wait States & RAM Refresh using PRT & DMA
- Using SCSI for Real Time Control
- Patching Turbo Pascal
- Choosing a Language for Machine Control
- Better Software Filter Design
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 1
- Using the Hitachi hd64180
- 68000: Why use a new OS and the 68000?
- Detecting the 8087 Math Chip
- Floppy Disk Track Structure
- Double Density Floppy Controller
- ZCPR3 IOP for the Ampro Little Board
- 32000 Hackers' Language
- MDISK: Adding a 1 Meg RAM Disk to Ampro Little Board, Part 2
- Non-Preemptive Multitasking
- Software Timers for the 68000
- Lilliput Z-Node
- Using SCSI for Generalized I/O
- Communicating with Floppy Disks: Disk Parameters & their variations
- XBIOS: A Replacement BIOS for the SB180
- K-OS ONE and the SAGE
- Remote: Designing a Remote System Program
- The ZCPR3 Corner: ARUNZ Documentation

Issue Number 32:

- copies still available -

Issue Number 33:

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

Issue Number 34:

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

Issue Number 35:

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

Issue Number 36:

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

Issue Number 37:

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

Issue Number 38:

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

Issue Number 39:

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

Issue Number 40:

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

Issue Number 41:

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

Issue Number 42:

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

Issue Number 43:

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

Issue Number 44:

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

Issue Number 45:

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

Issue Number 46:

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

Issue Number 47:

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

Issue Number 48:

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

Issue Number 49:

- Computer Network Power Protection
- Floppy Disk Alignment w/RTXEB, Pt. 1
- Motor Control with the F68HC11
- Home Heating & Lighting, Pt. 1
- Getting Started in Assembly Language
- PMATE/ZMATE Macros, Pt. 2
- Z-System Corner/ Z-Best Software

Issue Number 50:

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

Issue Number 51:

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

Issue Number 52:

- YASBEC, The Hardware
- An Arbitrary Waveform Generator, Pt. 1
- B.Y.O. Assembler...in Forth
- Getting Started in Assembly Language, Pt. 3
- The NZCOM IOP
- Servos and the F68HC11
- Z-System Corner, Programming for Compatibility
- Z-Best Software
- Real Computing, X10 Revisited
- PMATE/ZMATE Macros

- Home Heating & Lighting, Pt. 3
- The CPU280, A High Performance SBC

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

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

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

Issue Number 56:
- TCJ - The Next Ten Years
- Input Expansion for 8031
- Z-Sys Corner - Zed-Fest
- Connecting IDE Drives to 8-Bit Systems
- 8 Queens in Forth
- Real Computing - Linux, BSD 386, Minix
- Kaypro-84 Direct File Transfers
- Analog Signal Generation

Issue Number 57:
- Z-Sys Corner - Language Independence
- DR. S-100 - the start
- Home Automation with X10
- File Transfer Protocols - Info
- MDISK at 8 MHZ. - Ampro Update
- Shell Sort in Forth
- Introduction to Forth
- Z AT Last! - ZCPR on a PC? MYZ80!

Issue Number 58:
- Z-Sys Corner - Language Independence II
- Real Computing - Minix, UZI, and GNU
- Affordable Development Tools
- DR. S-100 - Tips and info
- Mr. Kaypro - Move the Reset
- Computing Timer Values - Monostables, C
- Multitasking Forth

Issue Number 59:
- Z-Sys Corner - ZMATE MACRO usage
- Moving Forth - Part 1
- Center Fold - IMSAI MPU-A
- Developing Turnkey Forth Applications
- Mr. Kaypro - Versions of Kaypros
- DR. S-100 - Vendors

Issue Number 60:
- Next Ten Years - Part II
- Moving Forth Part II
- Center Fold - IMSAI CPA
- Four for Forth - Forth CPU's
- Debugging Forth
- Z-Sys Corner - 8 years of Z-System
- Mr. Kaypro - Turning a Kaypro II into a IV
- DR. S-100 - Letters

Issue Number 61:
- Z-Sys Corner - Automating GEnie Mail
- Multiprocessing / 6809 part I
- Center Fold - XEROX 820
- QC Using the Commodore 64
- Real Computing - JPEG, WORM, archivers
- Support Groups for Classics
- Operating Systems - CP/M
- Mr. Kaypro - 5 MHz Upgrade

Issue Number 62:
- SCSI EPROM Programmer
- Center Fold - XEROX 820
- DR S-100 - Exploring the S-100 Bus
- Moving Forth part III
- Programming the 6526 CIA
- Reminiscing and Musings - 10th Year
- Modem Scripts

Issue Number 63:
- Z-Sys Corner - Failsafe Scripts in 4DOS
- SCSI EPROM Programmer - part II
- Center Fold - XEROX 820
- DR S-100 - Disk Drives and BIOS code
- Multiprocessing Part II
- 6809 Operating Systems
- IDE Drives Part II

Issue Number 64:
- Z-Sys Corner - Failsafe Scripts in 4DOS II
- Small-C - Review and comment
- Center Fold - last XEROX 820
- DR S-100 - Disk Drives and BIOS - part II
- Moving Forth Part IV
- Small Systems - 6800/6809 History
- Mr. Kaypro - Sign on and Clock Upgrade
- IDE Drives - Part III

Issue Number 65:
- Small System Support - 68xx Serial Comm
- Sinclair ZX81 - Letters and Books
- Center Fold - ZX80/81
- DR S-100 - Christmas letters
- Real Computing - Linux and Linking
- European Beat - AMSTRAD in Europe
- PC/XT Corner - Day-Old Computing
- Little Circuits - Reset Circuits
- Levels of Forth - Selecting a Language

Issue Number 66:
- Z-System Corner - Failsafe Scripts in 4DOS
- Real Computing - TCP/IP and OSI
- Small System Support - 'C' and 68xx
- Center Fold - Advent Decoder Board
- DR S-100 - Spring Letters
- Connecting IDE Drives (IDE part IV)
- PC/XT Corner - Day-Old Computing
- Little Circuits - Battery Backup Circuits
- Multiprocessing Part III

Issue Number 67:
- European Beat - more AMSTRAD history
- Small System Support - 6800/09 programs
- Center Fold - SS-50/SS-30
- DR S-100 - Trenton/Z-Fest & letters
- Serial Kaypro Interrupts in Forth
- Real Computing - Tiny-TCP and WIN
- Little Circuits - Wire and Cable
- Moving Forth Part 5

Issue Number 68:
- Small System Support - Languages
- Center Fold - Pertec/Mits 4PIO
- Z-System Corner II - Intro CP/M and Z-Sys
- PC/XT Corner - A bit of everything - Part I
- Little Circuits - CMOS and RC's
- Multiprocessing Forth Part 4
- Mr. Kaypro - Notes, Repairs, and Macros

Issue Number 69:
- Small System Support - 6809 ASM, Flex
- Center Fold - S-100 IDE
- Z-System Corner II - Intro, part 2
- Real Computing - Tiny-TCP
- PC/XT Corner - Stepper Motors and Forth
- DR. S-100 - Mail Bag
- Moving Forth Part 6
- Mr. Kaypro - Advent Decoder Construction

Issue Number 70:
- Small System Support - 6809 ASM
- Center Fold - Jupiter ACE
- Z-System Corner II - Intro part 3
- PC/XT Corner - Stepper Motors & Forth
- DR. S-100 - Mail Bag
- Multiprocessing Part 5
- European Beat - 8-bit idiot and AMSTRAD

Issue Number 71:
- Computing Hero of 1994 - David Jaffe
- Small System Support - 6809 ASM
- Center Fold - Hayes 80-103A S-100 modem
- Power Supply Basics
- PC/XT Corner - Stepper Motors
- Connecting IDE Drives (5) - GIDE Preview
- DR. S-100 - Generic IDE and CompuPro
- Moving Forth Part 7
- Mr. Kaypro - ROM options
- 8048 Emulator Part 1

Issue Number 72:
- Beginning PLD - good and bad
- Small System Support - 'C' and ASM
- Center Fold - Rockwell R65F11
- Playing With Micros - 5 to learn with
- Real Computing - Languages
- Small Tools Part 1 - Forth, 68HC11
- DR. S-100 - CompuPro 8080/8086
- Moving Forth Part 7.5
- 8048 Emulator Part 2

Issue Number 73:
- $10 XT - what you can get at a swap meet
- Small System Support - 'C' and ASM
- Center Fold - 640K XT
- IDE Part 6
- Real Computing - Linux
- Small Tools Part II - New Micros F68HC11
- DR. S-100 - Trenton and Letters
- PC/XT Corner - software quandaries
- 8048 Emulator Part 3

Issue Number 74:
- Antique or Junk - How to judge your system
- Small System Support - 'C' and ASM
- Center Fold - S-100 Power Supply
- Real Computing - Linux and Minix
- AMSTRAD PCW Now
- DR. S-100 - Mailbag
- Mr. Kaypro - Adding Composite Monitors
- Palmtech CPUZ180 - Review
- Disk I/O in Forth
- Moving Forth part 8

Issue Number 75:
- The European Beat - East German Z80
- Small System Support - 'C' and ASM
- Center Fold - Standard Bus and I/O
- Moving Forth part 8
- Real Computing - Rick moved
- Embedded Control Using the STD Bus
- DR. S-100 - Mailbag
- EPROM Simulator
- High-Speed Serial I/O for the Applicard
- Disk I/O in Forth, Pt. 2
- T9600 Source Code (Small Tools)

Issue Number 76:
- Real Computing - Minix and more
- PC/XT Corner - Bank Switching/Supercharge
- The European Beat - 10 years for user group
- Alternatives to the XT
- DR. S-100 - GIDE and the Jade Bus Probe
- Center Fold - JADE Bus Probe
- PC Time Clock - Improving Accuracy
- PC Security System - Home Security
- Small System Support - 'C' and ASM
- Floppy Disk Problems - design problems

Issue Number 77:
- Mr. Kaypro - External Video
- Hands-on with PLD's
- Center Fold - CPU280
- The First TRS-80
- Program This! - the Z80 SIO
- Small System Support - Prime Numbers in C

Issue Number 78:
- 6502 DIY Board
- Program This! - 8051 Startup Code
- Center Fold - AMR 80552
- Simplex III - Homebuilt TTL CPU
- Real Computing - Small C, C-64, Win95
- Small System Support - C and Assembler

Issue Number 79:
- PC serial port in Forth
- Program This! - AT Modem Commands
- Center Fold - P112 Z182 board
- Simplex III - Homebuilt TTL CPU, part 2
- Real Computing - Real-time Control
- Small System Support - C and Assembler
- Embedded Development Choices

Issue Number 80:
- B/P BIOS, Part 1
- Program This! - PC Serial Port, ASM and C
- Cheap Hard Disk Controller
- Simplex III - Homebuilt TTL CPU, part 3
- Real Computing - Newton, Java, Small-C
- Small System Support - C and Assembler
- High Speed Serial for 65xx

Special Feature

Editorial Comment

The Move to Linux

The Computer Corner

Part 2

By Bill Kibler

This article started out as an email article which is why the format is slightly odd. I plan on converting the TCJ/DIBs BBS to Linux one of these days (in my spare time) and I keep on asking Bill about Unix/Linux. We felt we ought to publish it for everyone.

More Power

Notes on upgrading to Linux

In a recent Computer Corner, I discussed moving from NT back to DOS or ahead to Linux. I was rather unhappy with NT's poor performance, especially items like crashing often or incredibly poor speed when using DOS boxes. Since I do more DOS type operations than windows it seemed that maybe going back to DOS might be an appropriate change. However I use Unix at work and have come to like having many terminal screens open and being able to cut and paste between text screens and so changing to linux sounds better. It will also give me the chance to test how well linux does DOS and Windows (which it can).

I had NT up and running, but tried to do my last article using PageMaker 5 and found several functions to NOT work correctly. I use version 6.5 on a NT station at work and was a bit surprised to see the difference. I think it has to do with changes to NT over Win3.1 API interfaces. The fact that it was different - sort of pushed me over the edge. For me that was the last I wanted anything to do with NT. So next up was backing off all the needed files and changing over to linux.

In my setup, I already have a linux server running with a very large drive. It uses samba and thus works very well as remote DOS file system to save files off to. I started copying files over and ran into a few problems. Linux worked just fine, but first the NT DOS box locked up while trying to delete subdirectories that had been transferred. It ended up sending a prompt message to the printer, which I had to turn off to stop printing even after closing the DOS box.

After dumping my D: and E: partitions, I tried to dump the main or C: drive. I copied about ten files then hit a protected file which stopped everything. The swap or pagefile.sys can not be copied, and thus so much for backing up a drive with it. I thought I could get around this by changing the configuration setup and move the pagefile to one of my other drives, and then tried, just to see how many other files will stop the backup. It didn't work since changing the pagefile location doesn't remove the old pagefile, which can leave an unused 70meg file that can't be deleted! So I just fdisk'd all away after saving some of the directories one at a time.

This is of course one of the other reason I hate NT and Win95, as I was logged in as administrator and as such should be able to do anything I want or need to do. But no, Microsoft knows better and will not let you do many things that most expert users find necessary to do. As part of this ongoing discussion I will explain how to do things in Unix or linux world. This whole topic of who has control is a great place to start explaining things.

In the Microsoft world, they know better than even the administators. In Unix the administrator is who is in charge, whether or not they know what they are doing. It is a two edged sword, you have full control of everything and can change or do anything - including trashing or erasing the whole file system. For those of us who have spent more than a few years working on systems, this full control is absolutely necessary if you are to properly maintain your system. Many times I have had programs, users, and just power failures cause problems that being root solved.

What is "root" you say. In Unix the altimate power user is "root." This person or user has full control over the system. With this power comes the abiltiy to fix anything, setup new users, erase any files, restore file systems, and just be in charge. This also explains why all the manuals are very clear about making sure you log in as a normal user without root access. That way if you make a mistake, the system levels of protection will prevent a major disaster. Since "levels of protection" is a whole topic or article (to be done later), users normally can see or read system files, but not write or erase them (you have flags associated with all files controling who can do what with the file).

To bring this start to an end, what will have happen under linux when it finds a protected file? It skips it. A very simple solution to a common problem. If I don't have authority to view the file, I can't copy it either. But copy will just step over that file and continue on, which is what makes more sense and what I wanted to happen.

>From here I will blow away NT and MSDOS for good and move on to linux as a power user. Stay tuned as I keep

you posted on my search for a better platform and teach you a few things about linux or Unix.

Bill.

P.S.

After talking to Dave at TCJ he asked about the cutting and pasting between terminals that I mentioned in the first paragraph. First I need to make sure you understand that the reference is to Xwindows term sessions (mostly). These term sessions are like DOS boxes where you are running a terminal process against the host system. You have a screen or box or window that looks and feels like a terminal, in fact you can even make it emulate your favorite terminal.

These are all part of the Xwindows feature, which can be found on all the releases of linux. You must install the X windows server for your hardware, and there are plenty of help files and utilities for that (more later). When you boot up linux, you are given a prompt >login: from a terminal session. After logging you in, you start your X session or can have it be started automatically as part of the login process. One of the ways is to run *startx* and there are others depending on which X server you have. Simply put, X is like windows on a PC only a bit more standardized and can work over a network not just on your physical PC (much more on X later).

Lets take the situation where you have a very long path name to some file. In a NT DOS box you must type this in by hand, each character without mistake. Often that can be very hard. In Xwindows, you might have several term windows open, and one of which has this 60 character path name. No problem, move the mouse to the window with the path, hit left button and drag down the path name highlighting it. Move the pointer to the other term session, type "cd " then hit the middle button or both left and right at same time, and the highlighted text pops in place. Simple, straight forward and the same whether or not it is a program or a simple term session.

So I tried to do this on NT and at first it couldn't be done. But I tried a few other things and discovered the pull down menu in the corner. This has the resize, edit, properties options. You select edit and see "mark" which you can click on. Now you can mark the text you want, and hit return which puts the test in the clipboard buffer. You then can go to another DOS box, program, another place in the screen and select the location to put it and select the pull down in the corner if it is a DOS box (no hot key for paste in DOS box) and the string will be pasted.

So not only are DOS boxes in NT slow, but cutting and pasting is almost useless. Now we could say they want it that way, but the point is, in linux or Unix the idea is a terminal session is still the fastest and easiest way to enter data and handle lots of normal items. So the terminal sessions work very well and are easy to use. Most Unix users have several terminal sessions open at once, cutting and pasting between them, quickly and effortlessly. I almost never use graphic oriented programs and thus anything that must be used graphically, by that I mean only by using a mouse, is not very productive for me.

In my column I mentioned that I like SPE, a professional editor. One of the reasons I like this program is it's abiltiy to cut and paste from text file to file without using a mouse (I have mine set up to use same keys as Wordstar). After using Unix for some time, I have found the use of the mouse for selecting just the text you need an exceptable compromise and almost as good as using my SPE which I keep planning on making a version for use under linux.

I have loaded DOSEMU or the DOS emulator on my linux system (actually was there all along, just needed setting up) and given it a try, with SPE no less. SPE worked just fine and I was able to cut and paste between screens. Now I tried this in Xwindows, then tried it with a regular terminal or VC for virtual console.

Linux has 6 virtual consoles (not Xwindows, but keyboard based terminals - think of them as actual terminals - only no serial cables here.. and each can be logged into as a different user as well) you use these terminals simply by doing an alt-f2 or f3 to f6. What you have is multiple sessions or terminals available at all times. The VC's are a bit like starting Xterm session in Xwindows, and in fact a program called "gpm" is suppose to allow you to cut and paste in VC's.

So far I know DOSEMU works a lot like OS2's DOS function in which you have a virtual PC platform, and must load the DOS files you want to run. If you do a DOS -A, it will boot the operating system that is on the disk in the A drive. A configuration file controls the normal DOS parameters and what file name you want loaded. You can load an entire hard drive image, much like *myz80* does. I look forward to playing with the option and will keep you posted as I test more programs.

bdk.

TCJ CLASSIFIED ADS

CLASSIFIED RATES!
$5.00 PER LISTING!

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

Commercial Advertising Rates:

Size	Once	4+
Full	$150	$90
1/2 Page	$80	$60
1/3 Page	$60	$45
1/4 Page	$50	$40
Market Place	$30	$120/yr

The Computer Journal
P.O. Box 3900
Citrus Heights, CA 95611-3900

MAGAZINES AND BOOKS

Historically Brewed. The magazine of the Historical Computer Society. Read about the people and machines which changed our world. Buy, sell and trade "antique" computers. Subscriptions $14, or try an issue for $3. HCS, 3649 Herschel St., Jacksonville, FL 32205.

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

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

Remember, TCJ has all Back Issues available. See the Back Issues page for a list of the articles in each issue.

FOR SALE

TRS-80 - MODELS I, 2, III, IV, 12, 16, POCKET COMPUTER, AND COCO: Software, hardware, internal and external disk drives (360K/720K), hard drive's (both complete and bubles), replacement motherboards, floppy drive controllers, video boards, RS-232 boards, keyboards, and more. Send 4x10 SASE for list.
Pete Bumgardner, Rt. 4 Box 36-H, Fort Payne, AL 35967-9408, (205)845-6581.

FOR SALE: THE FORTH ARCHIVE from taygeta.com on CDROM is available from Mountain View Press, Rt 2 Box 429, La Honda, CA. 94020 Ph: 415-747-0760 ghaydon@forsythe.stanford.edu.

SUPPORT / INFO WANTED

Wanted: Intel SDK-85 documentation. This is a single board design kit with the 8085 CPU, includes a hex keypad and 7 segment LED readout. I have several of these units and would consider trading for interesting older computers. Ron Wintriss, 100 Highland Ave., Lisbon, NH 03585.