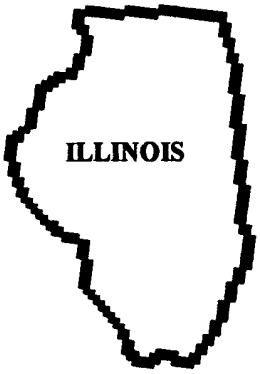
May/June 1997 Vol 4 Number 3 $5.00US (North America)

the world of 68' micros

Support for Motorola based computer systems and microcontrollers [illegible] system



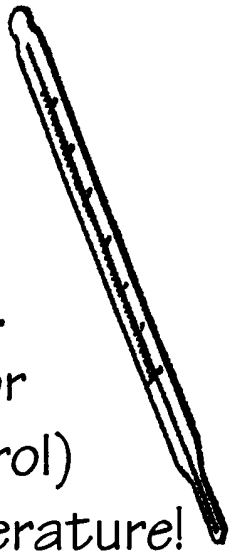
Reviewing one fest just passed...

As we prepare for another!

CONTENTS

COCO3 MEMORY SECRETS REVEALED!

Teach CoCo a new trick... tell (or control) temperature!



POSTMASTER:
If undeliverable return to:
FARNA Systems PB
Box 321
WR, GA 31099

If your address is incorrect, send me a postcard!

the world of 68' micros

Publisher:
FARNA Systems PB
P.O. Box 321
Warner Robins, GA 31099-0321

Editor:
Francis (Frank) G. Swygert

Subscriptions:
US/Mexico: $24 per year
Canada: $30 per year
Overseas: $50 per year (airmail)
Back and single issues are cover price. Canada add $0.50 per issue ($1.00 max) additional shipping, overseas add $2.00 per issue. Bulk/newsstand orders available.

Advertising Rates:
Contact publisher. We have scales to suit every type of business. Special rates for entrepreneurs and "cottage" businesses.

Contributions:
Any and all contributions welcome. Submission constitutes warranty on part of the author that the work is original unless otherwise specified. Publisher reserves the right to edit or reject material without explanation. Editing will be limited to corrections and fitting available space. Authors retain copyright. Submission gives publisher first publication rights and right to reprint in any form with credit given author.

General Information:
Current publication frequency is bi-monthly. Frequency and prices subject to change without notice. All opinions expressed herein are those of the individual authors, not necessarily of the publisher. No warranty as to the suitability or operation of any software or hardware modifications is given nor implied under any circumstances. Use of any information in this publication is entirely at the discretion and responsibility of the reader.

The publisher is available via e-mail at:
dsrtfox@delphi.com

A message from the editor...

Wow! Another CoCoFest gone. Almost seems like it was yesterday. Took me a good two weeks to recover from the trip! Of course, the fact that I had to leave for a weekend trip the very weekend after I returned didn't help anything!

Speaking of the fest, it was another great year! FARNA acquired the Nitro product line from Alan Dekok, and got permission to sell the CoCo 3 emulator from Jeff Vavasour. Those two made it possible for me to actually make a profit again. Without them, the fest trip would have cost me about $125. That's not bad, and I would have been pleased with that amount. I didn't exactly cut corners or go on a budget this year! My wife accompanied me, and we made the trip to Chicago and back a decent vacation for us. I hope to have something new for the next Chicago fest! If you missed Chicago this year, don't worry, as there WILL be another next year!

If Chicago is to far away or just not a convenient time for you, there is still another chance to attend a fest this year. Ron Bull is hosting a fest in Pennsylvania. FARNA plans on being there (baring any intervention of the military!) as well as many other vendors. There is also a good lineup of guest speakers planning to attend! Please see the ad on the back cover for more details.

The entire CoCo community seems to have been shaken up recently. I don't know why, but a lot of former users have been dragging their CoCos out of the closet and starting them back up. Here on my desk I have a letter from someone looking for information on a CoCo2... where to find hardware and software. At first I thought it was the emulators bringing on the resurgence. The emu's certainly account for some of the resurrgence, but not all. Maybe it is just the fact that the clone appliance is just that... an appliance. The CoCo is not... it is an old friend to many of us, an object with a unique personality to others (at least as far as computers go). There isn't a lot of personality in clones anymore. They can be customized some, but they are still pretty impersonal. With a CoCo, you have to learn it and it becomes a part of your life. Almost anyone can buy and use a clone. When was the last time you got emotional about your toaster?

Don't forget the CoCoFest in Pennsylvania! There are over 20 vendors signed up to be there, including FARNA! We hope to see you there too! This fest should be at least as big as the last one in Chicago!
(see ad on back cover for details)

I would like to thank the following people for renewing their subscriptions or becoming new subscribers:

Brian Goers	Bruce Gerst
David Ladd	Daniel Stock
Scott Griepentrog	James Jones
Stanley Scott	John Donaldson
Jack O'Driscoll	Boisy Pitre
Karl Sefik	L. Curtiss Boyle
Carl Daugherty	Howard Luckey
R.C. Smith	Jim Kirby
Tony Podraza	Allen Huffman
Richard Albers	John Schuster
Mark Marlette	K. Kounovsky
Joseph Consugar	Larry Bryant

The following have renewals now due or to be due after the next issue:

Edmund Bassick	Richard Batt
George Bethea	Carl Boll
David Breeding	Paul Zibalia
Hugo Bueno	Earl Casper
Malcolm Cleveland	J.W. Cross
Ralph Ferringer	John P. Francis
Robert Gibon	
Dale Hawley	Mike Holick
Ron McCauley	Ron Melin
Barry Miller	Don Peters
Gene Elliot	Rick Porter
Jason Reighard	R.E. Rutherford
Erik Selelstad	Ken Scales
Wayne Thompson	Stefan Topolski
Earl M. White	Donald Yehling

Messages from our readers...

Whew! I sure am glad that you send an extra issue with a reminder to renew on it. I don't want to miss any issues if I can help it. I meant to have already sent in my renewal, but you know how things sometime get laid aside and forgotten about. Not intentionally, but, none the less, they do. This seems especially true with the renewal date around the end of the year as mine is.

So I want to say THANK YOU for not cutting me off yet and reminding me that I needed to renew.

Keep up the good work! I enjoy the magazine very much. Hopefully before this year is out, I will be able to subscribe to microdisk also.

Larry Bryant

I appreciate all the kind remarks Larry. Believe me, I don't want you to miss an issues either! I realize people sometimes wait until the last minute to subscribe, that's why I send at least one additional issue. In the long run, it is no costlier than sending everyone postcards, which is time consuming also!

From: IN%"chasteen@juno.com"

John Chasteen

Pitfalls Installing the Coco3 Emulator

After 13 years of forgetting RSDOS commands and procedures , I went out a bought a copy of Jeff's coco3 Emulator. First I printed the two Doc files and then sat down in front of the DOS machine and made a coco3 Directory (Windows people say Folder) on my " c " drive. Since I started out with a 5.25 inch, 360 K floppy, I've included a 5.25 in HD floppy in myMSDOS machine.

Just like Jeff Vavasour said in his docs, I copied all the files from my MSDOS floppy into the C:\COCO3 directory. Then I ran the " PORT.EXE " file. Here is where I had to stop and re-read Jeff's doc files again. At the " top of the screen , at about equal distance from each side" I found the flashing cursor. This is where I made my first mistake. I tried to type the file name "GET3ROM.BAS". This kept me busy several hours reading and getting confused. Well after sending an e-mail to Jeff, I entered the MSDOS drive letter that I had placed the " coco formatted " diskette. " B: " (don't forget the colon) ... that's what I typed in the top, center coco field and firmly struck enter. Yep, the flashing cursor jumped to the bottom of the screen to the next field . I typed in the file name "GET3ROM.BAS" . Before this , I went to MS-DOS did a C: <enter> followed by cd \coco3 <enter> and typed DSKINI B: <enter> head " 0 " formatted 39 tracks. Just to make sure, I typed DSKINI B: /2 <ENTER> and watched as head " 1 " also formatted the other side.

Now I went to the COCO3 and placed the diskette in drive Zero and typed Run "GET3ROM.BAS". When the coco3 Floppy Drive had finished writing. I returned to the MSDOS machine and ran the PORT program. The Function keys F2, F3,F7 and F8 had to be " toggled" (switched) to the required position Jeff specifies in his doc. To play it safe, I set the Function keys to: F2 dos to coco, F3 to 0, F7 to BASIC (0) , and F8 to ASCII. I also had to change F2 when I PORTED the files from the coco to the PC.

Hope you have smooth sailing with the coco3 Emulator. I am watching the listserver coco@pucc.princeton.edu for things folks are using their emulators for. Either post your activity or send me an e-mail at chasteen@juno.com.

Thanks to Jeff (jeff@ physics.ubc.ca) for providing us with this program. The price is very reasonable.

John Chasteen
chasteen@juno.com

Well, that about does it as far as installation of the emulator, John... at least as far as I understand it. I am now authorized to sell the emulator, but have yet to find time to install it on my own system. I still have a CoCo3 setup and ready to run at any moment right next to my 486. It doesn't get used much nowadays, but I prefer it over the emulator. But I might have to give up the desk space one day!

At any rate, I'll probably install it at least on my 386 laptop, if it will run on the video. Currently, we know it won't run on any machine with a Cirrus Logic chipset. If anyone discovers a video card or brand of laptop that won't run the emulator, please let me know!

Jeff Vavasour (the emulator author) adds: "It's worth mentioning that the PORT.EXE program has a single line of yellow text at the bottom of the screen which tells you what it is expecting at each step". John's message was passed to him as well as myself.

I will be getting a CD-ROM writer soon. I have started thinking about making a CoCo emulator that will run from the CD with a bunch of CoCo share/free ware ready to run. I may need some help collecting it in .DSK format. If you would be willing to help archive software. let me know. Someone has written a program that would convert the .DSK files back to CoCo compatible files, so the CD may be good for CoCo people also!

The Sixth Annual "Last" Chicago CoCoFest

A review of the top CoCo & OS-9 event of the year!

by F.G Swygert

What can I say? Glenside Color Computer Club of Chicago pulled off yet ANOTHER outstanding CoCoFest! Better yet, plans are being made for another one next year! If you didn't get to the 1997 fest, do make plans to attend the 1998 edition. It could very well be the real "last" one! Of cousrse, that's what we thought about this year's fest also. The bottom line is that as long as there is enough attendance and volunteers to help in the area, there will continue to be CoCoFests in Chicago. So those who have been coming, keep attending! For those who haven't, try to come next year!

Speaking of attendance, it seemed to be pretty good this year. My guess is that there were around 150 people present. This was enough to make the show worthwhile, though all vendors would like to see more!

There were fourteen vendors present with lots of software and hardware to choose from, both new and used. There wasn't a lot of new items, but there was plenty for all to buy. All vendors appeared to be pleased with the turnout. Sales ranged from "as expected" to "much better than expected". No one was disappointed, which is always a good sign!

I'll quickly run through the vendor list as it appearred in the Glenside program. This way I can't be accused of playing favorites! I'll also ad some of my own comments about each.

Alan Dages

Al Dages always has a horde of used CoCo items available. This year he had several slightly used and new CoCo 3's in the original boxes. Prices were from $25 to $70, ranging from Korean made CoCo3s to American 512K models. The Korean models bring less because they have more timing problems than the American manufactured models. There are fixes for the problems, of course. The reason seems to be better quality of parts used in assembly and better quality control. If you need a spare CoCo, call Al at 404-469-5111 before he runs out!

Adventure Survivors

This is a CoCo adventure game club run by L.E. and Nan Padgett. Annual memberships and back issues were available at the show. There are reviews, tips, and maps for the best CoCo adventure games in the back issues. New tips and hints are constantly being found by the adventure game fanatics who subscribe, so if you are an adventure game person yourself, you may want to drop the Padgetts a note and subscribe! They sell a few adventure games, both text and graphics, that are not available anywhere else!!

Adventure Survivors
24 Perthshire Drive
Peachtree City, GA 30269
Phone 404-487-8461

Cloud-9

Mark Marlette is a new vendor in the CoCo community! He has come up with a SCSI interface which he should be advertising soon. Unlike other CoCo SCSI interfaces, this one supports parity and can be used with ZIP drives! In fact, that is the main reason he designed it! A working prototype was displayed along with a running ZIP drive on a CoCo. Since I'm adding a SCSI interface to my PC soon, I just may get a SCSI ZIP and Mark's SCSI interface for my CoCo. So I can share the drive between both systems, just swap disks!

FARNA Systems

This would be myself! I think I sold a copy or two of everything... CoCo Family Recorder (DECB and OS-9 versions), Tandy's Little Wonder, Mastering OS-9, and even a copy of ADOS 3! But the biggest sellers were Alan Dekok's Nitro software and Jeff Vavasour's CoCo 3 Emulator. These two items are what made this year's fest so great for me! 50% of all proceeds go back to Alan and Jeff, so a lot of encouragement was sent to them in the form of dollars to continue their work! There were some "bugs" found in Nitro, but they were packaging problems, not actual bugs in the program modules. FARNA is the only authorized distributor of Nitro and Alan's other programs. I am also authorized to sell the CoCo 3 emulator, along with Rick Cooper. Many people also renewed during the fest, and we gained a couple additional subscribers. All renewals and new subscribers are listed in the front of the magazine.

FARNA Systems
Box 321
Warner Robins, GA 31099
Phone 912-328-7859

Glenside Color Computer Club

Glenside was, of course, our main sponsor! Without Glenside, there would not be a fest.. the last five, for that matter! Back issues of the club newsletter, the "CoCo~ 123", were free until they ran out (a couple issues ran out, but there were plenty for everyone!). A few mugs and buttons were available, and nearly all the vendors bought T-shirts. That was my first purchase... as soon as enough money came in my wife took it and purchased our manditory T-shirts for the year!

HawkSoft

Chris Hawks has been a CoCo vendor longer than any other at the fest, I think. This is where one finds Plug & Power software for your CoCo 3, keyboard extender cables, hi-res joystick interfaces (switchable between hi & regular so you don't have to unplug them!), and several other pieces of hardware and software. If you have an MW1 or AT306, Chris also has a few things for you! He currently has a networking package that uses PPP and SLIP to connect the MW1 or AT306 to the internet or another computer!! He connected my display AT306 to his across the room and transferred some files as a demonstartion.

HawkSoft
28456 SR 2
New Carlisle, IN 46552
Phone 219-654-7080

Justin Time

Justin Wagner stumbled across a large cache of used CoCo and general computer items. He was selling these "things" under the Justin Time banner. If you are looking for a piece of hard to find hard or software, try Justin!

Justin Time
Box 82
Glen Ellyn, IL 60138
Phone 630-620-9370

Luckey Corner

Howard Luckey and Carl Boll teamed up to design and IDE interface for the CoCo. They will soon be available (see sidebar). A working prototype was shown.

Monk-OWare

Brother Jeremy is the official CoCoFest monk... and he REALLY is a Monk!! One of his hobbies has become the CoCo. He is responsible for resurrecting some CoCo oldies, and preserving a lot of CoCo history. When a man of the cloth tells you he isn't doing something for profit, he is usually believed! Rick Ulland of CoNect works with Br. Jeremey a lot and shared his booth. The Fast 232 pack was avaialble as well as a lot of used items.

continued on page 17

CoCo Temperature Interface

Mike Guzzi

Get your CoCo to do something useful!

I've been working on a project to use a Color Computer 3 as a controller to maintain a proper environment in reptile cages. Part of the controls is to sense the temperature inside the cages. Radio Shack sells a thermistor which is basically a resistor that varies with temperature. The Color Computer has analog joystick ports so I figured I would use them to sense the temperature.

The Color Computer 3 has a built in 6 bit A/D converter. This means it can sense 64 steps of voltage. The maximum voltage is 5 volts so dividing that you get 0.07938 volts or 79.3mV/Step. (that's 5 volts divided by 63)

The thermistor that Radio Shack sells gives a chart of its resistance verses temperature. If you plot this, the curve is not straight, however if you use a voltage divider for the thermistor, the curve becomes more linear. Since the CoCo can sense 64 steps of voltage I picked a range of 50 to 114 degrees for it to sense.

The first thing we need to do is to amplify the voltage from the thermistor so each degree change will match the 79.3mV of the joystick port. The second thing we need to do is zero it off at 50 degrees. I used two op-amps to accomplish this. The first one amplifies the voltage to match the joystick port, the second will zero off the voltage at 50 degrees.

Here is a parts list and cost for building this circuit.

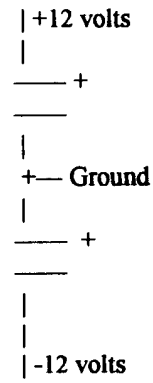
Part	RS Catalog #	Cost
Thermistor	271-110	$1.99
10k resistor (2)	271-1335	$0.49 (pkg of 5)
1k resistor (4)	271-1321	$0.49 (pkg of 5)
47k potentiometer (use instead of 25k shown in drawing)	271-283	$0.49
10k potometer	271-282	$0.49
1458 dual op-amp	276-038	$0.99
741 op-amp (2) (if you don't have the 1458)	276-007	$0.79
47uF capacitor (2) (replaces 33uf in diagram)	272-1027	$0.69
6 pin DIN	276-1474 (CoCo joystick jack)	$1.49
PC Board	276-150	$1.19
total cost		**$8.31**

Now you're probably wondering something, Op-Amps require a source of 12 volts, both positive and negative. Fortunately the CoCo3 has them inside, we just need to steal it from inside the CoCo3. Here is a list of voltage and where you can get them from inside the CoCo3 NOTE: this will require opening the case (no biggie, no CoCo3 these days has a warranty!) and soldering wires to the motherboard to steal the needed power.

Voltage	Location
+5V	Junction of R66 and C15 (large trace)
GND	large trace that connects to shield around outside
+12V	C29, positive terminal
-12V	junction D4 & C31, negative side.

NOTE: you can't draw a lot of current from these sources, but for running two op-amps, it should be OK, I never had a problem doing this. Also, don't use the 5 volts from the joystick port since it has a dropping resistor.

Now build the circuit as the diagram shows. You will need the two capacitors to filter the 12 volt supply. To make the picture fit into the space, I omitted it. It simply looks like this



This is needed to filter any noise from the power supply since the voltages are unregulated. The output of the second op-amp goes to the X input of the joystick port. I would run the ground wire to the joystick port to make sure the signal doesn't bounce around.

Now that you have the circuit, it's time to calibrate. To do this you will need a digital multimeter since we need to measure voltage accurately. The points marked "TP1" to "TP3" are test points.

First hook everything up and make sure it works as-is (no smoke out) and connect the meter to TP1. Try not to let the thermistor vary much during this test. we want the first op-amp to multiply the voltage by 3.12

So lets say you measure 2.28 volts at TP1, this means you want to read 7.11 volts at TP2. Connect your meter to TP2 and adjust the 47k potentiometer (used instead of the 25k shown in drawing) until you read 7.11 volts (by my example). Go back and forth between TP1 and TP2 to make sure you have it set up. OK, the second part is easy, we need to subtract off 5.58 volts from the first amplifier so that at 50 degrees F, we get a zero voltage reading. Connect your meter to TP3, Adjust the 10K potentiometer until you read 5.58 volts. Now it's ready!

NOTE: Some joystick ports have a voltage offset of about 0.38 volts from my experiments, if you constantly have a higher temperature reading you will want to ad-

From: *Dennis Bathory-Kitsz*

Hi folks! I've been hiding out in Vermont, but since it's the 10th anniversary of my company Green Mountain Micro's demise, I thought it might be time to put in an appearance here.

About 150 copies of 'Learning the 6809' (book only) remain, which I'd be happy to offer at $10 postpaid to anyone interested. If at least 10 people also want the original tapes, I'd be pleased to make up a set of those as well.

One of these days I'll tell my own tale ... amusing indeed...

Dennis Bathory-Kitsz
RD 2 Box 2770
Cox Brook Road
Northfield, Vermont 05663

<bathory@maltedmedia.com>
Malted/Media:
http://www.maltedmedia.com/

just the voltage at TP3 to read 5.20 volts instead.

As far as the programming, its easy, we just need to add the offset to get the actual temperature. An example using Color Basic would be:

T=JOYSTK(0)+50

T will be the temperature in degrees F.

OS-9 uses a system call to read the joystick ports. You must make sure the hires adaptor is disabled for this! If you use a mouse on the right joystick port, its no problem to put this project on the left port.

To read the joystick port, we will use the SS.JOY system call. Examples will be in BASIC09.

```
TYPEregisters=cc,a,b,dp:byte;x,y,u:integer
DIM reg:registers
dim temp:integer
(* see pg 8-116 of OS9 Book for details *)
regs.a:=0 \(* standard input *)
regs.b:=$13 \(* SS.JOY call *)
regs.x:=0 \(* 0 is right port, 1 is left port *)
run syscall($8D,regs)
temp:=regs.x+50
print temp
end
```

NOTE: to insure the port your using is low-res, use the setstt call SS.GIP to setup the port as low-res. (page 8-147)

Accuracy of this circuit is pretty good around room temperature, my experiments show that the error increases to +/- 3 degrees on the outer edges of the sensing range. This circuit is not accurate enough for what I need to do, but for general temperature sensing, it works well.

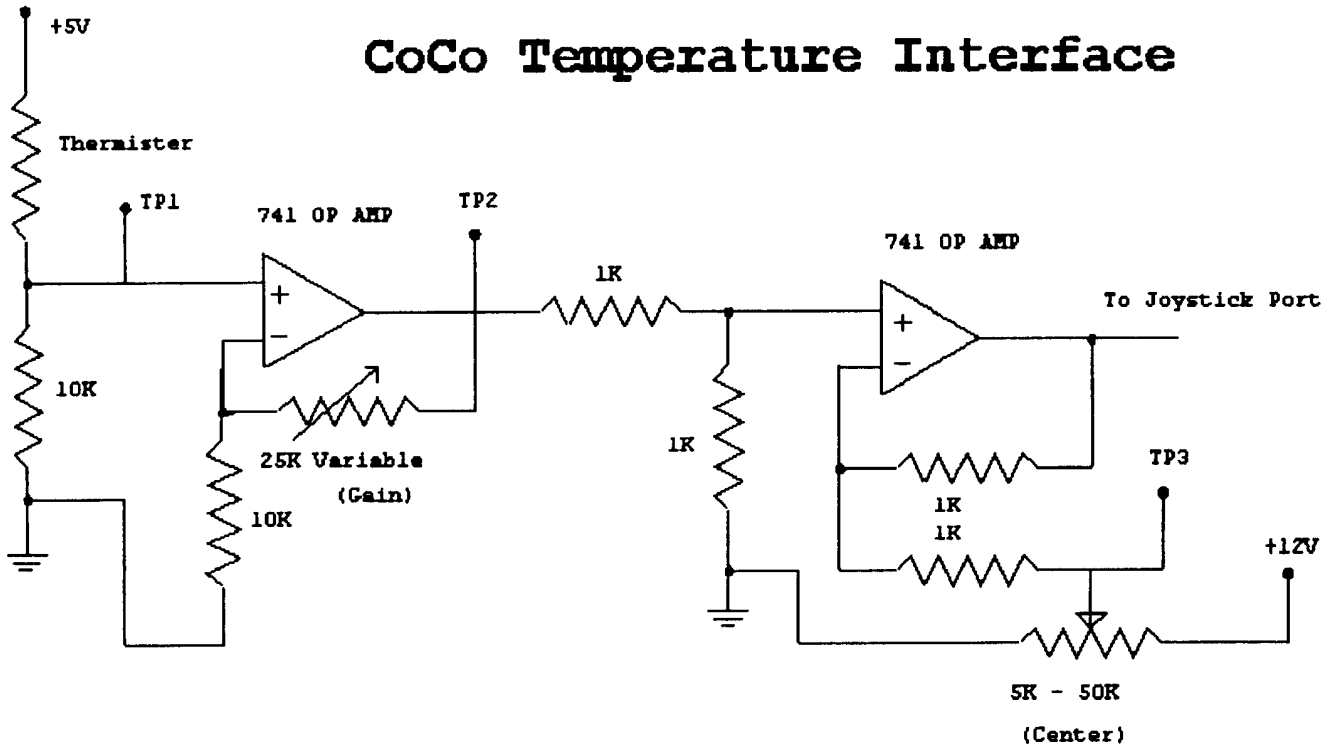
Need to change the range it senses? That's pretty easy, you simply need to adjust the centering voltage. I must note the thermistor curve gets very nonlinear outside the ranges i used, that's why the outer ranges start to get an error reading.

I'll toss this in as a add-on. There is a IC chip, the LM34D which is a temperature sensor that is linear and very useful for sensing accurate temperature. I am using this chip with a 8 bit A/D converter. Radio Shack does not sell this part. I bought them from Digi-Key. They cost $2.85 each for the ones that sense 32-212 degrees. The other one, the LM34 has a much larger range, but its also larger cost. Assuming you want to use the LM34D its easy to figure it out, It outputs 10mV/degree F, therefore at 70 degrees the LM34D outputs 700mV. Adjusting the circuit i described above is easy.

Eliminate the voltage divider that the thermistor uses, just feed the voltage from the LM34D right into the first OP-AMP. Now we need to alter the gain to set it up for the joystick port. Since the LM34D outputs 10mV/F and the joystick port is 79.3mV/Step the gain needs to be 7.93 In order to get this gain with the described parts, you either need to change the 10k resistor from the (-) input of the op-amp to 4.7K or increase the 47k potometer to 100k.

The center voltage needs to be changed to 3.62 volts (assuming the .35 volt offset of the joystick port) otherwise set the center voltage to 3.97 volts. Digi-Key's number is 1-800-344-4539 (http://www.digikey.com). The catalog number for this part is LM34DM-ND. They have no minimum order, however if your order is under $25, there is a $5 handling charge. (I never have a problem making the $25 order to avoid the charge :)

Well, enjoy! Any questions e-mail me devacon@microserve.com or devacon @epix.net



For Temperature sensing range of 50 to 114 degrees F

Gain = 3.12 (TP2/TP1=3.12)
Center = 5.58v (TP3)

AT306 Trials and Tribulations Part 2

Frank Swygert

SUCCESS!!!! Finally, most of the "bugs" are squashed!!!

Sometimes, the wait is worth the effort. Setting up the second full AT306, software and all, was a breeze! Not that there weren't any minor problems, but they all resolved easily! But I'm getting way ahead of myself... better go back to the last article and see where that left off.

To recap, we had a lot of problems with the video drivers. Carl has discovered the main culprit: the Trident BIOS wants to access a register in the EGA portion of a PC clone's BIOS, something the AT306 doesn't have. The short work-around is to install the card in a PC first. It can then be pulled and will work from then on in an AT306. There must be some small amount of flash ROM built into the Trident video chipset, as there are no other chips except the video RAM on the boards. Carl has "set" a board like this for me then sent it back through the mail. I waited 2-3 days before putting the board in an AT306. It worked fine. So the settings aren't just staying in the VRAM, it shouldn't be able to retain memory for several days! So the video card problem is solved for now. Carl is investigating just what the Trident BIOS wants or needs from the PC BIOS so that cards won't have to be initialized in a PC.

So there is now a bootable disk and a working video card. If one orders a motherboard kit from FARNA, an initialized video card will be included with a boot disk set for the included video card. The motherboard switches will be set to boot from the video card, not a terminal. An addition will be made to the AT306 manual noting that the video card must be initialized in a PC first. If a video card is purchased from a local vendor, get them to "test" it in one of the shop PCs first. Mail order might be a bit trickier, you'll need to install the card in a PC first or get a local shop to test it before it will work correctly.

When a motherboard kit arrives it just needs installing into a PC case. Connect the power supply and a 3.5" high density (1.4M) floppy drive, install the video card, and plug in an AT keyboard and a VGA monitor. Slip the included disk in the floppy drive and turn the power on. The system should come to life in 20-30 seconds. When it does, the power-on screen should come up with a nice screen with the Trident TVGA BIOS message, then a box containing the following messages:

```
Kreider Electronics AT306 Support
ROMS V1.04

Keyboard Controller OK

4096K RAM
Found VGA Card
OS-9/68K System Bootstrap
Attempting to boot from floppy drive
Attempting to boot from hard drive
Booting from non-contiguous Bootfile
```

The box and test messages are in the V1.04 ROMs only. Older machines will not display these messages.

Another interesting feature in the new ROMs is the ability to boot from the hard drive without checking the floppy first. Simply turn DIP switch 6 on! When this feature is enabled the computer will not attempt to boot from floppy or display the related message. FARNA ships all machines and motherboards set to check the floppy first. This adds a few seconds to the boot time, but prevents having to open the case and change the switch if there is a problem with the hard drive or a temporary boot is desired later. These ROMs may be used in all AT306, WCP306, and MM/1B computers. Contact FARNA for upgrade info.

It should be safe to assume that most users will be installing a hard drive. The AT306 has a built-in ATA/IDE compatible hard drive controller. Gigabyte hard drives are increasingly inexpensive, averaging around $180. In fact, this has caused a bit of a problem for FARNA Systems! It seems that IDe drives under 1.0GB are no longer being manufactured. There is still a good supply of new 300-540MB units on the market for now, but they are commanding a good price... around $140. For $40-$50 over the price of a new 540MB drive, one can purchase a 1.0GB drive! But a lot of PCs are still in use that don't directly support drives larger than 528MB. Corporations who use these usually replace bad units with drives of the same or slightly more capacity rather than use drivers to install larger hard drives. Or they add a second drive. This demand keeps the prices for existing small IDE drives up, except for drives under 100MB. A new 130MB drive, for example, runs around $85. Of course, OS-9/68K is still rather efficient when compared to Windows machines. All included AT306 utilities, OS-9/68K, Microware's K&R style C compiler, and the MGR graphic interface only take up a bit less than 23MB. That leaves over 100MB on a 130MB drive! With the small size of most OSK (OS-9/68K) applications, that leaves a tolerable amount of space free. With the scarcity of 200-400MB IDE drives, FARNA will be offering 130-170MB drives in the bargain systems and 1.0GB drives in the more expensive models. Then users can add a second hard drive or upgrade to a larger one later. The other alternative is to install used or refurbished drives, something I really don't want to do, although I have had good luck with them personally. Installing a used or refurbished drive should be a choice of the user.

Making the hard drive bootable is easy. For starters, I had the foresight to install the hard drive drivers and descriptors on the boot disk (for two hard drives, /h0 and /h1). The descriptors are edited for the installed drive specs and the computer is rebooted with the new descriptors. Then simply use OS9Gen to make the hard drive bootable and copy the system disk to the hard drive. Boot lists are also included on the floppy boot disk. This keeps problems to a minimum!

There is some considerations before installing a hard drive. OSK has a 64K bit map size. Using 512K sectors, this means that the disk size limit is 256MB. This limit can be increased in onw of two ways: increasing cluster size or partitioning the drive into two or more logical drives.

The normal method of formatting a hard drive is to use one sector as the

minimum allocation unit. Microware calls this unit a "cluster". Basically, one cluster is the smallest unit that the drive can allocate for a file to use (hence the name minimum allocation unit). So even a 256K file will take up 512K on a normally formatted hard drive. The other 256K is basically wasted space.

The cluster size does not, however, have to be 512K. Microware allows the cluster size to be increased in increments, doubled every time it is increased. So a single cluster can be 2, 4, 8, etc. sectors, or 1024K, 2048K, 4096K, etc. This allows hard drives to be as large as 512MB, 1024MB, 2048MB, etc., respectively. This is accomplished by using the -c=<#> option of the format command. Replace "#" with the number of sectors (bearing in mind that this number must double for every increase) to be used in a single cluster.

There is a performance penalty as well as a space penalty when using large cluster sizes. OSK takes longer to read the larger sectors. Not a lot longer, but when reading lots of small (under 1MB) files, it is noticeable. Remember that little 256K file? Guess how much room it takes up (and how much is wasted) when cluster sizes are 1024K or more!

The most efficient use of hard drive space is to use partitions. A large hard drive, say 1.0GB, is divided into four 256MB logical drives. As far as the operating system is concerned, it is addressing four separate drives instead of a single large drive. Some space will be lost due to the partitioning, but not as much as using large cluster sizes. One will need to brush up on some math to properly partition an OSK drive though!

For starters, let's go over the basic principle on how a drive is partitioned. It is really simple: the starting point on the physical drive is different for each partition. OSK always starts formatting a drive at sector 0. So what we do is tell it to start somewhere else! This is done by setting the PD_LSNOffs number to the first sector number to be used in a partition. The examples that come later illustrate this better.

There can be up to 16 partitions since the least four bits of the base address are used for the partitions. Since OS-9 views a drive as a linear array of clusters, the physical layout of a drive is of little concern. OS-9's format command does want some kind of parameters, so fictitious ones are used. Any number of sectors per track and number of heads can be used (the number of cylinders is determined by the size of the drive... the examples illustrate this concept best!). The number of cylinders in a partition is set in the descriptor with PD_CYL. D_TotCyls is the total number of cylinders for the entire drive. PD_SID is the number of sides (heads) and PD_SCT the number of sectors per track. PD_T0S is the number of sectors for track 0. Normally, this is set the same as PD_SCT. The PD_T0S setting is there for backwards compatibility with old drives that used a different number of sectors for the first track. These settings are changed using the DMODE command.

Examples!

As noted in a couple paragraphs above, the best way to illustrate how partitioning works is through a couple of examples. The necessary math is also explained.

Example 1

This is Chris Perrault's Western Digital 2.5" drive. The drive has an unformatted capacity of 540MB using 16 heads and 63 sectors per track. There are 1048 total cylinders.

I decided to partition this drive into three parts: one 40MB and two 250MB. The first thing to do is figure the number of sectors for the first 40MB partition: desired MB x 2048 / (sides x sectors per track) = cylinders for partition (2048 is the number of sectors in one megabyte). 40 x 2048 / (16 x 63) = 81.27

A partial number of cylinders can't be used, so 81.27 is rounded down to 81. The dmode parameter for device /h0a will be:

sid = 16 cyl = 81 sct = 63 t0s= 63 totcyls=1048 lsnoffs=0

The second partition will be 250MB. 250 x 2048 / 1008 = 507.94

(1008 is 16 x 63. Since the number of sides and sectors per track remains the same this can be used as a constant from now on). This time I rounded up to the nearest whole number. 0.27 sectors were "lost" in the first partition... more than enough to make up for the 0.06 needed in the second partition.

Next the LSN (starting sector number) for partition two needs to be determined. This is done with the following formula: (sectors x sides) x cylinders of first partition = LSN for partition two (total number of sectors in the first partition). 1008 x 81 = 81648 sectors. The dmode parameter for device /h0b will be:

sid = 16 cyl = 508 sct = 63 t0s= 63 totcyls=1048 lsnoffs= 81648

Partition three falls short of 250MB. How can this be? Wasn't the drive advertised as 540MB? Yes... but that is the UNFORMATTED capacity. If the drive could be used without sector location information, it would compute out to a full 540MB. Formatting (marking the locations of the sectors) takes some room, so the drive ends up with a formatted capacity of around 516MB.

This first becomes apparent when we figure the number of cylinders for partition three. The total number of cylinders is 1048. 81 cylinders were used in partition one (/h0a) and 508 in partition two (/h0b). 81 + 508 = 589. 1048 - 589 = 459 remaining cylinders. Had we figured this out before hand, partition two and three could have been made equal sizes. This isn't critical by any means - partitions can be any size as long as the total number of cylinders is not exceeded. Therefore, the last partition comes up to approximately 226MB.

Since the number of avaialable cylinders is known for the last partition, all we need to do is compute the number of sectors in partition two and add them to the number of sectors in partition one to find the offset value for partition three:

1008 x 508 = 512064 sectors
512064 + 81648 = 593712
dmode /h0c sid = 16 cyl = 459 sct = 63 t0s= 63 totcyls=1048 lsnoffs= 593712

Now that the descriptors have been modified, a new boot file needs to be made. The preferred method is to make a floppy boot disk that has the floppy as /dd with the new descriptors. Do this by making a new bootlist then using OS9Gen:

continued on page 14

operating system nine

Rick Ulland

The operating system

In the grand scheme of things, two kinds of desktop operating system have evolved- at one end, we have a single tasking, relatively simple bit of code designed to run one specific computer. At the other, a complex system of related utilities designed to work with some sort of real-time (or at least time sharing) kernal.

Examples of a simple system would be the CoCo's rom basic, or at the upper end of the range, MSDOS. The timesharing systems are more exotic, ranging from simple unix clones to minicomputer opsys like VAX/VMS. The gulf between the two types of system is wide and they normally have little in common.

Expanding a 'simple' system into something more complex is extremely difficult to do well- witness the problems MicroSoft has had with Windows. Shrinking a big multiuser system to fit a small computer is also near impossible, so small systems generally are limited to the small opsys they were shipped with.

OS9 doesn't fit into this model at all. It's primary use has been as a real-time opsys for controllers and consumer electronics. Here, concurrent execution is desirable to simplify the application writers job. With each process truly insulated, both from the world outside and each other, a series of simple programs, each handling a limited range of events, can be ran *concurrently* (at the same time) and depend on the opsys to notify any that have a piece of real-world data to handle, and to keep all the separate bits of code running in sync The writer is freed from the need to carefully time each subroutine or devise a new scheduling scheme- he doesn't even need to write the task up as a single program.

For this to work, a pretty complex layer of hardware and software signalling and scheduling algorithms shuffle the order that processes are called, reflecting what is happening outside the cpu. This level of task switching isn't all that common among small computer opsys, but it's far from unique- there are more than a few 'real time' controller kernals out there.

What makes OS9 different is *scalability.* The designer isn't limited to the small real time kernal, and he doesn't have to kludge all kinds of potentially incompatible driver software on top of it to get things done. Instead, there is another layer to be added, starting with the manager. All of the various types of data that could be fed into a computer can be grouped into a few main classes- for example serial ports, keyboards and monitor screens are similar in that the data is sequential. It's a simple list of words and the logical handling will be pretty much the same despite the act the code needed to operate the devices will be quite different.

A second major class would be data that arrives in 'random blocks'. A disk drive is a good example. Even though each sector is presented as a stream of data (computers are, after all, serial devices) it can be managed as discrete blocks. There are other types of data that don't fit neatly into either class, for instance pipes are handy when the source and the destination aren't operating at exactly the same speed. This is something that SCF isn't quite ready for, so PipeMan is needed to handle the stuff that backs up in the pipeline.

With a manager to rely on for logical functions, the individual drivers only have to deal with the details of operating their specific hardware. They can then dump nearly raw data in the manager's lap, saving much duplication of effort in the drivers themselves.

The manager scheme simplifies hardware drivers in another way- they provide a boundary layer between the complexities of multitasking and the simple job of working a peripheral. The driver can be written as if it's working only one piece of equipment with exclusive access to the computer. The concept that allows this to work is reentrancy.

Simply put, the data and scratch values used are stored outside the driver itself. Which brings us to the last layer. Each piece of hardware has it's own, unique descriptor, and the drivers scratch data is associated with the descriptor, not the driver itself. The manager can reuse a single task driver many times by simply flipping in different descriptor blocks.

Probably the easiest way to understand the concept is by looking outside the opsys, at the user code areas. When a new process is launched, it gets a 64K opsys sized ram area. Typically, shell is mapped into the first block of this area and it's given a data block at the top of the area- 2 blocks or 16K of logical RAM. But the code part is unchanging- *all* the data shell uses lies in the data block. As a result the code block doesn't have to be a unique copy- the block number of the copy already in RAM is simply assigned to this new process map so we've only used 8K of physical RAM.

This sounds like quite a handful. No matter how cleverly you sneak in a second copy, the code still has to be ran twice. You'd expect adding a few layers of elegant features would slow the system to uselessness. Like Windows 3.0 on a 286. But remember, the code started out to drive small controllers, using very tightly written routines and a kernal designed to trap thousands of realtime events per second. It's fast, and more importantly, it's 'preemptive realtime'.

When the system slows down (which it must, there's only so much a 6x09 can do) the system first ensures that any speed critical events get serviced in about the standard amount of time, then the noncritical things all get whatever's left. Even when the screen has stopped the keyboard buffer still works and the printer keeps moving. Thanks to this trait, many things catch up before it's obvious they've fallen behind. Perceived speed is also helped by the code size. Short code loads faster, and that's half the battle.

Then came CoCo

By now, you're probably lookingat your CoCo and thinking me quite mad. The truth is, there is one more level of operating system at work, *hardware emulation.* A 6809 based machine isn't inherently any cheaper to build than any other computer. The only way something like a CoCo can be made affordable is to do everything in software, using the cpu to emulate the missing interfaces. So CoCo has a software uart serial printer port, software A/D converter for the mouse, 'dumb' PIA polled keyboard, and sound that's hardly more than a few bits tied to the speaker.

All these emulations use up valuable cpu time, but the worst offender is the original add-on- a floppy controller that STOPS the cpu rather than buffer it's data. It doesn't matter how elegant you code, it's not going to run well when something keeps turning off the cpu from time to time. Luckily, you can relegate the weak floppy system to backup duty with a hard

The BlackHawk MM/1b

Based on the AT306 board from Kreider Electronics. Features built into the motherboard include:

16 bit PC/AT I/O bus with five slots
MC68306 CPU at 16.67MHz
512K to 16MB of RAM with 30 pin SIMMs (4 sockets)
IDE Hard Drive Interface (2 drives)
360K-1.44MB Floppy Drive Interface (2 drives)
Two 16 byte fast serial ports (up to 115K baud)
Bi-directional parallel printer port
Real-time clock
PC/AT keyboard interface
Standard PC/AT power connector
Baby AT size - fits standard PC case
BASIC (resembles Microsoft BASIC)
MGR Graphical Windowing Environment with full documentation
"Personal" OS-9/68000 Vr 3.0 (Industrial with RBF)
Drivers for Tseng W32i and Trident 8900 VGA cards
Drivers for Future Domain 1680 and Adaptec AAH15xx SCSI cards
OS-9/68000 Vr 2.4 with Microware C 3.2, Assembler, MW Basic (like Basic09), MW Debug, MW Programmers Toolkit
UUCP from Bon Billson
Ghostscript (software PostScript interpreter)
Many other utilities and tools

Prices start at $400!

BlackHawk Enterprises, Inc.
756 Gause Blvd. #29
Slidell, LA 70458
504-645-0184

HawkSoft

28456 S.R. 2, New Carlisle, IN 46552
219-654-7080 eves & ends MO, Check, COD; US Funds
Shipping included for US, Canada, & Mexico

MM/1 Products (OS-9/68000)

CDF $50.00 - CD-ROM File Manager! Unlock a wealth of files on CD with the MM/1! Read most text and some graphics from MS-DOS type CDs.

VCDP $50.00 - New Virtual CD Player allows you to play audio CDs on your MM/1! Graphical interface emulates a physical CD player. Requires SCSI interface and NEC CD-ROM drive.

KLOCK $20.00 - Optional Cuckoo on the hour and half hour!! Continuosly displays the digital time and date on the /term screen or on all open screens. Requires I/O board, I/O cable, audio cable, and speakers.

WAVES vr 1.5 $30.00 - Now supports 8SVX and WAV files. Allows you to save and play all or any part of a sound file. Merge files or split into pieces. Record, edit, and save files; change playback/record speed. Convert mono to stereo and vice-versa! Record and play requires I/O board, cable, and audio equipment.

MM/1 SOUND CABLE $10.00 - Connects MM/1 sound port to stereo equipment for recording and palyback.

GNOP $5.00 - Award winning version of PONG(tm) exclusively for the MM/1. You'll fo crazytrying to beat the clock and keep that @#$%& ball in line! Professional pongists everywhere swear by (at) it! Requires MM/1, mouse, and lots of patience.

CoCo Products (DECB)

HOME CONTROL $20.00 - Put your old TRS-80 Color Computer Plug n' Power controller back on the job with your CoCo3! Control up to 256 modules, 99 events! Compatible with X-10 modules.

HI & LO RES JOYSTICK ADAPTER $27.00 - Tandy Hi-Res adapter or no adapter at the flick of a switch! No more plug and unplugging of the joystick!

KEYBOARD CABLE $25.00 - Five foot extender cable for CoCo 2 and 3. Custom lengths available.

MYDOS $15.00 - Customizable, EPROMable DECB enhancement. The commands and options Tandy left out! Supports double sided and 40 track drives, 6ms disk access, set CMP or RGB palettes on power-up, come up in any screen size, Speech and Sound Cartridge support, point and click mouse directory, and MORE OPTIONS than you can shake a stick at! Requires CoCo3 and DECB 2.1.

DOMINATION $18.00 - Multi-Player strategy game. Battle other players armies to take control of the planet. Play on a hi-res map. Become a Planet-Lord today! Requires CoCo3, disk drive, and joystick or mouse.

SMALL GRAFX ETC.

"Y" and "TRI" cables. Special 40 pin male/female end connectors, priced EACH CONNECTOR -		$6.50
Rainbow 40 wire ribbon cable, per foot -		$1.00
Hitachi 63B09E CPU and socket -		$13.00
512K Upgrades, with RAM chips -		$72.00
MPI Upgrades for all small MPIs (satellite board) -		$10.00
Serial to Parallel Convertor with 64K buffer, cables, and external power supply -	**NOW ONLY**	**$28.00!!!**
Serial to Parallel Convertor (no buffer), cables, and external power supply -	**ONLY**	**$18.00!!!**
2400 baud Hayes compatible external modems -		$15.00
Modem cable (4 pin to 25 pin) -		$5.00

ADD $3.00 S&H FOR FIRST ITEM, $1.00 EACH ADDITIONAL ITEM

SERVICE, PARTS, & HARD TO FIND SOFTWARE WITH COMPLETE DOCUMENTATION AVAILABLE. INKS & REFILL KITS FOR CGP-220, CANON, & HP INK JET PRINTERS, RIBBONS & vr. 6 EPROM FOR CGP-220 PRINTER (BOLD MODE), CUSTOM COLOR PRINTING.

Terry Laraway
41 N.W. Doncee Drive
Bremerton, WA 98311
360-692-5374

drive, or find a 'nohalt' controller.

Another simplistic interface is the add on serial port. The traditional units were better than the back panel 'bitbanger'- at least they used a sort of UART to assemble bytes from bits and supply handshaking signals. Serial data is slow enough that drastic measures (like halting the cpu) aren't needed, but the ancient design still has the cpu killing a fly with a sledge-hammer- running OS9's realtime event trap for each character that appears.

In a way, these ports serve to show just how good OS9 really is! After the flurry of event calls pulling it away from all the hardware emulations that are pulling it away from the real work, 600-1000 separate real time events per second can be handled- but only by cutting way back on the amount of user code thats executed while the port is active. The fact that it works at all is outstanding, but way to much overhead is spent on a relatively basic function. This too can be replaced- but you don't *have* to.

Relativity

Still, no matter how tightly the kernal started out, after layering in a few dozen added functions the amount of physical ram required to store OS9 approaches the addressing limits of the tiny cpu. The result is a great piece of opsys with no room for applications. OS9 Level One always had this problem- one reason many experts of the day promoted other systems, like FLEX.

But the solution was already present- with processes insulated from each other and communicating with the operating system via system calls, OS9 was already separated into distinct parts. Level two takes advantage of this division by allowing everything to be switched in and out of the cpu's view. Now the opsys *can* occupy the entire 64K available to a 6x09, allowing a larger number of managers and drivers to be installed at the same time. Of course, applications get the same boost and can also grow to 64K each.

Using a small area of common ground for data exchanges, the MMU (Memory Management Unit - part of the GIME in the CoCo 3) simply switches in the proper program as needed. To stretch the concept a bit further, all screen drawing is normally done by one module of the opsys, which is moved into it's own 64K block along with the screen data. Viola- 192K of code per process, all within a 64K cpu address map. All that had to be added was the apps and utils the user needs to do useful work, and the little controller opsys becomes a full-fledged multiuser desktop system.

Unfortunately, the DECB side of the CoCo community tended to do a better job with the utility writing, making CoCo OS9 a system of a few applications massaged by an incredible number of utilities. The user's opsys skills have to be high because so much work is done from the command line.

For the dedicated, the software situation has improved, as applications tha were intended for large office systems (and so very expensive) are now available for a few dollars used. Sure, this old software is a bit dated and little of it uses the point and click so popular today. Still, this stuff has capabilities you wouldn't expect on a hobby system like the CoCo. With the improvements made to the machine over the years, these old warhorses now run on CoCo, and four users on a single Sculptor database can do every bit as much useful work as a 4 PC LAN with site-licensed Paradox. Of course, it's slower, but for data entry intensive applications, not that much slower. In addition, simpler programs written specifically for CoCo have fallen in price to practically free.

Boot Camp

In short, there are reasons for running a CoCo/OS9 system other than the joy of impersonating a system administrator. Over the next months, we'll dive back into the details of performing useful work on your CoCo. For now, lets look at the process of assembling OS9s layers to make the ultimate office bitty box. This has all been covered before, so I'll make it short.

Remember our scalability factor? We don't have One True Machine, where a few config files can be used to describe minor variations in installed hardware. So the initial task is to define the system. Everybody gets to build a boot disk.

Building an operating system yourself implies you know a bit about this system, computers in general, and your babel fish is screwed in tight. And this is the very first thing you have to do after removing the shrinkwrap, probably not the best time to start such a task.

Tandy knew this, and correctly assumed the heavy-hitter manual set wasn't going to sink into the average user fast enough to set up that first system, so they threw together a little program called 'config' to give users an easy menu way to customize their OS9 release. That's all config was ever intended to do, and that's all it does- solving the chicken or the egg problem well enough to assemble a toolbox.

The real tools are buried around mid-manual. The simplest is cobbler *(a cobbler makes boots... duh)*. This utility slaps a copy of the opsys as it exists in RAM at that moment onto a floppy disk. There are a couple of reasons for cobbler- First, there are many system defaults (printer baud rates, serial port setups, etc.) that will probably change. You could set up all these variables after the initial boot using utilities like xmode, but this is sure to increase load time by complicating the startup process. Better to make the changes once, then take a snapshot of the modified system with cobbler. Which doesn't mean you shouldn't write something like a 'config.sys' file with a text editor. You'll need to document what's normal for your machine in case the boot ever has to be recreated. As soon as you figure out how to get a certain modification installed, write the steps up as a shell script. Every time a new change is adopted, tack it on the end of your personal config maker. Should you ever lose a boot completely, you've got a relatively painless way to recreate all your work. And don't forget to make backups of your boot disks! With any luck, you should never have to create another totally from scratch. Doesn't mean you shouldn't be prepared to do so though!

Another reason for cobbler is OS9's complex simplicity (?!?). We've mentioned there is no 'default' version of OS9, and just picking a specific setup (like Tandy did) is no answer. Even with a shell script to tweak all of the adjustable parts, modules may need to be added and so on.

Floppy based systems, for instance, often use a bootable floppy- most of Tandy's programs did this. It's handy if you have a specific job to do, since the contents of a single floppy can be biased towards the one job, saving many disk swaps. Let's say you have a nice 80 column boot with your custom drive setup and so on, and decide Dynacalc should boot that way:

a) load cobbler, dsave, and format from your system disk

b) put a new disk in /d1, and the original program boot disk in /d0

c) format /d1 r "name";cobbler /d1; dsave /d0 /d1 ! shell

Of course, it's not that simple. The ver-

sion of Dynacalc Tandy sold was hard coded for CoCo sized (32x16) screens, and has to be patched. Still, you can see how this might be considered a simple way to create custom bootable program disks.

But there are limits to how far we can walk a boot, even on one machine. CoCo OS9, for example, provides two basic video systems. The normal window system has OS9 provide video remotely, which frees the application from having to store the screen data. The program has that much more RAM space for it's own use- important when the screen can be 32K!

This isn't good for games, because it's slow and because games often conserve RAM and processor speed by using the actual screen image rather than setting up a completely separate data array to store the positions of objects. Since a window is an output device there is no way to read what's on it under the window system. (If you have ever wondered why Basic09 doesn't have a POINT command to read the color of a screen pixel, now you know.) Rather than reinvent computer games a second video system is provided, which maps the screen right in with the application were it can be directly massaged. And we haven't even mentioned the competing hard drive systems or serial port types!

The upshot is you'll need to add things to the main boot, or perhaps make up some different versions of 'your' boot for different tasks. Cobbler isn't made for this, and config isn't complex enough to handle third party hardware well. What to do?

Home Brew

All of the OS9 parts we were discussing earlier are stored on disk, sometimes in multiple versions. The first step is to make or buy the proper module for your hardware- or simply pick from those you already have. In the Tandy release, these are found on the Basic09/Config disk in the MODULES directory. Before attempting to create a custom version of some stock module check here- the work may have already been done. Tandy supplied lots of disk drive alternatives, along with both versions of the windowing system and a few styles of serial port. You'll also get additional modules with many kinds of hardware- and of course there are the improvements made to OS9 itself since the mid eighties. The custom crafter writes or edits together his own modules, which are added to the pool. Which can quickly become cloudy, so the disk files are a little more descriptive than the name of the module contained within- for example ddd0_40d.dd breaks down to default drive, drive zero- forty track double sided device descriptor.

There is a third (fourth?) alternative- modify and save. It's impossible to store every module that could be imagined, and the distributor would prefer to not repeat near identical code over and over. So there are some simple tools to modify existing modules. Tandy provides the most basic- modpatch.

Modpatch isn't exactly the smartest patch utility ever made, but it uses a simple, easy to distribute ASCII text script and will at least check to see if the proposed patch hits it's target code. Being the lowest common denominator, it only changes the copy in RAM- so the simple out is to run it every startup (sloppy) or follow the usual magazine method, running modpatch then cobblering the patch home.

The elegant alternative is to do the modpatch, then save the modified module back in the bootmaking pool. In the past, it was semi-important to keep earlier versions of a module around as new patches often started from some early version module. The 6809 is more, shall we say, mature now. After modpatching your module *(where do they get these names?)* and test flying it, save a copy (as in term_win80.dt). Keep the old version around until you are sure all your software is pleased with the new code. Other tools with savable output include xmode and clones (like dmode, the disk drive adjuster).

Before you can save anything, of course, you'll need the save utility. For whatever reason, save wasn't a part of the main Tandy release, but was included in the developers system. There is a PD version of save, but you may not have to seek it out. Just to show how useful this util is, Tandy had to sneak a copy into it's MultiVue release to complete the installer program. To get this on disk as a separate utility, find the pmpts command on your MultiVue disk, and try:

load pmpts; save save save

confusing, yes. The three saves break down to '(run the command) save (on a module in ram named) save (and store it in a disk file named) save. There. Clear as mud. Tandy hid the save command by merging it in with a bunch of other commands. Not that they can be blamed... a lot of memory is saved that way!

To digress a bit, this is the place to compare xmode and tmode. When a path is opened, it assumes default characteristics from the base descriptor- these are copied out for use by the path. Xmode changes the original base descriptor while tmode changes an individual path's copy of this data. Once you know this it is easy to understand why xmode changes can be cobblered, while tmode changes are really and truly temporary (and why xmode changes won't affect programs that are already running).

Now that you've got a collection of parts, you'll need some way to assemble them into a valid OS9. OS9Gen will create a floppy based bootstrap automagically. There are other, perhaps easier ways to do this (EZGen, KwikGen) but everybody has OS9Gen, so we'll use it.

All you have to do is create a list of the modules you want as a normal text editor file (one name per line), and feed that to OS9Gen:

os9gen /d0 <bootlist

Tandy even supplies bootlists with original OS9 and MultiVue, but these reflect the system as shipped from Tandy. They make a good starting point, but your boot will almost certainly be different. Doing an ident -s OS9boot, you'll get a list of the modules you are currently using. Keep in mind the module name and the filename it's stored under are two different things, and the filename is usually extended to reflect details beyond simply the name of the module inside.

Since most of the important info is *only* present in these disk filenames, you could be reduced to checking CRC values or something- pretty obnoxious. Afraid the only answer is to keep notes or just know what you're doing. *(OS9 is easy to use)* At least the terminal is easy to identify- TERM is the vdg/uppercase term_vdg.dt while Term is the wind/grfint mixed case version.

Most of the remaining will be obvious- for example the module RBFMan is stored in the file RBFman.mn- or at least decipherable (to pick a clock, first match your power cps *(OS9 is easy to use)*). Find the disk file that corresponds to each module in your ident, and create a text file with each filename, one per line.

There are some ordering concerns, for simplicity try to keep the same order as the orginal ident. To keep things simple,

this file should be placed in the MODULES dir alongside all the pieces parts.

To illustrate the concept, let's look at the first major OS9 upgrade most users see- MultiVue. Ignoring it's install script, you'll see it has a bunch of commands (must go in CMDS) a system file (env.file, must go in SYS) and a few data type things (Pick a dir). You'll also see the last of the available graphics systems for CoCo- WindInt. Replacing GrfInt with this and adding a few extra window descriptors (these are like 'buffers' in MSDOS- you have to have plenty to start with 'cause you can't make more) is all that you really have to do to install MultiVue*(OS9 is easy to use)*. This has gotten confusing enough for one last digression!

Video Magik

Remember the two main types of video? VDGInt represents the mapped into application space, directly addressable version preferred by games. It's very fast, but only capabile of one screen by itself. GrfInt is the remotely stored, reachable via system call video system that not only frees up an applications RAM, but allows the multiple window system that makes multiuser ability useful on your average CoCo. (OK, all the 'users' are you, but it's handy having a rack of terminals all connected to one monitor/ keyboard by a switch.)

WindInt is an extended version of grfint, with scroll bars and pop down menus built in- nice boarders too! It's a shame more programs didn't use windint.... Anyway, we have three drivers representing two basic video systems. There is an additional wrinkle to all this- rather than display a window for an application, GrfInt/WindInt has the option of allowing the app to do it's own video, using VDGint. In use, the vdg video appears to be just another window in the que! The only pernality is, you have to have both video systems in your boot.

This brings up the subject of programs that 'require' MultiVue. If you are floppy only and hate waiting on the ICON clicky interface thing, don't dispair! What these programs really need is WindInt. If you want to write a script to set up the window type and maybe the RAM size assiginment, MultiVue programs will run fine from the command line, provided WindInt has replaced GrfInt in the boot. The opposite condition (programs that require VDG video instead of windows) is covered above. You can run Koronis Rift from an icon.

If you are really observant, you'll see one additional graphics module in your MDir. GrfDrv is the actual video screen driver, and you might wonder why it's not part of the OS9Boot file. If you'd like to figure this out for your self, the hint is this- Os9 has to remap by MMU blocks. This is why files are always loaded into separate blocks. Savvy users know there are things that don't need to be remapped, and merge modules that are commonly used together into 8K sized files to save RAM. OS9Boot does the same thing.

Give? Although grfdrv is a part of the opsys, it has to be remapped. Remember our triple map (OS, screen, app) system? Screens are complicated things, and the video driver has to massage a LOT of data in the process of drawing a screen. So a copy of grfdrv gets remapped into the same 64K as the (32K big) screen image. OS talks to grfdrv via system call, then grfdrv does the changes from inside the screens cpu map. Enough video.

Did I say one digression? Sorry, I lied.There is one 'sort of' module in the stock boot that you may want to remove- CC3Go. The idea behind CC3Go is simple- it's the first process started after OS9 is installed in RAM, and it's job is to get a shell fired up, then run the startup file and look for an autoex. After doing all that, CC3Go was designed to remain in RAM, as the process of last resort. If every shell running was killed off, CC3Go would start up another one, and go through the startup/autoex process again. If you are running CoCo as a single user system, this isn't required! The reason is could be needed is system security.

The only way every last shell could be killed is if they are all 'mortal'. A mortal shell can be killed by sending an end of file character (ESC) to the command line- very handy for additional shells created after startup. On a single user system, you can make the original shell 'immortal'- if anything happens to an immortal shell, another one is immediately fired to replace it. This is what CC3go normally does, which means it will never be called on to do the refire a shell and save the day bit. But when it's installed as part of the boot file, there's no way to get rid of it- so it's stuck there taking up valuable real estate in the 64K opsys RAM.

If you move CC3Go out of OS9Boot (into the CMDS directory) it will load as a separate file, temporarily taking up an additional 8K block of RAM, which is released once CC3Go runs. This isn't such a good idea on a 128K machine, where loading autoex might use up all the available RAM- but 512K machines can usually handle the extra (temporary) baggage. So move it out!

The exception would be a secure, multiuser system. Here, you want the machine to boot into login, so users will have to type a password before getting access to anything. This is simple to set up, just rename login 'autoex' and make up the password file. There is a rub- you don't want to let anyone get access to the (superuser) shell that's running login, or the shell that ran startup to ex login.... but any error will drop them right into it!

The trick is to edit CC3Go, removing the -i parameter that appears just after the word 'shell' by covering it with spaces. This changes the initial shell to a mortal

Announcing Nitro Level III!

How many times have you been unable to load a driver due to not enough system RAM? How would you like to have up to 32K of system RAM available? How is this possible?

In effect, Nitro Level III turns the system into a Kernel only process, with 48K or RAM, and 2 IO processes (RBF and SCF), each with 16K of RAM. This is similar to Grfdrv having it's own 64k memory area.

The kernel process contains the minimum modules to run an OS-9 system, and also the descriptors. The RBF/SCF processes contain the IO modules, and the IO buffers.

There are 2 big benefits here:

1 - Both RBF and SCF are not in system memory at the same time, so you save RAM.

2 - You don't have 16K of SCF or RBF modules, so everything up to 16K can be used as device data storage (sector buffers, etc.)

Level III works only with Nitro (all versions). It can be purchased from FARNA Systems alone ($20) or with the latest version of Nitro ($45 for Nitro v2.00 and Level III). See the FARNA ad in this issue for ordering information.

one which can still be used to run startup, login, etc- the only change is any device or error that would abort the boot process will also kill the shell and lock the console. The system is now secure, but you need CC3Go to perform as originally designed- if it's been moved out of boot, a bad login will leave the machine without a command prompt. With CC3Go in boot *(or linked)*, login will restart.

Where was I?

The best order for a bootlist has been cause for much debate. Sometimes, a given ordering will appear to OS9Gen alright but it just won't boot! Simply rearranging the order of the modules sometimes fixes everything. This problem was widespread enough to be given a cutesy name- the BLOB (boot list order bug). Nowadays, we know it's caused by a newt or small dwarf....ummm, some floppy controller chips...ahh, offset between driver and manager... even byte boundaries?... anyway, I usually arrange bootlists after the Tandy/MWare examples, with OS9parts, then first manager, driver, descriptors, driver, desc.... next manager, driver... The sensible order doesn't seem to get blobed as often as completely random arrangements. A short example:

```
OS9p  '.....the core
Init
IOMan
RBFMan '.....start of block devices
CC3Disk                 'floppies
D0
D1
CC3HDisk, BBFHDisk, etc'hard disk
DD
H0
H1
SCFMan '.....start of serial devices
CC3IO          'console WindInt
W
W1
...
ACIAPak, SACIA, s16550, etc.
T2                      'serial port
T3
Printer                 'printer
P
PipeMan                 'pipes
Piper
...
```

Walk Through:

Many folks don't have hard drives, or identically sized floppies- if this is the case you'll have to first make an odd sized copy of most of your old boot disk, to fit the top drive. Boot in /d0, blank in /d1

format /d1 r "bootcopy"

dsave /d0 /d1 ! shell

you'll also want to dsave the Basic09/ config disk to /d1. Now to build the boot disk in the boot drive. Assuming you have a fresh format in /d0, you've loaded OS9Gen, and inserted/chd to the modules dir on /d1, you can type:

OS9Gen /d0 <bootlist

which will create the OS9Boot file. Then put the original boot disk (or copy) in /d1, and then type:

chd /d1;chx /d1/cmds; dsave /d1 /d0 ! shell

which will copy the rest of the boot disk down. Reboot with the new disk. Now, a wrinkle- of you've changed the size of / d0, you'll have to format _another_ disk under the new boot to get the new size. Do so, then cobbler the new bootfile to it. Do the rest of disk copy step again.

Conclusion!

Well, I hope I havent befuddled anyone. Iintended to clear things up somewhat. If everything is still clear as mud, write the magazine or contact me (or the publisher) through e-mail:

pulland@omnifest.uwm.edu

That's about the best way to get ahold of me nowadays!!

AT306 Trials...

continued from page 8

os9gen /hs1 -b=256k -e -z=bootlist (-b=256k sets the buffer to 256k, -e allows a non-contiguous bootfile, -z designates the name of the bootlist to use... a full path can be defined). After this is done, reboot the system with the new floppy. Format the hard drive partitions. Then modify the bootlist with /h0a as dd and use OS9Gen to make the hard drive bootable. Note that any of the partitions can be the boot drive, but only one.

Example 2:

So let's try that again, this time with a different drive, a Seagate ST3630A 631.1MB drive with 1223 cylinders, 16 heads, and 63 sectors per track. Note right away that the number of heads and sectors per track is the same as the first drive! This is typical of IDE drives. In this case, we will reuse some of our numbers from the last example. Make sure these numbers are checked on each drive before partitioning your own!

My first decision was to partition this drive into four parts (there are only four descriptors included with the AT306, but the existing descriptors can be modified). Since it really doesn't matter about partition sizes, I used the unformatted number to divide the drive into one 31MB and three 200MB partitions. If even figures are desired for the partitions, use the total number of cylinders to divide the drive. Forget about how many megabytes each partition will be. If you wish to know, note the number after the drive is formatted. With 16 heads and 63 sectors per track, there are approximately 2.03 sectors per megabyte. Or reverse the equation using the number of sectors to find the number of megabytes:

MB = sectors x 1008 / 2048

Remember, 1008 is the product of the number of heads and sectors per track.

Using 30MB as our first drive, the dmode settings are:

sid = 16 cyl = 61 sct = 63 t0s= 63

totcyls = 1223 lsnoffs = 0

The second and third partitions are the same except for the offset:

sid = 16 cyl = 406 sct = 63 t0s= 63

totcyls = 1048

lsnoffs = 61488 for /h0b (1008 x 61... the number of sectors in /h0a)

lsnoffs = 470736 for /h0c (1008 x 406 = 409248... the number of sectors in / h0b, + 61488... the number of sectors in /h0a).

This leaves 350 cylinders for /h0d (61 + 406 + 406 = 873. 1223 - 873 = 350). The offset would be the sum of the sectors of the previous three partitions, lsnoffs = 879984 (61488 for /h0a, 409248 for /h0b and /h0c). This method should work for ANY OS-9 computer where there is an lsnoffs setting in the hard drive descriptor.

Setting the second AT306 (the 631MB hard drive) was much easier than the first (the 540MB hard drive). Nearly all the possible bugs have been worked out and solutions found for the existing problems. While setup isn't exactly fool proof, it is easy enough for most with a little experience in CoCo OS-9 to set up with no insurmountable problems.

CoCo3 Extended Memory Secrets Part 1

Herber Enzman

Exploring use of the GIME's built-in Memory Management Unit (MMU)

Introduction

This 6 part tutorial series will cover some of the secrets of the COCO-3 and the MMU. There has been plenty of information written on the subject, but not much on HOW to do it. I've spent much of 1994 and part of 1995 researching the subject and performing experiments; and have decided to share this information with the dwindling COCO community. I have included TEXT, TABLES and LISTINGS with each part. Although it is geared more for assembly language programmers; BASIC programmers should be able to figure out how to use the information provided.

This tutorial will run for several parts, so save all the parts to make one large tutorial so that you can get the most of your 128 or 512K worth of memory. The TABLES alone contain information that I haven't found anywhere. This is a "hands-on" tutorial with experiments to try out and DEMOs to run and experiment with.

PART 1 - Block swapping

This TUTORIAL SERIES will help to unlock some of the well kept secrets of the COCO-3, and the MMU. I have found some information on the subject, such as 'this register does this, that register does that'; but very little on HOW to do it! HOW do you swap blocks? HOW do you write to the 80 column screen? WHAT do you plug into the $FFxx registers to get some useful work out of them, other than strange looking screens? I have spent most of the summer of 1994 looking for the answers to these questions and have decided to share this information; because I have discovered that there are others in the same boat.

I make NO claim to be an expert on the subject; the information presented here is a collection of experiments and notes that I have kept while doing my investigations. It is geared more for assembly language programmers over BASIC, because I program 99.9% in assembly (I don't know a MID$ from a LEFT$); but BASIC programmers might be able to figure out how to use this information in basic programs.

The main problem will be in BLOCK and TASK REGISTER switching, because the BASIC interpreter is always "running the show". I'm using a 512K COCO-3, an RGB monitor and EDT/ASM 6309 for an assembler. I'll be using E/A 6309 as an example in this tutorial, due to it's use of block and task register switching, plus it also uses an 80 column screen. Standard DISK EDTASM users can follow along; I've tested the listings for both editors. Also, I have the FIRST version of E/A 6309; so any addresses that I mention for E/A 6309 might not be the same for your version (be fore-warned now).

During this tutorial, the following filename extenions have the following meaning: ".TXT" is the text file; ".TAB" are the TABLES for that part; ".LST" are the listings for that part. SECB = Super extended color basic; E/A 6309 = EDTASM 6309; TR 0 = task register 0; TR 1 = task register 1.

The TABLES and LISTINGS are numbered sequentially, and NOT by PART, so if PART 3 says to use LISTING 1, there is no listing 1 for PART 3; just LISTING 1.

I would recommend that you format a new disk, and just copy the files nessesary to run EDTASM or EDTASM 6309. This will leave you plenty of room for all of the listings included in this series. Then as EDTASM gets modified, you won't have to worry about your backup copy getting changed. At the end of the series, I will demonstrate how to load several files at once, even into blocks that are not in the task register at that time.

Even though I am programming in 6309 code, I kept all of the listings in 6809 code, so the programs can be used by Standard Disk EDTASM users without modification WITH THE EXCEPTION OF LISTING 16. It is the exact routine that I use to auto load EDTASM 6309, and I just didn't have the time to re-write it for 6809 users.

I'll start with the blocks first, for those who are new to the COCO-3. A 512K COCO-3 has 64 8K blocks of memory (numbered from $00 to $3F), that can be used by you with the exception of blocks $36 (reserved for screen memory); and $3C - $3F (where the ROMS are located). If you wrote a complete 'stand alone' program, that used no ROM routines, you could even use the reserved blocks.

Even with the reserved blocks, there are still 59 unused blocks available. Each block can be 'MAPPED' into any of the $FFAx registers and the same block # can be mapped into several $FFAx registers at once. Heck, you could map the same block # into all the $FFAx registers but that would be of little use! I won't get into how the offset is done, because there has been plenty written about that. The use of the same block # into several $FFAx registers can be seen in TABLES 1 & 2.

Lets look at TABLE 2 first, since it is what BASIC looks like at power-up. You will see that 3 block numbers are the same in TR-0 and TR-1. This is because BASIC runs primarily in TR-0 and uses TR-1 when it uses certain commands such as graphics, at which time it will swap to TR-1. The reason for the use of the block numbers twice is that when you swap blocks OR the task register, you CAN NOT swap the one that your program resides in or you will "pull the rug out from under your

program". Block $38 stays because that is where BASIC keeps it's 'TEMPS', and it has to be able to find them. Block $3F stays because BASIC will call routines in SUPER ECB (SECB) so it MUST remain. If the three blocks were not mapped into both TR's, then a crash would be certain. The column labeled 'SECB' will be discussed later.

Now when you start up either standard DISK EDTASM or E/A 6309, the block map will look like TABLE 1. DISK EDTASM only uses TR-0, so TR-1 will still look like TABLE 2. E/A 6309 uses both TR's, so it re-maps TR-1 during it's start-up, to what you see in TABLE 1. It mapped blocks $30 primarily in TR 1 and swaps to TR 0 when it needs to use the ROM routines —— KEEP this in MIND).

Now that you have some information under your belt, the best way to get the hang of it is to experiment further. Switch in different blocks, write data into each block that is unique to that block, then read them back, to prove that each block is unique.

Standard DISK EDTASM users can only safely change $FFA3, with Z-BUG; or you can write routines that will swap the TASK REGISTER and use $FFA8 thru $FFAF. $FFA2 may look tempting, because the edit buffer is there; but remember that Z-BUG resides there too.

E/A 6309 users can safely change $FFAC thru $FFAF ($E0ED - $E0F0) with Z-BUG, or write routines that change $FFA0 - $FFA7. In either case, I strongly recommend that you OPEN your drive doors, just in case a crash occurs. If you have a newer version of E/A 6309, you can check to see where the edit buffer ends by checking $11B3/B4 with Z-BUG. If it contains $5xxx, then you can also use $FFAB for block switching; but if it contains $6xxx, then don't use it because Z-BUG extends into that area. I've heard that there are quite a few revisions of EDTASM 6309 out and most are different; so I can't give you SPECIFIC addresses. Remember, I have the first version (I guess) that was sold thru COCOPRO. Robert Gault (see ad in this issue) is probably the best current source for E/A 6309.

As this tutorial progresses, this information will start to fall into place. You will see the advantage of swapping blocks and TR's; especially when we get to text screens and GFX (graphics) screens.

Next time, I will cover information that I have discovered on the use of the 80 column screen; how to write directly to it; how to have multicolor screens; and how to use the second half of the screen block.

TABLE 1
EDT/ASM 6309 setup for task registers

TASK REGISTER = 0 $FF91 = 0

```
block  register  range
 ( comment )
$38   $FFA0   $0000-1FFF
 ( DOS / EDTASM )
$39   $FFA1   $2000-3FFF
  ( EDTASM plus screen block ($36) )
$3A   $FFA2   $4000-5FFF
  ( EDTASM plus EDIT BUFFER )
$3B   $FFA3   $6000-7FFF
  ( EDIT BUFFER )
$3C   $FFA4   $8000-9FFF
  ( E.C.B. )
$3D   $FFA5   $A000-BFFF
  ( BASIC )
$3E   $FFA6   $C000-DFFF
  ( DECB )
$3F   $FFA7   $E000-FFFF
   ( SUPER E.C.B. plus I.O. )
```

TASK REGISTER = 1 $FF91 = 1

```
block  register  range
 ( comment )
$38   $FFA8   $0000-1FFF
 ( DOS / EDTASM )
$39   $FFA9   $2000-3FFF
  ( EDTASM plus screen block ($36) )
$3A   $FFAA   $4000-5FFF
  ( EDTASM plus EDIT BUFFER )
$3B   $FFAB   $6000-7FFF
  ( EDIT BUFFER )
$30   $FFAC   $8000-9FFF
  ( EDIT BUFFER )
$31   $FFAD   $A000-BFFF
  ( EDIT BUFFER )
$32   $FFAE   $C000-DFFF
  ( EDIT BUFFER )
$33   $FFAF   $E000-FFFF
  ( EDIT BUFFER to $FDFF plus I.O. )
```

TABLE 2
Super E.C.B. table VS. hardware with BLOCKS on power- up.

TASK REGISTER = 0 $FF91 = 0

hardware	SECB	block	comment
$FFA0	$E0E1	$38	RAM (basic's temps)
$FFA1	$EOE2	$39	RAM
$FFA2	$E0E3	$3A	RAM
$FFA3	$E0E4	$3B	RAM
$FFA4	$E0E5	$3C	EXT. BASIC
$FFA5	$E0E6	$3D	BASIC
$FFA6	$E0E7	$3E	DECB
$FFA7	$E0E8	$3F	SUPER ECB

TASK REGISTER = 1 $FF91 = 1

hardware	SECB	block	comment
$FFA8	$E0E9	$38	RAM (basic's temps)
$FFA9	$E0EA	$30	GFX video page
$FFAA	$E0EB	$31	GFX video page
$FFAB	$E0EC	$32	GFX video page
$FFAC	$E0ED	$33	GFX video page
$FFAD	$E0EE	$3D	BASIC
$FFAE	$E0EF	$35	graphics STACK area
$FFAF	$E0F0	$3F	SUPER ECB

TABLE 3
MEMORY BLOCK useage example
(block #'s are in HEX)

block #	used for
0-16	1 RAMDISK (23 blocks)
17	printer spooler program
18-29	spooler storage (18 blocks)
2A / 2B	UNUSED
2C	labeler program buffer
2D	utilities #1
2E	utilities #2
2F	"Format" routine buffer
30	EDT/ASM 6309 edit buffer
31	EDT/ASM 6309 edit buffer
32	EDT/ASM 6309 edit buffer
33	EDT/ASM 6309 edit buffer
34	unused
35	unused
36	80 column video
37	extra video
38	DOS /EDTASM
39	EDTASM
3A	edit buffer
3B	EDIT buffer
3C	E.C.B.
3D	BASIC
3E	DECB
3F	SUPER ECB

LISTING 1
A simple task register swap routine

```
CLEAR   ORCC  #$50     disable interupts
   LDB    $FFAx    get current block #
   STB    SAVE     save it for later
   LDA    #$xx     (xx) = block # you want to use
   STA    $FFAx    set it up for your use
   CLR    TASK     clear $FF91 RAM image
   CLR    $FF91    set TR=0
   ANDCC  #$AF     enable interupts
   JSR    xxxx     do your routine OR
                   a ROM routine here
SET   ORCC  #$50    disable interupts
   LDA    #1
   STA    TASK     set $FF91 RAM image
   STA    $FF91    set TR=1
   LDB    SAVE     get previous block #
   STB    $FFAx    restore it to what it was
   ANDCC  #$AF     enable interupts
   RTS             return to calling routine
SAVE  RMB  1       temp for storing block #
TASK  RMB  1   temp for RAM image of $FF91
    END
```

NOTES:
1) FFAx would be from FFA0 thru FFA7 for TR1 to TR0 swap.
2) FFAx would be from FFA8 thru FFAF for TR0 to TR1 swap.
3) This routine is for a TR1 to TR0 swap. If you want to swap from TR0 to TR1, then just exchange the CLR $FF91 and LDA #1, STA $FF91 instructions.

LISTING 2
Standard DISK EDTASM 80 column setup

```
GO    JSR   $F679
 LDD   #$XXXX REPLACE X's with colors
                  A= fore B= back
  STA   $FFB8  set foreground
  STB   $FFB0  set background
  STB   $FF9A  set border
 SWI
    END
```

LISTING 3
EDT/ASM 6309 test

```
GO    ORCC  #$50    disable interupts
      CLR   $FF91   set TR= 0
       LDA   #$xx   block number YOU want
                    to set
    STA   $E0EC   set $FFAB equiv. in
                  SECB table
    LDA   #1
    STA   $FF91   now set TR back to 1
    ANDCC #$AF    enable interupts
    SWI           DONE
    END
```

LISTING 4
Multiple use block swap routine example.
** DO NOT run this program! It is just an example of a block swapping 'routine'.

```
      ORG   $3418     assemble here
      LDW   #$6000    = OPTION routine start
      BRA   RUN       go do it
      LDW   #$6003    = FUTURE USE entry
                      point
      BRA   RUN
      LDW   #$6006    = FUTURE USE entry
                      point
RUN   BSR   MOVEIT    go move IRQ
                      routine before block switch
      LDD   #$372D    = blocks to switch
RUN2  PSHS  X,Y,U,DP
      LDX   $FFA2     get current blocks
                      in use
       STX   STORE    save for later
      STD   $FFA2     set NEW blocks at
                      this time
      ORCC  #$50      disable interupts
      BSR   CLEAR     go set task register
                      to 0
      LDD   #$E2F8    = NEW IRQ location
      STD   $0F56     tell DOS where it is
      CLRB
      TFR   B,DP
      JSR   ,W        Now do ROUTINE
                      in TR-0
      LDD   #$503B    = original IRQ start
      STD   $0F56     put back for DOS
                      to find
      BSR   SET       go set TR back to 1
      ANDCC #$AF      enable interupts
      LDX   STORE     get the original
                      block #'s
      STX   $FFA2     put back where they
                      belong
      PULS  X,Y,U,DP,PC return to caller
KILL  BSR   MOVEIT    go move IRQ
                      routine
      LDW   #$6000    = start of KILL routine
      LDD   #$372E    = blocks we are going
                      to use
      BSR   RUN2      go do it
      TFR   V,D       put passed error code
                     into 'D'
      TSTB            was there an error?
      BEQ   NOERR     no, the normal exit
                      back to caller
      JMP   $1820     yes, then exit thru
                      EDTASM's error routine
NOERR RTS             back to caller
MOVEIT PSHS  D,X,Y
      PSHSW
      ORCC  #$50      disable interupts
      BSR   CLEAR     set TR = 0
      LDX   #$503B    source address for
                      block move
      LDY   #$E2F8    destination address
      LDW   #$5A      = byte count for block
                      move
      TFRP  X,Y       move IRQ routine to
                      new location
      BSR   SET       set TR back to 1
      PULSW
      ANDCC #$AF      enable interupts
      PULS  D,X,Y,PC  return
CLEAR CLR   $FF91     clear hardware
                      register
      CLR   TASK      clear RAM image
      RTS             return
SET   LDA   #1        **
      STA   $FF91     ** set hardware register
      STA   TASK      set RAM image
      RTS             return
STORE RMB   2         TEMP for original
                      blocks
TASK  RMB   1         RAM image of $FF91
      END
```

CoCoFest 1997....

continued from page 4

Together they are working to bring back the Disto 2MB upgrade, which should be available soon.

Monk-O-Ware
Box 1903
Racine, WI 53401
Phone 414-634-8979
(see ad for CoNect!)

Music Men
Mike Carey, Brian Schubring, and Mike Knudsen are the CoCo music masters! They had MIDI packs and Mike's fabulous UltiMuse software!

Music Men
606 Willowwood Drive, Apt. 106
Carol Stream, IL 60188
Phone 630-260-9514

Q-Box
This was a group that showed some other 68K based systems... Sinclairs! A great piece of computer history...

R.C. Smith
R.C. is another Georgian who always seems to make it to Chicago! R.C. always has a large cache of used hardware and software... make an offer! If you are looking for something, give him a ring!
Phone 404-469-6601

SBUG
Anre Lavelle of the South Bay Users Group in California always comes with a large quantity of general computer supplies. He also has lots of new and used Tandy software and soem hardware. If you need a serial to parallel adapter, 512K, or Deluxe joystick, he probably has some!
SBUG
1251 W. Sepulveda Blvd, Suite 400
Torrance, CA 90502-2677
Phone 310-539-9702

Strongware
The Strong brothers have a lot of games and utility software for the CoCo and MM/1 computers. In fact, they are the number one source of MM/1 games! They will also be carrying Sub-Etha products in the future.
Strongware
Box 361
Matthews, IN 46957
Phone 317-998-7558

The Glenside Color Computer Club is working on building an IDE interface for the Tandy Color Computer. Orders are now being taken. In order to reserve a board you will need to send a deposit of **$15.00 US >IMMEDIATELY<.**

This interface allows you to use a standard IDE drive with the CoCo. It is a stand alone interface. All you need to provide is the drive, a cable and power for the drive. The board does not come with a case. **Price is estimated to run around $45.00 plus shipping and handling.**

If you want one you must reserve one by sending in a deposit right away. These will be produced only the one time so if you don't order one, chances are you won't get one. Send a check or money order (no postal orders) to:

IDE Interface
C/O Carl Boll
6242 S. Menard
Chicago, IL 60638

ADVERTISER'S INDEX

What are you waiting for?

Get your friends to subscribe to the only magazine that still supports the Tandy Color Computer... *"the world of 68' micros"*!
The more people who want the support, the longer it will be here!

FARNA Systems

Your most complete source for Color Computer and OS-9 information!

Post Office Box 321
Warner Robins, GA 31099
Phone: 912-328-7859
E-mail: dsrtfox@delphi.com

ADD $3 S&H, $4 CANADA, $10 OVERSEAS

BOOKS:

Mastering OS-9 - $30.00
Easy to follow instructions and tutorials guide one through learning OS-9. With a disk full of added utilities and software!

Tandy's Little Wonder - $25.00
History, tech info, hacks, schematics, repairs... almost EVERYTHING available for the Color Computer! A MUST HAVE!

Quick Reference Guides
Handy little books contain the most referenced info in easy to find format. Size makes them unobtrusive on your desk. Command syntax, error codes, system calls, etc.
CoCo OS-9 Level II : $8.00
OS-9/68000 : $8.00

Complete Disto Schematic set: $15
Complete set of all Disto product schematics. Great to have... needed for repairs!

"Inside Disto's 2 Meg Kit" : $10
Schematics and explanation of how the 2 meg CoCo 3 upgrade works.

"the world of 68' micros"
That's right, we publish this magazine! Published bi-monthly, just $24 per year US, $30 Canada and Mexico. The only CoCo, OS-9, and OSK support magazine still in print!

SOFTWARE:

CoCo Family Recorder: Best genealogy record keeper EVER for the CoCo! Requires CoCo3, two drives (40 track for OS-9) and 80 cols.
DECB: $15.00 OS-9: $20.00

DigiTech Pro: $10.00
Add sounds to your BASIC and M/L programs! Very easy to use. Requires user to make a simple cable for sound input through a joystick port. Requires CoCo3, DECB, 512K.

ADOS: Most respected enhancement for DECB! Double sided drives, 40/80 tracks, fast formats, many extra and enhanced commands!
Original (CoCo 1/2/3) : $10.00
ADOS 3 (CoCo 3 only) : $20.00
Extended ADOS 3 (CoCo 3 only, requires ADOS 3, support for 512K-2MB, RAM drives, 40/80 track drives mixed) : $30.00
ADOS 3/EADOS 3 Combo: $40.00

Pixel Blaster - $12.00
High speed graphics tools for CoCo 3 OS-9 Level II. Easily speed up performance of your graphics programs! Designed especially for game programmmers!

Patch OS-9 - $7.00
Latest versions of all popular utils and new commands with complete documentation. Auto-installer requires 2 40T DS drives (one may be larger).

NEW ITEMS!!!

FARNA Systems is pleased to announce that we are now distributors of the following, formerly from Northern Exposure! Note: If you never received your order from NX, send a copy of yourcancelled check along with $5 to cover S&H and I'll fill the order!

NitrOS-9 : $35.00
A complete rewrite of OS-9 Level II that takes advantage of all features of Hitachi's 6309 processor. Easy install script! 6309 required.

TuneUp : $20.00
If you don't have a 6309, you can still take advantage of some of the Nitro software technology! Many OS-9 Level II modules rewritten for improved speed with the stock 6809!

Thexder OS-9
Shanghai OS-9 : $25.00 each
Transfers your ROM Pack game code to an OS-9 disk! Please send manual or ROM Pack to verify ownership of original.

Rusty : $20.00
Launch DECB programs from OS-9! Allows loading of some programs from hard drive!

FARNA Systems AT306 Based Computers

Complete computer systems based on the AT306 board from Kreider Electronics. Systems are completely setup and ready to go. Just add a VGA monitor (or we can supply that too)!

Both systems include:
16 bit PC/AT I/O bus with five slots
MC68306 CPU at 16.67MHz
4 30 pin SIMM sockets
IDE Hard Drive Interface
1.4MB Floppy Drive
Two 16 byte fast serial ports (up to 115K baud)
Bi-directional parallel printer port
Real-time clock
PC/AT keyboard
Desktop Case and Power Supply
(mini-tower case optional, no cost!)
BASIC (resembles Microsoft BASIC)
MGR Graphical Windowing Environment
with full documentation
"Personal" OS-9/68000 Vr 3.0
(Industrial with RBF)
Drivers for Tseng W32i
and Trident 8900 VGA cards
Drivers for Future Domain 1680
and Adaptec AAH15xx SCSI cards

Many other utilities and tools

FARNA-11121 Includes:
2MB RAM
150MB Hard Drive*
Trident 8900 1MB Video Card
$910.75

FARNA-11225 Includes:
2MB RAM
500MB Hard Drive*
Tseng W32i 1MB Video Card
$1114.47

This is the SMALLEST amount of formatted space available.
Prices fluctuate - we get you the largest drive possible for the money allotted!

HACKERS MINI KIT (FARNA-11100): Includes AT306 board, OS-9 and drivers, util software, assembly instructions/tips, T8900 1MB video card. Add your own case, keyboard, drives, and monitor! *ONLY $500!*

Call for a quote on different configurations and components.
Warranty is 90 days for labor & setup, components limited to manufacturers warranty.

Microware Programmers Package -
Licensed copies of Microware C compiler, Assembler, Debugger, and many other tools!
***With system purchase:* $65.00 *Without system:* $85.00**

Color Computer / OS-9 Show & Sale Information

August 2 & 3, 1997
(Sat. 10am-5pm; Sun. 10am-3:30pm)

at the

EMBERS INN

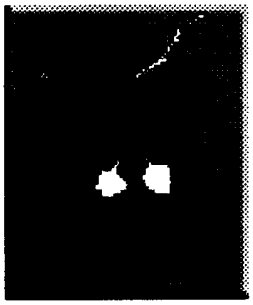
1700 Harrisburg Pike
Carlisle, PA

Overnight room rate: $60
Be sure to ask for the "FEST"rate!

Exit 16 off of the PA Turnpike I-76
Turn to Harrisburg, go 1.3 miles, it's on the right.
OR I-81 Exit 17, turn left, go 1/10th mile, on the right.

Call 1-717-243-1717 OR 1-800-692-7315 OR Fax (717) 243-6648 for reservations! Limited supply of rooms reserved for the show. Rooms will be released on July 1 and will NOT be available at the show rate!
ADMISSION: $5.00 per person per day or $7.00 for both days (paid in advance). Children under 10 accompanied by a responsible adult are free!

There is even more information on Ron Bull's Web site!
www.geocities.com/SiliconValley/Vista/1412/BullsBarn.html



For further information, general or exhibitor, contact:

Ron Bull
115 Ann Street
Duncannon, PA 17020-1204
(717) 834-4314
OR Email me: ronbull@aol.com

Vendors who plan to attend:

Elite Software
PA Online
Carl Boll
StrongWare
SBug
Bargeman Research Labs
MonkWare
CoNect
Alan Dages
SubEtha Software
R.C.Smith
FARNA Systems
Rick Cooper-CFDM
Adventure Survivors
Unlimited Electronics Repair
Paul W. Zibaila III
Black Hawk Enterprises
Glenside CoCo Club

FEATURED GUESTS:

Steve Bjork - Bring your ZAXXON and Gwana Bwana manuals - he said he would autograph them for you! Will be giving a seminar on game programming in general.

Kevin Darling - Will enlighten us with "OS-9 and Multimedia"

Marty Goodman - may do a seminar! One of the most infamous CoCo personalities still with us... or is that *the* most infamous of us all???

For a FREE PA State map and Visitor's Guide call 1-800-VISIT-PA - It's FREE!