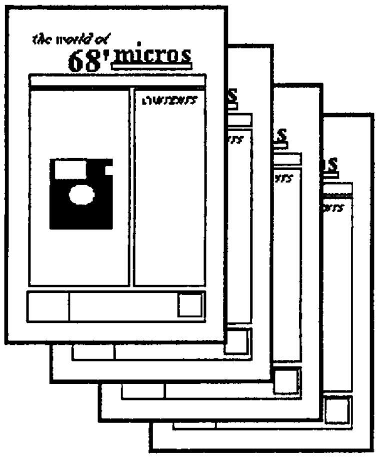
September/October 1995 Vol. 3 Number 2 $4.50 Canada, $4.00 US

the world of 68' micros

Supporting Tandy Color Computer Disk BASIC, CoCo OS-9, and OS-9/68000

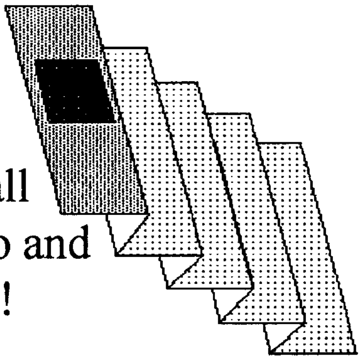


The "official" Volume 1 Index!

For those in attendance, welcome to the 6th Annual Atlanta CoCoFest! For all others, you missed it, and we missed you! Details in the next issue!

New Series!

"Roadmap to the Internet" - learn all about getting onto and using the Internet!



CONTENTS

Uptime Merger Update

See heading in letters!

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

Address Correction Requested

the world of 68' micros

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

Editor: F. G. Swygert

Subscriptions:
$25/year (6 issues) US; $32/year for Canada/Mexico ($13 US, $17 C/M for six months- three issues). Overseas $50/year ($26 for three issues) AIR; $38/year, $18 six months for surface mail.
microdisk: $20 per year, $12 six months, or $6 per issue. Overseas add $10/year, $5/six months, $1/single issue for air mail delivery. Contains programs and source listings from magazine; not stand-alone.

Advertising Rates:
Contact publisher for rates. We have scales to suit every type business. Special rates for entreprenuers and first time advertisers.

All trademarks/names property of their respective owners.

Any and all contributions welcomed. Submission constitutes warranty on part of the author that the work is original and not copywritten by another party. All opinions expressed herein are those of the individual writers, not necessarily the publisher or editor. FARNA Systems reserves the right to edit or reject any submitted material without explanation. Renumeration discussed on an individual basis.

Back issues are $4 per copy. Overseas add $1 each for surface, $2.50 airmail delivery.

Newsstand/bulk orders available. Dealers should contact the publisher for details.

Problems with delivery, change of address, subscriptions, or advertisers should be sent to the publisher with a short description.

The publisher is available for comment via **e-mail** at dsrtfox@Delphi.com. The Delphi CoCo and OS-9 SIGs on Delphi are also frequented (The Delphi SIGs are still sponsored by **Falsoft**).

ENTIRE CONTENTS COPYRIGHT 1995, FARNA Systems
(Authors retain copyright to their articles)

The editor speaks...

F.G. Swygert

Well, here we are again! With a lot of CoCo and OS-9 support going, it sure is good to be here! And for those picking this up at the Atlanta CoCoFest, especially for the first time, it is good to have you in the subscriber base! And I should also say the same for thsoe loyal readers who have stayed with us through the last two years, and all current subscribers. You, the CoCo and OS-9 communities, make keeping up all this hard work worthwhile!

I haven't received a lot of negative comments about going bi-monthly. In fact, I've not any real negative comments at all. The couple I have simply state that they are sorry to see me go bi-monthly, but would rather see that than the magazine fold. Most other remarks have been encouragement to just keep good information coming. Well folks, with that type of encouragement and support, I hope to do just that!

I'm expecting a lot of changes in my life soon. I have put in for recruiting duty with the Air Force. This will mean six weeks in Texas (Lackland AFB), I'm not sure if I'll get in before the end of the year or not, but that is a possibility. So one issue could be late, depending on the timing of going to school. I'm working on getting an issue built ahead of time, but that doesn't help printing, assembly, or mailing time much. Luckily, I do have enough material to build a good issue ahead of time. I'll try my best to keep from being late... I may even take my computer with me!

Of course, sometime after school I'll be moving! The time of the move will depend on when the area I choose opens up for a new recruiter. Recruiting is sort of unique in that respect. It is a four year controlled tour in the area of your choice... provided there is an opening for a recruiter. The problem here is that a particular city only opens up every four years. But I should be able to stay close to the area of my choice, which is the upper part of South Carolina, possibly north-east Georgia. I've been at Warner Robins long enough (seven years!) that I'm due for an overseas tour, and I don't want that at this time. Recruiting will be a challenge and some longer hours (one other reason for going bi-monthly), but at this point in my Air Force career I need the challenge. I will probably stay with it the next seven years and retire, especially if I get to stay roughly in the area of my choice (which recruiting allows better than any other field). And this will allow me to continue supporting you.

One more thing related to all this... make sure I have your correct address! I'd hate for someone to miss an issue due to an incorrect address. And the Post Office doesn't always forward third class mail (and when they do, they charge me extra for it!!).

Let's see, there is one other change I hope to make in the future. That is support for Motorola microcontrollers, specifically the 68HC11 and maybe the 6805 series. These use a 6800 machine code set, so assembly programming will be familiar to those who program in 6809 assembly. It is even possible to use a CoCo as a program development platform in many cases.

The 68HC11 series has enough of a "following" that there is an Internet listserv group. So information shouldn't be a problem. And this should add to the subscriber base, which will mean a longer life for the magazine in general.

Speaking of a longer life, remember that each and every subscription is crucial! I must keep over 200 continental US subscribers in order to qualify for third class mail. Without third class mailing, cost will go up about 70 cents per issue, or rounded up to $5.00 per year. Lower numbers also make it harder to justify the amount of time and work that goes into each issue.

So don't forget to renew your subscriptions on time! And if you know of someone who might be interested, give them a nudge into getting their own subscription. You'll be doing them, yourself, and all the other subscribers a big favor by helping to keep "68' micros" coming out.

Letters to the Editor

UpTime Merger Update!

Find enclosed a check to renew my subscription for another year. I was under the impression that my subscription had already been extended for a few issues due to the UpTime merger.

Larry Osborne
650 Parliament St., Apt. 1614
Toronto, ON M4X 1R3
Canada

Larry, the "merger" with Uptime fell through. Our deal was that UpTime pay me their printing cost for each issue owed a subscriber. I went ahead and sent everyone on the mailing list one issue anyway. The funds never came through - the publisher was counting on some delinquent advertising bills to cover at least part of the cost, and that turned out to be uncollectable. Complaints should be forwarded to UpTime, not me. As it was, it cost me over $100 to send each UpTime customer just one issue.

I enjoyed the articles about hard drives. Now how about an article or two about backing up and organizing your hard drive? I can't imagine the problems when a monster hard drive crashes.

Kent Lorentzen
HC 02 Box 54
Jacobson, MN 55752-9707

Kent, you hit a crucial consideration when dealing with large hard drives: you MUST keep the drive organized nad make backups! For the CoCo, Rick Ulland and others prefer STREAM, as it backs up over several floppies continuously. I think it is on a copy of "microdisk" if you can't get it anywhere else. I don't know what is currently the most used software for OSK.
Organization is something personal. I keep things on my hard drive organized by subject, each subject with its own directory, some subdirectories after that. An example: all games are in a "games" directory, then each in its own subdirectory. This works well for me. Just think of your hard drive as a filing cabinet and the directories as folders and sub-folders. In fact, that is exactly what Apple (Macintosh) calls them... and why! It is a seemingly natural way to organize everything.
The only way you ever want to have to recover from a crash is to restore a backup! Recovery of data is tedious or impossible otherwise. I make backups of my data by keeping copies on floppies as well as the hard drive. I clean the files no longer needed off the hard drive regularly, I can always reload from floppy later. I only backup data files regularly. If I have a program with a difficult setup, I make a copy of the initilization or data file required and put it with the original program disk, or make notes on the options used when installing. This saves backup disks, and I can always reload the program later.

NOTE: The following letter was long, so is highly edited!

I have been a subscriber since the first issue. I renewed up through February of 96, but you crossed out that date and wrote "last issue" on my mailing label. I'm renewing again now becasue I don't want to miss an issue, but did I miss something? On a constructive note, here is a list of ideas I think will better the magazine:
Bring back the Telecommunications column: Some of us live in rural areas. It would be nice to know what programs are out there before making long distance calls. Briefs or transcripts of some of the Delphi conferences would be nice also.
Bring back product reviews: These would be helpful to new buyers.
Get someone from Microware to write something on what is going on with OS-9 and related items.
More general computing news: I find the general computer news interesting even though it isn't directly related to the CoCo or OSK machines.

David Hazelton
10 Country Club Drive, Apt. 8
Manchester, NH 03102

David, to answer your first query, I seem to have goofed! Sorry for the scare, I'll correct the database. And thanks for renewing early just to be safe!
I've asked for letters like yours many times, and get few. Thanks for letting me know what you want! I'll take them all into consideration, especially the idea of posting what programs are newly available on certain BBS systems. Maybe I can get a sysop or someone to post that kind of information for me. Anyone running a BBS supporting the CoCo, OS-9, and/or OSK should feel free to send me updates on files being uploaded. I'll definately print the info along with the name and number of the board.
I print product reviews when I receive them. I don't have the time to review a lot myself, so I have to rely on others. Anyone willing to write a review of a new product they have received should contact me. If a vendor sends me a new product, I will get someone to review it for the magazine and give the vendor permission to quote and/or copy the article.

Roadmap to the Internet, Part 1

Patrick D. Crispen

A complete guide to getting on and understanding the Internet!

"What hath God wrought?"
— Samuel F.B. Morse
The first telegraph message
ever sent (1844)

WELCOME TO ROADMAP!! According to a recent poll by Louis Harris and Associates, thirty-four percent of the adults in America have recently seen, heard, or read something about the mysterious "Information Superhighway." Sixty percent even said that they thought that the Information Superhighway is a really neat idea, even though they have absolutely no idea what it is.

That's where this workshop comes in. Over the next few issues I am going to show you around the Internet, give you some basic commands that will help you use the tools of the Internet more effectively, point you in the direction of people who can help you if you ever get lost, and even give you a glimpse of what the coming Information Superhighway will actually look like.

How am I going to do all of this? Well, each one of these daily lessons will give you a glimpse at one small part of the Internet. We'll talk about particular tools and sites, showing you some traps to avoid, and even showing you some basic commands that will help you use the tools to your own advantage. In the end, I hope that you will gain a better understanding of the individual parts and pieces that, when put together, make up the Internet.

While my goals are lofty, I also have to be realistic. There are so many computer systems out there running so many different software packages, each with their own unique commands, that there is absolutely NO way that I will be able to teach you everything you need to know about the Internet in a few months. Instead, I will teach you the basic commands that are common on most systems, and I point you in the direction of someone who can help you with your questions about the system that you are using.

Count on the fact that the one system that I will fail to give commands for will be yours. Remember a little while back when I asked you to find the name and telephone number of someone at your local Internet service provider who can answer your questions? If I leave anything out in a lesson, if you have ANY questions, or if you are frustrated or confused, call this person!!! I'm going to show you the basics, but your contact at your local Internet service provider will be there to give you the specifics for your system and to answer most of the questions that you may have.

DELPHI NOTE (DN): *Notes will be added to this material to show you where functions can be found in DELPHI. These additions will assume you are doing things live on DELPHI and will NOT cover off line readers.*

"Patience is a necessary ingredient of genius"
— Benjamin Disraeli

When you subscribe to an Internet list, you send an e-mail letter to LISTSERV@XXXXX.XXX which says:

SUBSCRIBE LISTNAME YOUR FIRST NAME YOUR LAST NAME

in the body of your letter. Well, the SUBSCRIBE command

SUBSCRIBE listname <full name>

is just one of dozens of LISTSERV commands that you can use by sending an e-mail letter to any LISTSERV address with a command in the body of your letter!

First off, what is a LISTSERV? Well, a LISTSERV is a mailing list program designed to copy and distribute electronic mail to everyone subscribed to a particular mailing list. We will talk much more about LISTSERVs and LISTSERV commands later, but LISTSERVs work on a concept called "mail explosion."

A single piece of e-mail is sent to a central address (the LISTSERV's address), and the LISTSERV then "explodes" the letter by duplicating that single letter and sending one copy of that letter to every single person subscribed to a particular mailing list (1). This "mail explosion" concept is what allows me to communicate with a large number of people with just a single e-mail letter sent to a central address. What we are going to talk about now, however, is the LISTSERV file server.

What is a LISTSERV file server? Well, besides distributing letters, LISTSERVs can also serve as a "library" of files — files that YOU can retrieve using nothing but a simple e-mail letter sent to the LISTSERV's address with a few simple commands in the body of that letter.

When you subscribe to a list, you mail an e-mail letter to:

LISTSERV@UA1VM.UA.EDU

(a list on the University of Alabama's server) with this command in the body of your letter:

SUBSCRIBE list-name <full name>

To get files from the University of Alabama's LISTSERV file server, you would send another letter to LISTSERV@UA1VM.UA.EDU with a NEW command in the body of your letter:

GET filename filetype F=format

Now that may look a little intimidating, but you are about to see that the GET command is as easy to use as the SUBSCRIBE command. Let's break the GET command down into its individual parts:

GET tells the LISTSERV that you want it to send a file to you.

filename filetype tells the LISTSERV the name of the file that want it to get (for example: COPYNOTICE, ROADMAP94-00001, RFC 1462, etc.).

F=format tells the LISTSERV how you want the file sent to you. For what we are doing, lets use F=MAIL (that way the LISTSERV will e-mail the files to you).

Now suppose I tell you that there is a

file on the LISTSERV file server at the University of Alabama called COPY NOTICE. What do you have to do to retrieve this file? Well...

1) Address an e-mail letter to LISTSERV@UA1VM.UA.EDU (remember, you are about to send a command, and all commands must be sent to the LISTSERV address).

2) In the body of your letter type GET COPYNOTICE F=MAIL

DN: See bottom of this post for DELPHI commands to do this.

How about if I told you there was a file on the LISTSERV file server at the University of Alabama called RFC 1462? Well, again you would send an e-mail letter to LISTSERV@UA1VM.UA.EDU, but this time the body of the letter would say GET RFC 1462 F=MAIL

Think you can handle this? I hope so ... because this is your first homework assignment (eeeeek!). There are three files on the LISTSERV file server at the University of Alabama (LISTSERV@ UA1VM.UA.EDU). Those files are:

filename	filetype	description
COPY	NOTICE	The Copyright notice for the entire Roadmap workshop.
NET	INTRO	My own special explanation of what the Internet is and how it works
RFC	1462	The OFFICIAL "What is the Internet" RFC/FYI by Krol and Hoffman (this is kind of advanced stuff).

What I want you to do is use the GET command to get at least one of these files (you can get more than one if you want). What do I want you to do with the file after you get it? READ IT!! That's your homework. Have a GREAT weekend!!

What if the GET doesn't work? First, realize that it may take several hours for the LISTSERV to process your request and send the file back to you (hence the "patience" quote at the opening of this issue's lesson). 25,000 requests, even at one second per request, is going to take a LONG time to process!

If, after an incredible amount of time has passed, you have not heard back from the LISTSERV, double check that you used the correct address: LISTSERV@UA1VM.UA.EDU (that's "you-ay-won-vee-em"). Second, make sure the GET command is in the BODY of your letter. Finally, make sure that you have included all of the parts of the GET command (GET filename filetype f=format).

If, after all of this, the command still does not work, talk with your local Internet service provider (do NOT write to me). Chances are, the problem is that your mail program is putting the wrong return address onto your letters. This is a local problem, and your local Internet service provider should be able to give you some suggestions. (Again, do NOT write to me!). Have fun :)

SOURCES:

(1) LISTSERV User Guide, EARN Association, July 21, 1993

THE VIEWS EXPRESSED IN THIS LETTER DO NOT NECESSARILY REPRESENT THE VIEWS OF THE UNIVERSITY OF ALABAMA OR UNIVERSITY OF ALABAMA - TUSCALOOSA

On DELPHI, if you are not using an off-line reader you can send these requests by typing MAIL and pressing ENTER from most menus.

Once in the MAIL area type SEND and press ENTER. DELPHI will prompt you for TO:. When sending mail to the INTERNET you must enter the address as:

IN%"address@location.loc"

For the homework, enter the address as:

IN%"LISTSERV@UA1VM.UA.EDU"

then press ENTER. DELPHI will ask for a subject, you may leave this blank.

You will then get a screen at which you may type a message. For the homework, type:

GET COPY NOTICE F=MAIL

Then press CTRL-Z (the CRTL and Z keys, simulataneously).

This same sequence can be used for getting the other documents Patrick talked about.

THE COCO IN THE DOS

James Toth

A couple of years ago I made my sole concession to the ever-growing world of IBM clone computers. I bought a tiny palmtop computer, the microscopic equivalent of an XT. The other computers in my house are pre-MS-DOS Tandys of various vintages, three of which are working CoCos.

As I peeked and poked around inside of my little clone palmtop, I often was struck by similarities between the palmtop and my venerable CoCos. At first I thought it was simple coincidence that so much CoCo-like stuff was present in the palmtop, but coincidence kept piling on top of coincidence.

Eventually an undeniable truth dawned on me — (trumpet blast here) — these IBM compatibles are the direct descendants of the CoCo! The spirit of the CoCo lives inside of every MS-DOS based computer!

This may be difficult to believe at first. After all, the sleek 6809 chip of the CoCo comes from a very different ancestry than the cruder CPUs of the IBM clones. That makes the CoCo and the clones fundamentally different at their very cores. How could one be the descendant of the other? Perhaps these claims are just the wild-eyed ruminations of a die-hard CoCo user. Well, at least part of that last sentence is true. Read on.

My suspicions first were aroused when I attempted to do some graphics on my palmtop, using a version of BASIC that is popular in the world of MS-DOS computers. I found that in order to draw a horizontal line I had to type something like:

```
LINE (100,60)-(150,60),3
```

That command feels familiar to anyone who has drawn lines using CoCo BASIC graphics commands. A coincidence maybe.

By changing the above command into the following,

```
LINE (100,130)-(120,150),3,B
```

The line turns into a box. Still familiar. Very interesting. The plot thickens. Now replace the B in the preceding command with a BF and you get a filled box. All of this is pure CoCo.

Perhaps you are saying, so what? So how else COULD it be done? Maybe this is the only logical way to draw lines and boxes. Believe

continued on page 18

Want to move your CoCo keyboard? Get a

Puppo PC/XT Adapter

Allows use of any PC/XT compatible keyboard with your CoCo. Easy access to OS-9, DECB, and other options. Hold space bar on power-up to skip menu. Switchable and most auto-sensing keyboards supported.

Only $72.50 post paid!
$100.00 post paid with keyboard!

FARNA Systems
Box 321
Warner Robins, GA 31099-0321

NOTE: Keyboards will be 101 key. FARNA reserves the right to substitute 84 key models if necessary without notice.

the world of 68' micros Official Index

Index for volume one.

Articles are sorted in volume then page number order. An ASCII text listing is available on this issue's "microdisk". It can be used with any word processor's "search" or "find" function to quickly locate articles.

FORMAT:
Title; Author (Initial. Lastname); Description; Volume Number, Month/Year, Page *(All info not present for all articles)*

TOTAL ISSUES:	8
TOTAL ARTICLES:	120
TOTAL PAGES:	240
AVG. PAGES EACH:	30

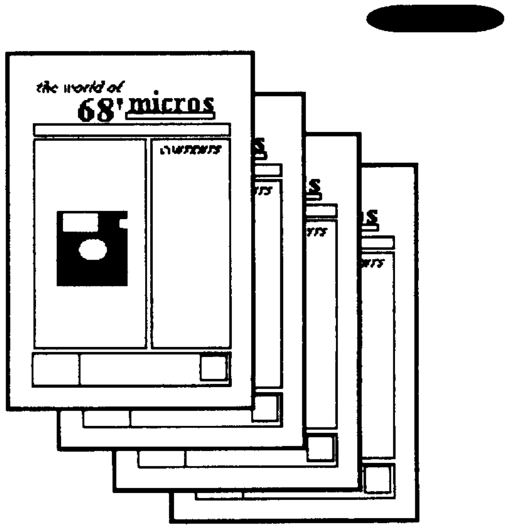
Industrial OS-9 User; C. McKay/D. Reid; OS-9000 use at CANOPUS; V1 N7, 05/94, P14
OS-9/OSK Answers!; J. Hegberg; Adjust terminal settings without tmode; V1 N7, 05/94, P19
OS-9/OSK Answers!; J. Hegberg; MM/1 keyboard driver, read system time, signal notes; V1 N8, 06/94, P18
CoCo Hard Drives; R. Ulland; Burke&Burke CoCoXT system; V1 N8, 06/94, P9
The HD6309E CPU; R. Gault; Native mode and DECB; V1 N8, 06/94, P11
Hardware Hacker; M. Goodman; Use CC3 & MM/1 monitor with video games; V1 N8, 06/94, P13

PROGRAMMING (14 articles)
Programming in C; F. Swygert; Introduction to C; V1 N1, 08/93, P16
Programming in C; P.J. Ponzo; Input and Output; V1 N2, 09/93, P19
Programming in C; P.J. Ponzo; Strings; V1 N3, 11/93, P20
Key Codes; S. Carville; Determining key presses in C; V1 N4, 12/93, P4
Programming in C; P.J. Ponzo; Functions, How to compile; V1 N4, 12/93, P17
Programming in C; P.J. Ponzo; FOR WHILE loops; V1 N5, 02/94, P19
Basic09 in Easy Steps; C. Dekker; Differences between DECB, PC, & Basic09; V1 N5, 02/94, P21
Beginner's Showcase; H. Harwell; Notes on CoCo OS-9 C; V1 N5, 02/94, P24
Programming in C; P.J. Ponzo; IF FOR WHILE loops; V1 N6, 03/94, P21
Basic09 in Easy Steps; C. Dekker; Get Basic09 up and running; V1 N6, 03/94, P23
Programming in C; P.J. Ponzo; Pointers & arrays; V1 N7, 05/94, P22
Basic09 in Easy Steps; C. Dekker; Data structures; V1 N7, 05/94, P24
Programming in C; P.J. Ponzo; More functions, main( ); V1 N8, 06/94, P20
Basic09 in Easy Steps; C. Dekker; Setting up graphics windows; V1 N8, 06/94, P22

HARDWARE (23 articles)
Hardware Hacker; M. Goodman; CoCo floppy drives; V1 N1, 08/93, P10
Repackaging the CoCo; F. Swygert; General considerations, keyboard cable; V1 N1, 08/93, P18
Bob Puppo's Amazing Keyboard Adapter; F. Swygert; Schematics; V1 N2, 09/93, P5
The MPC601; F. Swygert; Motorola Power PC details; V1 N2, 09/93, P7
Repackaging the CoCo; F. Swygert; Portable CoCo3, use external power supply; V1 N2, 09/93, P21
Hardware Hacker; M. Goodman; 512K w/o board; V1 N3, 11/93, P10
PT68K and CDX68X20 Cards; E. Gresick; Which PC cards work; V1 N3, 11/93, P13
Repackaging the CoCo; F. Swygert; CoCo in Tandy 2000 case; V1 N3, 11/93, P22
Hardware Hacker; M. Goodman; CoCo repairs; V1 N4, 12/93, P9
Repackaging the CoCo; F. Swygert; CoCo in PC case; V1 N4, 12/93, P19
Hardware Hacker; M. Goodman; Interfacing CoCo to outside world; V1 N5, 02/94, P9
The HD6309E CPU; R. Gault; HD6309E info, detect presence of; V1 N6, 03/94, P6
Hardware Hacker; M. Goodman; Using CoCo as a controller; V1 N6, 03/94, P10
Tandy Hi-Res Interface Mod; N. Steward; Add by-pass switch to Tandy hi-res pack; V1 N6, 03/94, P22
The Hard Drive; F. Swygert; Hard drive use, types; V1 N7, 05/94, P4
CoCo Hard Drives; F. Swygert; Ken-Ton SCSI Interface; V1 N7, 05/94, P6
OS-9/68000 Hard Drives; F. Swygert; PT68K & KiX hard drives; V1 N7, 05/94, P8
Hard Drive Sources; F. Swygert; Used/refurb drive sources; V1 N7, 05/94, P8
The HD6309E CPU; R. Gault; Use 6309 code to speed DECB; V1 N7, 05/94, P9
Hardware Hacker; M. Goodman; CoCo3 addressing, MacIntosh null modem cable, use DMP 13x with PC; V1 N7, 05/94, P11

REVIEWS (15 articles)
68340 Accelerator; MM/1 upgrade; V1 N4, 12/93, P22
68340 Review Update; V1 N5, 02/94, P25
Canon BJ200e Inkjet Printer; V1 N8, 06/94, P23
Color Schematic Designer; CC3 schematic program; V1 N1, 08/93, P25
Hewlett Packard Deskjet 500 Inkjet Printer; V1 N8, 06/94, P24
Historically Brewed; Historic computer newsletter; V1 N5, 02/94, P25
J&M Microtek 6809 Single Board Computer; Basic 6809 controller board; V1 N7, 05/94, P13
OS-9 Disk Mailing List (DML9); Level II mailing list program; V1 N2, 09/93, P26
QuickLetter; Basic word processing for Level II; V1 N6, 03/94, P26
RASCAN Video Digitizer; CC3 Hardware; V1 N1, 08/93, P25
Rusty; Run DECB programs from any OS-9 drive (hard drive); V1 N7, 05/94, P26
Smash!, Breakout type game for Level II; V1 N5, 02/94, P25
Star NX2480 Rainbow Printer; V1 N8, 06/94, P21
SuperComm; Level II Telecom program; V1 N2, 09/93, P26
Tandy's Little Wonder; CoCo history/technical book; V1 N2, 09/93, P23

TELECOMMUNICATIONS (8 articles)
Telecom; A. Huffman/D. Graham; GEnie & Delphi news; V1 N1, 08/93, P22
PT68K Internal Modems; E. Gresick; Internal modem installation; V1 N2, 09/93, P5
Telecommunicating; F. Swygert; Explaining telecommunications; V1 N2, 09/93, P4
Bitbang; R. Kottke; Use CoCo serial port for OS-9; V1 N2, 09/93, P8
Hardware Hacker; M. Goodman; Error Correcting Modems; V1 N2, 09/93, P13
Connecting a Modem to the CoCo; F. Swygert; Make CoCo cable; V1 N2, 09/93, P14
Telecom; A. Huffman/D. Graham; GEnie & Delphi news; V1 N2, 09/93, P24
Telecom; A. Huffman/D. Graham; GEnie & Delphi news, access Internet from GEnie; V1 N3, 11/93, P18



Hi-lights from the Past, Part 2

Robert Gault

Single Stepping in Basic

rgart2.txt

One of the more interesting simple programs I have seen in Coco magazines, gave the user the ability to slow down Basic. Mostly we want our programs to run as fast as possible. You can see this in the trend towards faster and faster clock rates in the Intel world. But if you are studying Basic, or are learning to write in Basic, things can happen so fast that you miss seeing what occurs. Following a program step by step can help you find errors.

The programs below will allow you to control the speed of your Basic program with a joystick. You can slow it down to a dead stop if desired. Each routine is presented as a Basic driver for those that can't write assembly code and an m/l routine.

Two techniques are used. One intercepts the Basic command and keyboard scanning loops and reads the joystick after each Basic instruction. This can slow down the interval between each Basic instruction but complex instructions still are fast.

The second technique scans the joystick at each vertical interrupt. The VIRQ occurs about once every 3000 machine instructions, so complex tasks like drawing a line get interrupted after every few pixels.

Note how the two drivers use different methods for locating the ml code. The first checks for free high memory, the second just POKEs at $7000.

```
10 REM "SLOW" ALLOWS THE RIGHT
20 REM JOYSTICK TO CONTROL THE
30 REM LETTER BY LETTER SPEED
40 REM OF "LIST" AND SINGLE
50 REM STEPPING IN BASIC.
60 REM (C) ROBERT GAULT APRIL
1995
70 REM START=HIGHEST AVAILABLE
MEMORY
80 GOSUB250:START=START-&H51
90 CLEAR200,START
100 GOSUB250:REM WE NEED TO RE-
CALCULATE THIS BECAUSE 'CLEAR'
WIPED VARIABLES
110 LI=160
120 FOR M=START TO START+&H51
STEP10:SUM=0
130 FOR I=0TO9:READ
A$:VA=VAL("&H"+A$):SUM=SUM+VA:POKE
M+I,VA:NEXT:
    READ CHK:IFSUM<>CHK THEN
PRINT"ERROR IN LINE"LI:END
140 LI=LI+10:NEXT
150 EXEC START:END
160 DATA 30, 8D, 0 , 27, BC, 1 , 68, 27, C
, 10, 588
170 DATA BE, 1 , 68, 10, AF, 8D, 0 , 40, BF,
1 , 883
180 DATA 68, 30, 8D, 0 , 18, BC, 1 , 9B, 27,
C , 712
190 DATA 10, BE, 1 , 9B, 10, AF, 8D, 0 ,
2D, BF, 930
200 DATA 1 , 9B, 39, 8D, A , 6E, 9D, 0 , 21,
8D, 805
210 DATA 4 , 6E, 9D, 0 , 1D, D , 6F, 26, EF,
34, 753
220 DATA 56, AD, 9F, A0, A , B6, 1 , 5B,
27, A , 911
230 DATA 81, 3F, 27, F3, 5F, 83, 0 , 1 , 26,
FB, 990
240 DATA 35, D6, 00, 00, 00, 00, 00, 00, 00,
00, 267
250 START=256*PEEK(&H27)+PEEK
(&H28) :RETURN
```

```
00100 * SLOW (C) BY ROBERT GAULT
APRIL 1995
00110
00120 IOPNTR EQU $6F    SYSTEM
I/O POINTER 0=SCREEN
00130 VERTJR EQU $15B   HOLDS
VALUE OF RIGHT VERTICAL JOYSTK
00140 CONOUT EQU $168   JUMP
ADDRESS CALLED AT EACH WRITE
00150 CMDLUP EQU $19B   JUMP
ADDRESS RUN FOR ALL COMMANDS
00160
00170 * INSTALL SLOW DOWN ROU-
TINES IF NOT ALREADY DONE
00180 EXEC  LEAX  RTN1,PCR
00190       CMPX  CONOUT  ADDRESS
OF CONSOLE OUT ROUTINE
00200       BEQ   A@
00210       LDY   CONOUT
00220       STY   IMGCON,PCR  SAVE
ADDRESS FOR FUTURE USE
00230       STX   CONOUT   SET NEW
POINTER
00240 A@    LEAX  RTN2,PCR
00250       CMPX  CMDLUP  ADDRESS
OF COMMAND LOOP
00260       BEQ   B@
00270       LDY   CMDLUP
00280       STY   IMGCMD,PCR
00290       STX   CMDLUP
00300 EXITEQU*
00310 B@    RTS           RETURN TO
CALLING PROGRAM
00320
00330 * SLOW DOWN ROUTINES
00340 RTN1  BSR   TSTIO
00350       JMP   [IMGCON,PCR]
00360 RTN2  BSR   TSTIO
00370       JMP   [IMGCMD,PCR]
00380 TSTIO TST   IOPNTR
00390       BNE   EXIT
00400       PSHS  D,X,U
00410 A@    JSR   [$A00A]  RUN JOY
STICK ROUTINE
00420       LDA   VERTJR   GET RIGHT
VERTICAL READING
00430       BEQ   Q@       QUIT IF NOT
SLOWED
00440       CMPA  #63      MAXIMUM
POSSIBLE VALUE
00450       BEQ   A@       STALL OUT
IN LOOP
00460       CLRB
00470 B@    SUBD  #1       COUNT DOWN
TIME WASTER
00480       BNE   B@
00490 Q@    PULS  D,X,U,PC
00500
00510 * STORAGE FOR ORIGINAL
JUMP POINTERS
00520 IMGCON RMB  2      IMAGE
OF ORIGINAL CONSOLE OUT
00530 IMGCMD RMB  2      IMAGE
OF ORIGINAL COMMAND LOOP
00540
00550       END
00560
00570 THIS PROGRAM WILL WORK
ON ANY COCO RUNNING ROM BA-
SIC.
```

```
00580 IT WILL PERMIT THE USER TO CONTROL THE SPEED OF BASIC
00590 AND ML ROUTINES VIA THE RIGHT VERTICAL JOYSTICK.
00600 UP=NORMAL DOWN=SLOW TO STOP
00610
00620 SHOULD BE LOADED WHERE IT WON'T BE OVER WRITTEN. SEE
00630 EXAMPLE OF BASIC LOADER.
```

Second version using interrupts:

```
10 REM SLOWRTI
20 LI=80
30 FOR M=&H7000 TO &H7042 STEP10:SUM=0
40 FOR I=0TO9:READA$:VA=VAL("&H"+A$):SUM=SUM+ VA:POKE M+I,VA:NEXT:READ CHK: IF SUM < >CHK THEN PRINT"ERROR IN LINE"LI:END
50 LI=LI+10:NEXT
60 EXEC &H7000
70 END
80 DATA 30, 8D, 0 , 19, BC, 1 , D , 27, 13, 1A, 500
90 DATA 50, B6, FF, 2 , 10, BE, 1 , D , 10, AF, 930
100 DATA 8D, 0 , 2C, BF, 1 , D , 1C, AF, 39, 7D, 775
110 DATA FF, 3 , 2A, 2 , 8D, 4 , 6E, 9D, 0 , 1B, 741
120 DATA D , 6F, 26, F0, 34, 56, AD, 9F, A0, A , 1042
130 DATA B6, 1 , 5B, 27, A , 81, 3F, 27, F3, 5F, 892
140 DATA 83, 0 , 1 , 26, FB, 35, D6, 00, 00, 00, 688
```

```
00100 * SLOW (C) BY ROBERT GAULT APRIL 1995
00110
00120 IOPNTR EQU  $6F    SYSTEM I/O POINTER 0=SCREEN
00130 VERTJR EQU  $15B   HOLDS VALUE OF RIGHT VERTICAL JOYSTK
00140 VIRQ   EQU  $10D   VERTICAL IRQ
00150
00160 * INSTALL SLOW DOWN ROUTINES IF NOT ALREADY DONE
00170        ORG  $7000
00180 EXEC   LEAX RTN,PCR
00190        CMPX VIRQ   JUMP ADDRESS FOR VSYNC IRQ
00200        BEQ  A@
00210        ORCC #$50   TURN OFF INTERRUPTS
00220        LDA  $FF02  CLEAR FLAG
00230        LDY  VIRQ   GET CURRENT JUMP ADDRESS
00240        STY  IMVIRQ,PCR SAVE AN IMAGE
00250        STX  VIRQ   SET NEW POINTER
00260        ANDCC #$AF  RESTART INTERRUPTS
00270 EXIT   EQU  *
00280 A@     RTS         RETURN TO CALLING PROGRAM
00290
00300 * SLOW DOWN ROUTINES
00310 RTN    TST  $FF03  IS IT A VERTICAL INTERRUPT?
00320        BPL  A@
00330        BSR  TSTIO  IS I/O IN PROGRESS
00340 A@     JMP  [IMVIRQ,PCR]
00350 TSTIO  TST  IOPNTR
00360        BNE  EXIT
00370        PSHS D,X,U
00380 B@     JSR  [$A00A] RUN JOY STICK ROUTINE
00390        LDA  VERTJR GET RIGHT VERTICAL READING
00400        BEQ  D@     QUIT IF NOT SLOWED
00410        CMPA #63    MAXIMUM POSSIBLE VALUE
00420        BEQ  B@     STALL OUT IN LOOP
00430        CLRB
00440 C@     SUBD #1     COUNT DOWN TIME WASTER
00450        BNE  C@
00460 D@     PULS D,X,U,PC
00470
00480 * STORAGE FOR ORIGINAL JUMP POINTERS
00490 IMVIRQ RMB  2      IMAGE OF ORIGINAL JUMP ADDRESS
00500
00510        END  EXEC
```

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
<ab282@detroit.freenet.org>

NOW AVAILABLE!

The long awaited update of Paul Ward's "Start OS-9"...

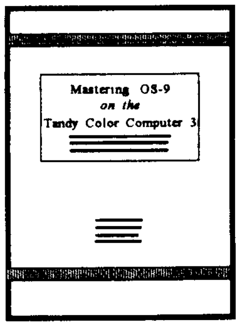
"Mastering OS-9"

Edited and revised by Francis G. Swygert

This new edition contains revised, easier to follow tutuorials, an index, revised information articles, several NEW informative articles (including Rick Ulland on making an OS-9 boot disk), and several extra utilities on disk! 262 softbound pages (5.5"x8.5").

Price for book and disk is only $35 post paid (US)

(Canada add $2 for s&h, overseas add $10 for airmail)


Mastering OS-9
on the
Tandy Color Computer 3

FARNA Systems
Box 321
Warner Robins, GA 31099-0321
912-328-7859

The Hardware Hacker

Mike Jenkins

Using Tandy's "X-Pad" with the CoCo 3

One Sunday back in the summer of '91, my wife talked me into going garage saling. After a few hours I had about all of this that one person could endure. I then spotted a box with "TRS-80" on it and investigated. It was an X-Pad for the CoCo! Every cloud does have a silver lining.

Unfortunately, there was no software or manuals with it. So I went down to the Shack and ordered the service manual. Four weeks later it arrived. I opened it right up to the part that has the good old standby PEEKs and POKEs -- I didn't pay any attention to anything else. I got right down to writing a Basic09 program just to see the X-pad do something. Here is my short program:

```
PROCEDURE x_pad_test
DIM a:INTEGER
DIM stat, ytem, ycor:BYTE
DIM xtemp, xcor:BYTE
RUN gfx2("clear")
1 FOR a=1 TO 5000
NEXT a
xcor=PEEK(65376.)
ycor=PEEK(65377.)
stat=PEEK(65378.)
PRINT "Xcor = "; xcor; "Ycor=  ";
ycor; "Stat= "; stat
GOTO 1
```

And I got the coordinate numbers of the X-pad pen. Of course none of the numbers made any sense, and it was not working the way I thought it should. So back to the book.

After reading about how the X-pad operates, I foound out that it uses the E clock from the CoCo not only for logic timing, but to generate the 55 KHz signal to the pen. It generates the required signal by dividing the E clock signal down to 55 KHz. With that in mind, I slowed my CoCo 3 down and low and behold, numbers that can be used! But the CoCo shouldn't have to be slowed down every time you want to use the X-pad...

Looking at the schematic, the clock signal goes straight to a 4013 dual d flip-flop setup as a divide by two. So all that is needed is another one of these to divide the CoCo 3 2MHz signal by two again. This worked just fine!

The Hardware Fun

What I did was piggy-back a 14 pin socket over IC8 (a 4013). You will also need another 4013 chip. I chose piggy-backing for simplicity. There are only two 4013 chips in the X-pad cartridge. Pull the cover off the cartridge. Hold the exposed board with the edge connector to your left and the components up. Just to the upper right of the center post is IC8 with it's pin one closest to the post and IC9 (a 4040) just under it.

Here is how to prepare the socket ofr mounting:

Take pin 1 and bend it straight out.
Bend pin 2 up.
Bend pin 3 straight out
Leave pin 4 down.
Bend pin 5 up.
Leave pin 6 and 7 down.
Cut pins 8, 9 and 11 short, leaving just enough to solder to.
Leave pin 10 down.
Cut pins 12 and 13 off (not used).
Leave pin 14 down.

You could, of course, do this with the extra 4013, but it is easier to work with the socket.

Now for the soldering. Solder a wire from pin 2 to pin 5 of the socket. Then solder a wire across pins 8, 9, 10, and 11, tying them all together. Now solder the socket to the existing 4013 (IC8). The first joint is tricky. Pin 3 of the socket goes to pin 2 of the chip. Now solder socket pins 4, 6, 7, 10, and 14 to the same pins on the 4013.

Now put the 4013 in the socket. If all went well, your X-pad should now work easily with a CoCo 3.

Some Software

Now for some software that will do something with those numbers. Not being a very good programmer, I have only made a rather simple little program to copy the pen movements to a graphic screen.

```
PROCEDURE xpadraw
 DIM stat, ytemp, ycor:BYTE
 DIM xtemp, xcor:INTEGER
 RUN gfx2 ("clear")
 xcor=0 \ ycor=0
10 stat=PEEK(65378.) —(If this location
     is anything but 3 the pen is not
      touching the pad)
IF stat=3 THEN
 xcor=PEEK(65376.) — (x range of pad)
 ycor=PEEK(65377.) --- (y range of pad)
 xcor=xcor*2.5 --- (matches pad grid to
                  screen resolution)
 REPEAT
  xtemp=xcor
   ytemp=ycor
  xcor=PEEK(65376.)
  ycor=PEEK(65377.)
  stat=PEEK(65378.)
  xcor=xcor*2.5
  RUN gfx2 ("line",xtemp,ytemp,xcor,
   ycor)
 UNTIL stat <> 3
 GOTO 10
ELSE
 GOTO 10
ENDIF
```

Sorry for the line number, but I did say simple! If I remember right, this is set up to use a type 7 graphic screen.

I have started a program that will have a lot more to it, like a save screen to VEF format. It will also be able to pick screen type and resolution. But it is far from being done. As soon as I get further along I will send it in for publication also.

I would really like to see someone write a device driver and descriptor for the X-Pad. Maybe use it in palce of a mouse as well as for graphic input. I would be willing to pay for something done right!

Mr. Jenkins may be reached for comment at:

Mike Jenkins
12A South McKinley
Kennewick, WA 99336

The Industrial OS-9 User

R. Boys

Information on the 68000 series of microprocessors

Editor: All of the information below is from the 68xx0 FAQ. This series will be more specific than the previous article.

68K Features Compared

Feature	68000	'EC000	68010	68020	68030	68040	68060
Data bus	16	8/16	16	8/16/32	8/16/32	32	32
Addr bus	23	23	23	32	32	32	32
Virtual memory	-	-	Yes	Yes	Yes	Yes	Yes
Instruct Cache	-	-	3	256	256	4096	8192
Data Cache	-	-	-	-	256	4096	8192
Memory manager	<+(68451 or68851)+>			68851	Yes	Yes	Yes
FPU interface	-	-	-	68881 or 68882	-	-	
built-in FPU	-	-	-	-	-	Yes	Yes
Burst Memory	-	-	-	-	Yes	Yes	Yes
Bus Cycle type	<++++++++asynchronous+++++++>					both	synchronous
Data Bus Sizing -	-	-	Yes	Yes	use 68150		
Power (watts)	1.2	.13-.26	.13	1.75	2.6	4 - 6	3.9-4.9
at frequency of	8.0	8-16	8	?	?	25-40	50-66

NOTES:

a) 68010, 68008, 68451 are apparently no longer available from Motorola (check second sources, old stock).

b) FPU (floating point arithmetic unit) has eight 80 bit registers.

c) MC68008 is a MC68000 with an 8 bit external data path and A0 pin.

d) MC68882 is an enhanced version of the MC68881. Check the appropriate data sheets for more information and interchangeability.

e) The MC68000/10 external address bus consists of pins A1 to A23. A0 is an internal signal. Using this scheme, the processor accesses memory in steps of 16 bits for a maximum total of 8 mwords or 16 mbytes of memory. The external outputs UPPER* and LOWER* data strobes can be used to effect byte transfers. See the 68000 data sheet for more detail. Other 68k series members have a A0 pin. All internal address registers are 32 bit.

f) MC68060 has a 256 byte entry branch cache. It is also a 3.3 volt part.

g) MC68040V and MC68EC040V are 3.3 volt parts, the rest are 5 volts.

h) MC68HC000 is low power version of the 68000 using HCMOS technology.

i) MC68HC001 is a HC68000 with either a (at reset) 8 or 16 bit data bus.

j) The MC68040V, 68LC040, 68EC040 or 68EC040V do not have a FPU.

k) The MC68HC000 is a CMOS (low power) version of the MC68000.

l) The MC68EC020 is a low cost '020. It has a 24 bit address bus.

m) HCMOS= CMOS combined with HMOS (high density NMOS)

EC and LC (68XC0x0) Family

This family is essentially a subset of the MC680x0 product line. It is designed for low cost (LC) embedded controller (EC) applications. The major differences between the two are listed below. Check the appropriate data sheet for more detailed information.

a) MC68EC000 is a MC68000 with selectable 8 or 16 bit data bus and A0.

b) MC68EC020 is a MC68020 with a 24 bit address bus rather than 32 bits.

c) MC68EC030 is a MC68030 without a paged memory manager (PMMU).

d) MC68LC040 is a MC68040 without a built-in math coprocessor (FPU).

e) MC68EC040 is a MC68040 without a memory manager or built-in FPU.

f) MC68LC060 is a MC68060 without a built-in math coprocessor (FPU)

g) MC68EC060 is a MC68060 without a memory manager or built-in FPU.

External FPUs (MC68881/2) are not easily attached to EC or LC processors since the co-processor instructions are not present on these CPUs.

In the next issue, we will take a look at the features of the 683xx family (used in the MM/1 computers).|

CoCo Laser Show, Part II

Steve Noskowicz

Laser graphic basics.

Computer graphics, the bare basics.

This is a basic explanation of some common computer graphic concepts which are common to any display. It is not intended as a comprehensive discussion, but to introduce most of the basic concepts in simple terms to allow the programmer to figure things out on their own. Most of the concepts are quite simple, but some level of understanding of high school geometry and trig is required. I've been told you can figure out the rotation equations with just trig (sounds about right), but wasn't motivated to try and got these from a demo Basic program on an H.P computer around 1983. I consider everything else "common sense" to someone with this level of education since I figured it all out by drawing pictures and various thought experiments (as well as trial-and-terror code). Believe me, my first rotations were abysmal. I hope this helps you.

Axes and their representation

For all graphics methods, we work in some coordinate system. Though there are several, rectangular coordinates are the most common for graphics. One minor area of possible confusion is that there are different ways of orienting the axes and which way a positive rotation goes.

I believe two common orientations are:

1. X=Horiz, Y=Vert, Z=Out of the screen (or into...see what I mean)
2. X=Horiz, Z=Vert, Y=Out of/into screen.

In the end, the math is the same, just the names would change to protect the images (bad pun - sorry). Also, I think in system #1 but the rotation equations below are in #2 if I recall correctly. I tend to call a clockwise rotation positive, but in classical math it is CCW.

Two dimensions

When we are in 2 dimensions, everything is on a flat surface and we have the old X and Y. This is the graph where you measure the "X" along the horizontal and "Y" along the vertical. A point is represented by an X value and a Y value, or (X,Y) in shorthand. For the point which is right 5 and up 3 we use (5,3). Every point on an object has some X and Y coordinate.

In one of the simplest forms of graphics called "vector graphics", this is as far as we need to go. This is the common "dot-to-dot" puzzle. Images are drawn by connecting the dots with straight lines (called vectors). We (us vector guys) also refer to the X,Y pairs as vectors. 2D vector images consist simply of a list of coordinate pairs. This is basically what a laser light show image is (there are 3D objects, but more on this later). For vector graphics, we commonly use hardware to draw the lines so that computations are only required on their end points, thus requiring relatively little computer power.

Three dimensions

The first step is understanding objects and their representation inside the computer. Later we'll get into the additional manipulations to get them to look right when put into their proper place and then displayed on our (usually) flat screen.

3D requires the third dimension (redundancy if I ever saw it), or "Z" axis, as it's commonly called, to represent the depth, or distance from the observer. Each point of an object has three coordinate values and the short hand form becomes (X,Y,Z). Using three dimensions, we can represent any object we wish. This is commonly called "object space".

Each object is represented by a list of three coordinates for every point on the object. Notice that each object has its own (0,0,0) point. I usually think of this as the "center" of the object and draw my objects around (0,0,0). It really depends on what you are doing, but to start, objects can be stored with the origin at the center or centroid. This is also the center of rotation.

As we get to more advanced graphics, objects are defined in more advanced formats to allow for surfaces, but the three dimensions are the same and the calculations to do things like motion, rotation and perspective are the same except that as images get closer to the photo realistic, more calculations are required to compute the position of all that image data. Just keep in mind that the fully rendered graphics use the same concepts and basic calculations, and that the more advanced forms of representation, such as the matrix, are tools to reduce the notation and allow the programmer to move to a higher level and away from the detail of the math.

Now, what we have so far is just a way to represents objects, as numbers in a computer. Keep in mind that the goal here is to simulate reality. We want to draw something on the screen which looks like the real thing. More importantly, we want it to behave like the real thing as we move it around. Common CRT or LCD screens however, are two dimensional surfaces. They can not display a truly three dimensional scene, so we must do something to make us think we are seeing in 3D. I've always thought of it as "reality simulation". There are things (you might consider these as obvious) which can help an image simulate reality. One is size; draw an object smaller to make it appear farther away. Another is the fact that a closer object can (partially) cover a distant object.

Two common ways to make objects look "real 3D" are:

1. Perspective on a 2D screen and
2. Binocular methods where each eye sees a different image, as they do in real life.

There are also some tricks which can be used to help intensify the perception of depth- even though they may not be not part of the reality. One is to blur a far object. Another is to dim them. Of course, objects do not change resolution or brightness as they change distance, but these type of effects can "help" the illusion at times.

Object motions; Translation;

Remember, in object space, objects are stored at the origin. To translate an object to its desired location, simply add the coordinates of this new location to all the object's coordinates. This moves the object, if you will, into "world space". World space is a synthetic world you are creating and the center of the object is translated to that point. Now, here is where you may want to put the object's (0,0,0) point in a special place. For a glass I may put the (0,0,0) point at the bottom center so that when it is on a table, the height is that of the table top.

Perspective;

The most bang-for-the-buck. Gurus say "project onto the screen" Rotates are pretty nifty when you see the fruits of your labors, but the simple operation of perspective puts everything in your world onto the screen.

Perspective means we go to all the trouble to set up a 3D space and we h ave to put it back onto a 2D surface. This is where we go from "world space" to "display space." A trivial way of doing this would be to simply discard the Z coordinate and put everything at the X and Y coordinated. This is just projection on a plane. Perspective is the primary, and frequently, the only thing needed to simulate 3D on a 2D screen. Perspective refers to the fact that things which are further away appear smaller. They subtend a smaller angle at the eye which means they project a smaller image on the retina. If we draw a scene on a flat screen and put in the proper amount of perspective, the resulting picture will duplicate what one of our eyes sees (projected on the retina). This provides a strong and sufficient "depth cue" to the brain. It also means that when part of a single object is moved farther away, due to rotation, that part also gets smaller. Therefore, we sort of modify the shape of an object to make it look real (add perspective).

You may recall that the primary tool artists use to draw in perspective is the "vanishing point". This is a way to get everything to look the right size for its distance. We, I assume, have a computer, so we can calculate the proper size.

To understand how, lets set up a side view of the eye, screen and object so we can see the classical Z axis. A diagram like this, along with some simple geometry, is all that is needed to come up with the "divide-by-Z-to-get-perspective" rule. Remember to display this in a mono spaced font (T, B and three vertical lines line up vertically).

Hawksoft

244 S. Randall Road • Suite #172 Elgin, Il. 60123
(708)742-3084 eves & ends • MO, Check, COD; US funds
Shipping included for US, Canada, & Mexico

MM/1 Products (OS9-68000)

COMING SOON!!! CDF - CD-Rom file Manager! Unlock a wealth of files on CD with the MM/1!!

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 reader.

KLOCK $20.00 - Optional CUCKOO on the hour and half hour!! Continuously displays the digital time and date on the /term screen or on all open screens. Requires I/O board, 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 together or split into pieces. Record, edit and save files with ease. Change playback/record speed. Convert Mono to Stereo and vice-versa!! Record and Play requires I/O board, cable, and audio equipment

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

GNOP $5.00 - GNOP is the AWARD-WINNING version of PONG(tm) exclusively for the MM/1 !! You'll go crazy trying to beat the clock and keep that @#$%& little ball in line! Professional Pong-ists 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 Coco 3!! Control up to 256 modules, 99 events!!

HI & LO-RES JOYSTICK ADAPTER $27.00 - Tandy Hi-Res adapter or NO adapter at the flip of a switch!!

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

MYDOS $15.00 - CUSTOMIZABLE! EPROM-ABLE! The commands Tandy left out. Optional 6 ms. disk drive speed. Supports double-sided and forty track drives. Set CMP or RGB palettes on power-up. Power-up in any screen. Speech and Sound Cartridge supported. Point and click mouse directory and MORE For all Coco 3 with Disk Basic 2.1. More options than you can shake a joystick at!!

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 CoCo 3, one disk, and joystick or mouse.

SMALL GRAFX ETC.

"Y" & "TRI" cables. Special 40 pin male/female end connectors, priced EACH CONNECTOR	$6.50
Rainbow 40 wire ribbon cable, per foot	$1.00
Hitachi 63C09E CPU and Socket	$13.00
512K Upgrades, with RAM chips	$72.00
MPI Upgrades	
For all large MPIs (PAL chip)	$10.00
For small #26-3124 MPI (satellite board)	$10.00
Serial to Parallel Convertor with 64K buffer, cables, and external power supply	$50.00
2400 baud Hayes compatible external modems	$40.00

ADD $2.00 S&H TO EACH ORDER

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

TERRY LARAWAY, 41 N.W. DONCEE DRIVE
BREMERTON, WA 98310 206-692-5374

INTRODUCINGTHEMM/1B!

A new machine based on a board produced by Kreider Electronics featuring:

o 16 bit PC AT I/O Bus with 5 slots
o MC68306 CPU at 16.67 MHz
- code compatible with 68000
- 2.4 MIPS
o 0.5MB to 16MB DRAM (4 SIMM sockets)
o IDE Hard Disk Interface (2 drives max)
o 1.44MB Floppy Interface (2 drives max)
o 2 16 byte FIFO serial ports
(up to 115K baud)
o Bi-directional parallel port
o On board RS232 buffers
o Battery Backed Real Time Clock
o AT Keyboard Interface & Standard AT power connector
o Baby AT Size Footprint
o BASIC (resembles Microsoft Basic)
o MGR - graphical windowing environment, full documentation!
o "Personal" OSK V3.0
(Industrial with RBF)
- Display drivers: Tseng 4K, generic inexpensive VGA
- SCSI card support: Future Domain 1680 & Adaptec AAH 15XX
o OSK version 2.4 including network file manager, PCF, SCF, SBF, RBF, Pipeman, RamDisk, MW C Compiler version 3.2 with r68/168, MW Basic, MW Debug, MW Programmers Toolkit (Mail, print spooler, UMacs)
o UUCP package from Bob Billson
o Ghostscript (PostScript interpreter)
o Many other utilities and tools

Pricing as low as $400!
(motherboard, Personal OSK, & MGR, no RAM)

BlackHawk Enterprises, Inc.
P.O. Box 10552
Enid, OK 73706-0552
Phone 405-234-2347
Internet: nimitz@delphi.com

```
            Top    Rear
          /          | object
        /            |
Eye   _ _ _ _ _  | _ _ _ _ _Bottom
 z axis
```

I don't know how to do it here, so draw a line from the eye to the top of the object (along the dots). There are five main points (capital letters) of interest here E, S, B, A and T. There are two triangles we will focus on. E-B-T (the eye-object) and E-S-A (the eye-screen).

The goal of all this: where the heck is "A" for each part of the object. Notice that the "A" point for the top Rear point is lower than for the Top front - that's the farther-is-smaller effect. So, whereas "A"? As soon as you realize this is a simple proportion and similar triangles, the math is simple. It just takes a lotta words.

Now, bla, bla, bla similar triangles, remember? Length SA is a fraction of length BT. It is the same fraction, or ratio, as the ratio of the triangle bases (object over screen). The equation is this: SA = BT * times the ratio of SE to BE. or SA = BT * (SE / BE).

Here's where it gets slick. Put your eye at Z=0 and the screen at Z=1. This makes SE=1 and BE equals the Z coordinate of the point. Done. What this says is this: To get the X or Y coordinate you use on the screen, take the Object's coordinates and divide by its Z coordinate. If you don't like that scale, you should be able to do simple scaling of the Z's before doing perspective.

Perspective approximations

If we take a simplistic look at perspective, we can make the math very simple. I want to stress, however, that this is an approximation which works very well for vector and wire frame. I suspect that it would work for non moving 3D polygon representations also.

Three things make this work. First, remember the farther-is-smaller idea. Second, mathematical divisions (and multiplications are nasty (time consuming). Third, this is an approximation.

Calculate the new Z (positive Z toward eye) depth (object at origin) as usual. Take one fourth of it (half is too much, but an eighth would look ok too) and simply add it to the X and Y coordinates for display. Done. I call this "25% linear perspective".

A couple of comments on this approximation. This is a fixed amount of perspective. Scale an object to make it look far and the perspective is too much for the distance unless you scale-then-do-perspective. This looks like one multiplication, doesn't it? I simplified it further because I use fixed point (integer) math and at the assembler level it is two shifts to the right. It's simple, fast and the distortions are acceptable in my application.

Calculations in general;

Never change the object itself, just the displayed coordinates (I think I call this "display space"). The object stays in memory with all its coordinate values untouched. Mess with them and the round off distortion will add up and bend your beautiful objects. Unless, of course, you're going to change it only once for static viewing and sit back and marvel at your handiwork.

The order of calculations has an effect on the outcome. i.e. scale or rotate first. For the ultimate "I'm-doing-exact-no-compromise-all-exact-real-world-simulation-calculations" I suspect, there's only one order for all the operations. If you just want to make a wire frame house or car look right on a static screen, you can do some simplifications and change the order though.

For my laser graphics, I want control of where an object appears ON the screen, so I translate last. I also wanted enhanced perspective when objects are small, so I rotate first. The following algorithms are designed to allow you to be in control of where ON THE SCREEN the object is, a common desire in laser graphics, instead of controlling where in simulated "real-space" objects are located.

Some random thoughts (as if the rest of this is organized). For perspective, the divide-by-Z calculation is done at a certain time. Objects must be stored at the origin (0,0,0). For "object-only" perspective, the rotation and Z translation must be done before perspective. The X and Y translation must be done after perspective or the perspective also translates it closer to the Z axis. For "true" perspective, it comes after rotation and translation.

Full Equations of rotation.

NOTE: The equations presented here are exactly those I received and use a different coordinate system than I use in my laser graphics. This choice is a matter of convenience or preference. It may seem like an unnecessary complication, but this is what I had to deal with and the substitution of variables required to get to your favorite coordinate representation is trivial. I'm sure this all sounds like the unified field theory, but, (like the unified field theory) once you understand it, it's really quite simple.

X, Y, Z Object space coordinates.

X1, Y1, Z1 Coordinates after being "center-of-rotation" shifted.

RCX, RCY, RCZ Center of rotation (or Rotate Center in object space).

P,B,H The rotation angles (Pitch, Bank, Heading)

X2, Y2, Z2 are the post-rotate coordinates.

ROTATION AXES

P = Pitch (X axis) The line through your shoulders. An airplane nose pitched up and down.

B = Bank (roll or Y axis) The line perpendicular to the screen. An airplane rolls about the engine crankshaft.

H = Heading (Yaw or Z axis)) A vertical line.

These equations rotate about the origin or (X,Y,Z) = (0,0,0). Objects are usually drawn with the origin (0,0,0) roughly at the Object's center of gravity. This way, when a rotation is done without any shift in the center of rotation, the object appears to "rotate about its center." This is subjective and at the discretion of the artist. We are in what is referred to as "object space".

To place the center of rotation elsewhere, subtract the rotate center offset from the X,Y,Z coordinates of the object BEFORE rotating and add them from the new coordinate after rotating.

To translate (move) the object around the screen, add the offsets AFTER rotating. This is what is called the "display space" or "observer space".

Equations of rotation:

Rotate center shift:
X1=X-RCX
Y1=Y-RCY
Z1=Z-RCZ

The rotation of coordinates:

```
X2=[COS(B)*COS(H) -
SIN(H)*SIN(B)*SIN(P)] * X1 +
              [-COS(B)*SIN(H) -
SIN(P)*COS(H)*SIN(B)] * Y1 +
      [COS(P)*SIN(B)] * Z1

Y2=[SIN(H)*COS(P)] * X1 +
      [COS(P)*COS(H)] * Y1 +
      [SIN(P)] * Z1

Z2=[-COS(H)*SIN(B) -SIN(H)* SIN(P)*
COS(B)] * X1 + [SIN(H)*SIN(B) - SIN(P)*
COS(H)*COS(B)] * Y1 + [COS(P)* COS(B)] *
Z1
```

Restore the rotate-center offsets:
X3=X2+RCX
Y3=Y2+RCY
Z3=Z2+RCZ

oooooooo
Move offsets applied here
oooooooo

Perspective

Only the X and Z axis rotations cause parts of an object to change distance from the observer, requiring perspective effects. (remember the Z axis of the above equations is the vertical one, not the one coming out of the screen as I use in my laser graphics - change the variables so it is the Y axis if you like)

For perspective, divide the post-rotate X and Z coordinate by the Y (distance from the viewer) (or a fraction of it depending on your scale). An effective pseudo perspective can be obtained by adding a fraction (1/4 looks very good) of the Y distance to the X and Z coordinates. I refer to this as linear perspective. It does produce a slight sideways shift (wobble) as the angle passes through zero.

To rotate on one axis, set the other two angles to zero. This makes the SIN function equal zero causing these entire terms to drop out and the COS function equal one causing just that trig function to disappear.

No, I don't understand the unified field theory.

Basic09 In Easy Steps

Chris Dekker

Advanced Programming: Memory Management

NOTE: the text in this article is a continuation of the last article. You may want to take another look at that, so you won't lose the connection.

For most programs I, and I suppose most programmers, try to use the technique that best optimizes performance and ease of programming. As we saw in the last issue's examples, this optimum is different for just about any kind of program. To keep this article to a reasonable length, I won't discuss all options here.

Some of the examples in the last article are so obvious that anyone can do it. I won't discuss the "virtual buffer" in-depth either. It's concept is fairly simple: the program checks the buffer first and then goes to the disk file if necessary. The gritty parts of the code are in maintaining the correct values for various pointers. But that code is specific to and different for every application you write.

One last comment on it is that this approach seems to work best for programs like "fast check", where most of the action concentrates on the last 50 - 100 records (assuming you keep your books regularly). This approach is rather useless for programs like "mailer", which sorts the names in it's data files alphabetically. Since it is reasonable to assume there are as many names starting with A as there are with Z, a "virtual buffer" would be a lot of programming for little improvement in performance.

For a program like "mailer" you still have some choices: you can have the program "run from disk" which is slow going. Or you can run the program with a small buffer and use lots of small datafiles, like one for family, one for friends, one for businesses, etc. This approach can be inconvenient if you forget who is in which file.

A third approach would be to use the computer's memory outside your addressing space as a buffer and hope you have enough K's installed. Since this technique involves block switching that is what I will concentrate on here and now.

First we will take the easy way: through the use of graphics buffers. I know it sounds weird: using a GRAPHICS buffer to hold simple data (especially since these are the same buffers as the ones used by the GET/PUT graphics functions) but there are too many advantages to overlook or dismiss this option.

For starters we can set up the buffers with gfx2's "defbuff" function. Secondly we can allocate them dynamically (as needed). Thirdly there is only one system call we have to deal with, which in turn leads to few errors that have to be trapped. Best of all we can write all these functions into a single generic module that can be run by any program.

There are two considerations for your own program: first of all it needs a way to transfer bytes back and forth between the graphics buffer and it's own data structures. You can PEEK and POKE this, but that is rather slow. A better idea would be to use the MOVE routine (from a previous installment) or similar code.

The other thing your program has to do is destroy the buffers before it exits. If it doesn't you may end up filling the CoCo's memory with used buffers. Granted, chances of that happening are low, but it doesn't take much to avoid it either. Assuming your program knows how many records there are in memory and how many there are per buffer (requirements that are not too outrageous for useful code) this code would get rid of your buffers:

```
DIM  i,group,buffers,total_records:
                                INTEGER
  buffers=total_records/50  \
                  (* 50 records/buffer
    IF MOD(records,50)>0 THEN
    buffers=buffers+1
    ENDIF
    RUN syscall($0C,regs)
   group=regs.a  \
          (* buffers linked to process ID
   FOR i=1 TO buffers
   RUN gfx2("killbuff",group,i)
   NEXT i
```

Note that the IF ... ENDIF block is necessary to deal with the effects of the integer division. If you skip this code, your program will not destroy the buffer with the highest number unless the total number of records happens to be a multiple of 50.

I have named the module that makes the graphics buffers available to your program: blockswitch (which seemed appropriate enough). To interface with this module your program needs two INTEGERs. The code would look like this:

```
   DIM block,start:INTEGER
  block=1 \start=0
   RUN blockswitch(block,start)
IF block=-1 THEN ERROR start \ENDIF
```

The above code causes blockswitch to switch buffer 1 INTO your address space. If start is has a value other than zero, buffer 1 will be switched OUT of your address space. If blockswitch is succesful it will return the buffer's starting address in start, even if it had to create a buffer first. If a buffer has been switched OUT, start will be reset to zero.

So all your program MUST do is initialize "start", when the program itself starts AND load "block" with a valid number before it calls blockswitch. This would be a value in the range of 1 - 255; the number of buffers allowed per group. And now the code that makes it all happen:

```
 PROCEDURE blockswitch
    TYPE  registers=cc,a,b,dp:BYTE;
x,y,u:INTEGER
  DIM regs:registers; ID:INTEGER
  DIM secloop:BOOLEAN
   PARAM block,start:INTEGER
  RUN syscall($0C,regs) \ID=regs.a
  secloop=FALSE \regs.x=256*ID+block
  IF start=0 THEN regs.y=1
 ELSE regs.y=0 \ ENDIF
 10 regs.a=1 \regs.b=$84
  RUN syscall($8E,regs)
 IF regs.y=0 THEN start=0 \ END \ ENDIF
  IF regs.x=256*ID+block THEN
  IF regs.b=219 THEN
  RUN gfx2("defbuff",ID,block,7900)
  GOTO 10 \ ENDIF
  IF regs.b=132 THEN regs.y=0
  IF secloop=TRUE THEN 20
  secloop=TRUE \RUN syscall($8E,regs)
  regs.y=1 \ GOTO 10 \ ENDIF
 20 block=-1 \start=regs.b
 ELSE start=regs.x \ ENDIF
 END
```

Following is a description of how this module works. First we execute system call 12: GetID. In this code we link the buffers (every buffer needs a group number) to the process ID. In a multi-user system you would normally use the user ID (returned by the same call in register y) for this purpose so every user has his or her own group of buffers available. Since CoCo's rarely run in multi-user setups, I used the process ID as group number. This allows you to run multiple copies of the program simultaneously without worrying about messing up data files.

Secondly blockswitch executes the I$Setstt call, function $84. This function is called ss.MpGPB and is described on page 8-

128 of the technical reference. This call is the heart of the code as it switches the buffers in (regs.y=1) and out of (regs.y=0) our address space.

Unfortunately this system call has a bug in it: the carry bit doesn't always get set if an error occurs. The best way to work around this is to check the contents of regs.x. If it's value is not changed an error occurred. This technique works fine when mapping a buffer, but gives problems when unmapping it. That's why the code skips this check when unmapping a buffer (IF regs.y=0...).

The portion of code after this line may be the hardest to understand. If NO error occurs it jumps to the ELSE statement after line 20 and returns the buffer's starting address to the calling program.

If an error occurs it first checks for error 219. This means that the buffer your program needs does not yet exist. In that case "blockswitch" will create it and loop back to line 10 to switch it into your workspace. The reason for defining the buffer as 7900 bytes instead of the 8192 available in an 8K block is twofold.

First OS-9 reserves the first 32 bytes in the block as a buffer header. It contains, among other things, the buffer's number and group ID. For real graphics buffers this header also contains info on the type, size and position of the data in the buffer.

This would leave us with 8160 bytes in the buffer. This will work most of the time but not always. If your 64K address space fills up and OS-9 switches the buffer in as the highest block (block 8 so to speak) of your address space, you won't be able to access the highest 256 bytes of your buffer. The reason for this lies in the CoCo's hardware. In those last 256 bytes (1 page) of the address space it always maps the same 256 physical addresses. These addresses contain the I/O ports and some important vectors (for interrupts, etc.) Without access to these addresses your program couldn't do much.

So, to avoid conflicts of interest and loosing the odd record, we limit our buffers to 7900 bytes in size. If you are really hard pressed: 7904 is the maximum under this setup. Beyond that you proceed at your own risk. The real problem here is that we can not be sure when this situation will occur and when not, so it is better to avoid it altogether.

The second error "blockswitch" checks for is #132. This is a bit of a weird situation because the value in regs.b has not been changed (#132 = $84) but the system call has not returned a starting address either. Fortunately this happens in only one situation so we can treat it as a valid error code. What happened is that the block you want to switch into your address space already is a part of your address space. The system call more or less verifies it's existence, but doesn't return the block's starting address.

Assuming that your program doesn't know the starting address either, the easiest way out of this dilemma is to unmap the block and then remap it. This is the approach the code in "blockswitch" takes. The code works fine but will occasionally get stuck in an endless loop because the margin for error here is quite small.

To avoid that I used the BOOLEAN variable secloop. This variable allows one unmap/remap operation, but triggers an error the second time around. Which wraps up our discussion of blockswitch.

Another way of switching 8K blocks is through the use of system calls $4F (map block) and $50 (clear block). This way of switching blocks has less overhead, but requires your program to keep track of block numbers, a starting block number and the block's starting address. If one of these three misses, you can not get to or get rid of a block.

Also keep in mind that "starting block" here is an actual block number. You can not number your blocks: 1, 2, 3, etc. but have to use the block numbers that the system calls return to you. The following code should give you an idea of how this works:

```
 DIM start,startblock,block_offset:
INTEGER
 REM first we reserve 5 blocks with
F$Allram
  regs.b=5
  RUN syscall($39,regs)
  IF LAND(regs.cc,1)=1 THEN ERROR
regs.b
  ELSE startblock=regs.b \ENDIF
 REM now that we have the space we must
get to it
  regs.x=startblock \regs.b=1
  RUN syscall($4F,regs)  \(* map block
  IF LAND(regs.cc,1)=1 THEN ERROR
regs.b \ENDIF
  start=regs.u
 REM your program can now access data in
the block
 REM in the same way as with graphics
buffers
 REM when we're done we must free the
address space
  regs.u=start \
(* if program altered "regs.u" regs.b=1
\regs.x=startblock
  RUN syscall($50,regs) \
(* clear block
  IF LAND(regs.cc,1)=1 THEN ERROR
regs.b \ENDIF
 REM will return error if your program gets
suicidal
 REM to access the second of our 5 blocks:
  block_offset=1
  regs.x=startblock+block_offset
  regs.b=1
  REM repeat system call $4F, etc.
```

At the end of the program we must free up the reserved memory. If your program doesn't do this, OS-9 won't be able to access those blocks again until you reboot the computer.

```
 regs.x=startblock \regs.b=5
  RUN syscall($51,regs)  \(* deallocate
RAM
 REM this call returns no error
```

Note that system call $4F can map several 8K blocks into your address space at once. In this case regs.b must be loaded with a value larger than 1. If you want your program to determine the number of blocks it can switch at any given time, you can use the following approach.

Use a variable, for instance: blocks, to load register b. You can initially set blocks to 5. This will most likely trigger an error 207 because there isn't enough room in our 64K address space to accomodate those 5 blocks. At this point you decrement blocks (to four) and try again. Should "blocks" reach zero your program must report an error because then something is really wrong.

If F$Allram ($39) returns an error it usually means that your memory is full or that it has become too fragmented to be useful (Allram only allocates contiguous memory). A solution here could be unlinking some programs or terminating a ramdisk. As you can understand this has to be done manually.

This pretty much wraps up the discussion on memory management. If you thumb your way through the technical reference of your manual, you will see that OS-9 has many more system calls dealing with memory management. This should not be surprising since it is one of OS-9's main tasks. However, since most of these calls deal with specific internal tasks that are not useful or not available to Basic09 programs, we won't discuss them here.

I know that the last few articles have dealt with difficult subjects. I hope they weren't too difficult for you and will help you get a little more out of Basic09 and your CoCo.

Chris can be reached in care of this magazine or directly at:
Chris Dekker
RR #4
Centreville, NB E05 1H0
CANADA

From Color to Black and White

Craig Wheeler

Convert color programs to run on a monochrome monitor.

What would it be like if all the programs in your library wouldn't display output to your screen? I imagine you'd hardly think your collection was worth very much. This is the situation I find myself in sometimes, because I use a monochrome monitor. I'm back in college, and I don't have the spare cash for a color monitor. In fact, if the Manistee CoCo club hadn't done some bargain hunting for me, I would still be using a CoCo 2 without a disk, (thanks, Art). Luckily, my dad found a monitor at a school auction, so I can read an 80 column screen.

Now, don't think that you won't ever have to deal with monochromes' problems just because you have a Mega-Sync SuperColor15 perched on your power strip. What will happen when you try to help out a fledgling CoCoist who happens to be in my situation? What if your club gets a new member with this problem, (you DO try to help other CoCoists, don't you?)? Just keep this advice wedged in the old brain bone in case you need it.

The thing that motivated me to learn how to modify the programs for use on black and white, (B/W), was Rick Coopers CoCo Registry, and the ads that came with it. I first figured out, mostly by trial and error, how to modify the "CR" basic program that loads the Registry. When it came to the "ADS" program, I got more systematic, and that's when I thought my method might be of general interest.I also found the modifications look pretty good on a TV monitor.

When a program displays on a CoCo, it has sixteen colors it can use, whether it's a basic program or a machine language one. When colors besides white, black, or red are displayed, a monochrome monitor divides them into striped blocks, like the old "artifacted" colors from CoCo 2 programs look on an RGB monitor. This makes combinations like orange on green pretty much unreadable on B/W. The thing you need to do is to change the colors assigned to all the pallettes so they are all black, white, or red (I get a pretty smooth grey out of red, or 32). To do this, you need to modify a basic program, or write a short basic program that modifies the pallettes and loads the M/L program.

The way you modify a palettes contents is with the PALETTE command in basic. The syntax is PALETTE <palette number>,<color value>. Now, thiscolor value is a single byte number, and the bigger number is sort of the brighter color.When you change the value in a palette register, EVERYTHING that was drawn with that palette will be changed to the new color.

What I did was write one line, save it with the ascii option, and used a word processor to merge it into the programs I wanted to modify. By the way, you can append the line as many times as you need it, and just assign it the line number you want. DECB will put the lines in the right order when you load the modified program.

Here's the program line I start with:

```
179 PALETTE 0,00: PALETTE
1,32:PALETTE 2,64: PALETTE
3,00:PALETTE 4,32: PALETTE
5,64:PALETTE 6,00: PALETTE
7,32:PALETTE 8,64: PALETTE
9,00:PALETTE 10,32: PALETTE
11,64:PALETTE 12,00: PALETTE
13,32:PALETTE 14,64: PALETTE 15,00
```

You might wonder why I use two zeroes, but I'll get to that. The assumption with this line is that two consecutive palette numbers willprobably need to contrast. It won't work out that way many times, but it's a start.

What you do with a line like this is to insert it in a program right after the line that prints or displays the stuff you can't read. As an example, in the "ADS" program, I gave this line the numbers 101, 179, 345, and 385, because they would execute right before the "inkey" loops and so wouldn't be modified by the M/L routines that load the pictures.

As I said, this is only a start, and in the "ADS" program it wasn't very close. This is also where I explain that the extra zeroes leave room for you to change the palette values from zero to 32 or 64, (like the old Chicago tune). What I do next is to start changing one palette at a time to some new value, and maybe keep notes of what is affected by each change. If you are lucky, a few quick changes will leave the output readable enough that you can use the program. The "ADS" program, by it's nature, required a more systematic approach, since each ad had different colors, (at least, I think they did, I couldn't see them). By the way, if you have to go to a lot of trouble making a table of colors used by the program, it could really help if you could run the program on a color monitor to see what colors are in use, rather than flying blind like I did.

Anyhow, for the really determined, the slow but sure way is to change almost all the colors to zero, (black). If you decide to try this, then palettes zero and 8 still need to contrast, or you won't be able to read the "OK" prompt when you <BREAK> out to edit your colors. The easiest way is to set all palettes to zero except palette 8, which you set to 64, making it white :

```
179 PALETTE 0,00: PALETTE
1,00:PALETTE 2,00: PALETTE
3,00:PALETTE 4,08: PALETTE
5,00:PALETTE 6,00: PALETTE
7,00:PALETTE 8,64: PALETTE
9,:PALETTE 10,00: PALETTE
11,00:PALETTE 12,00: PALETTE
13,00:PALETTE 14,00:PALETTE 15,00
```

If you run the program with these palettes set, anything you seeas white will be the result of palette 8 being set to white, and you note it down as, say, Reactor Control Panel Background - 8. Switch palette 0 to 64 and 8 to 0 to check it, and when you hit the break key the foreground and background will still contrast. From here on in, a good strategy is to set a palette to 32, and see what becomes gray when you run the program. If there is anything helpful that you would like to see on your next trial, change it to 64 and it will be white. Pretty soon there will be a table of things that you can effect with your palette changes and you can puzzle out how to assign white, black, and gray so that the information you want will contrast with its background. You can also try "commenting out" any ATTR commands in your program, as they affect the colors, too.

Finally, as you'll see below, for large blocks of color, or blocks with pure white text on them, those darned "artifact" colors you tried so hard to get rid of don't necessarily make bad backgrounds, and can help return some of the original "flavor" of the old color program. I've put the single line files on a disk, so you can swipe them without having to type them, and here is the final list of palette assignments to view the "ADS" on B/W. Put this line in as lines 101, 179, 345, and 385.

```
179 PALETTE 0,00:PALETTE 1,00:
PALETTE 2,08:PALETTE 3,08:
PALETTE4,04:PALETTE 5,08:
PALETTE 6,00:PALETTE 7,32:
PALETTE 8,64:PALETTE9,64:
PALETTE 10,64:PALETTE 11,64:
PALETTE 12,00:PALETTE 13,16:
PALETTE14,64:PALETTE 15,64
```

Now, for many basic programs, or for those machine languageprograms that can be loaded by a basic program and then leave the palette assignments alone, this method is adequate. For hard-coded machine language, the only remedy I can think of is to disasseble it, and modify any instruction that writes to the palette register addresses. I think I have those addresses on file somewhere, and I'll be giving this method a try later.

Well, I hope any of you who've tried my previous method are seeing nice shades of grey when you run your favorite program. Changing a Basic program is certainly a lot simpler to understand, and a short Basic loader will even come in handy when the program you want is pure binary.

Previously, we had inserted a short Basic line several times at strategic points in a Basic program, and that changed our palettes to values that worked in monochrome. If you tried this on the "ADS" program from the CoCo Registry, you found that the program kept switching the colors back from what we'd set. How come?

Well, the Basic program wasn't doing it, so it must have been the machine language (M/L) subroutine that was activated with those EXEC&H0E00 calls. Fortunately, I found a program that would tell me the start and end addresses of a .BIN file, so I could use a disassembler to see what the routine did. (Actually, I had to use a cassette and the EDTASM+ cartridge, anybody know a good disassembler for the COCO disk?). If you've never heard of one, a disassembler takes binary data and translates it back to source code, in this case assembler language.

When you use one on "ADLOADER.BIN", you get:

```
0E00 LDX #1000
     LDY #0FFB0
0E07 CMPY #0FFC0
     BEQ #0E13
0E0D LDA ,X+
0E0F STA ,Y+
     BRA 0E07
```

This isn't the whole program, but this is the part that loads the palettes. What goes on when "ADS" is running is that the ads are LOADM'ed to another area of memory, (betcha it starts at &H1000), and ADLOADER moves them onto the screen. At the top of each picture are the new palette assignments, and this short loop puts them in the PALETTE REGISTERS, which are at locations &HFFB0 to &HFFBF. If you can stop this from happening, then you can set the palettes once, and the program won't modify them when it's running.

Now, those of you who can see exactly how to fix this, take pity on those of us who have to try lots of fixes. The one thing I didn't want to do was to write the ADLOADER over and re-assemble it. This is a common loop, the first two lines load the indexes X and Y with the first address of the source and the destination. Then the third and fourth lines check to see if Y points to the last palette in the table and GOTO's out of the loop if it does. The fifth and sixth lines load the A register with what X points to (The data table starting at 1000), increment X (that's what the + does) and store that value in the palette table (pointed to by Y, that's what the comma means) and increment Y.Then the BRA instruction makes the program go back to the comparison on the third line and step through the loop again, this time with X and Y pointing to the next element in their tables.(Whew!)

Just to show that I'm not an expert yet, I'll tell you what I tried that failed. First, I figured I'd change the first line to load X with the value of the top of the palette and the loop would just take the palette values and put them back in. The mistake was that the program continues to use X after this loop, because it will then point to the start of the picture. You can't change Y because it is doing double duty as the loop counter. If you change Y to FFC0, the colors are okay, but the text of the picture is in the wrong place. There's an easy change that can be poked into one address by the Basic program, and here it is:

```
POKE &H0E0F,&HA6
```

The clever ones who knew what to do can see what I did. This poke changes the OPCODE of the fifth line so it now reads:

```
0E0F LDA ,Y+
```

When you do something like this, make sure you use the opcode for the addressing mode of the original line, or the postbyte will not be interpreted right. Now the program makes the required changes to X and Y, but it doesn't store anything anywhere, so no changes are made.

This is what we used to call Hacking a program, and it works well for this problem. Just set the palettes the way you want in the Basic program, and insert the poke right after the machine code is loaded.

I submitted this because, though I'd used assembler some, the idea of changing a store to a load this way was new to me. It could be useful when modifying or debugging (porting?) any code that writes to addresses you don't want written to. I hope you find it helpful.

```
1000PALETTE0,00:PALETTE
1,32:PALETTE2,64:PALETTE
3,00:PALETTE4,32:PALETTE
5,64:PALETTE6,00:PALETTE
7,32:PALETTE8,64:PALETTE
9,00:PALETTE 10,32:PALETTE
11,64:PALETTE 12,00:PALETTE
13,32:PALETTE 14,64:PALETTE 15,00
```

```
1000PALETTE0,00:PALETTE
1,00:PALETTE2,08:PALETTE
3,08:PALETTE4,04:PALETTE
5,08:PALETTE6,00:PALETTE
7,32:PALETTE8,64:PALETTE
9,64:PALETTE 10,64:PALETTE
11,64:PALETTE 12,00:PALETTE
13,16:PALETTE 14,64:PALETTE 15,64
```

```
1 REM ADS
10 WIDTH80
20 POKE &HFF9A,0
25 ATTR0,0
30 LOADM"ADLOADER"
40 LOADM"MAINMENU"
41 PALETTE0,0:PALETTE
1,0:PALETTE2,8:PALETTE
3,8:PALETTE4,4:PALETTE
5,8:PALETTE6,0:PALETTE
7,32:PALETTE8,64:PALETTE
9,64:PALETTE 10,64:PALETTE
11,64:PALETTE 12,0:PALETTE
13,16:PALETTE 14,64:PALETTE 15,64
42 POKE &H0E0F,&HA6
50 EXEC &HE00
60 DIM A$(40)
63 J=0
65 READ K$
70 IF K$="END" THEN 100
75 J=J+1
80 A$(J)=K$
85 GOTO65
100 LOCATE 30,21
120 ATTR3,1:PRINT"ENTER# OF
YOUR CHOICE";:ATTR 1,3
121 I$=INKEY$:IF I$="" THEN 121
122 IF I$=>"1" AND I$<="9" THEN
125
123 GOTO 121
125 ATTR3,1:LOCATE53,21:PRINT
I$;:ATTR0,0
127 J$=INKEY$:IF J$="" THEN 127
129 IF J$=CHR$(13) THEN 140
130 IF J$=CHR$(8) THEN ATTR
1,1:LOCATE53,21:PRINT J$;:ATTR
```

```
0,0:GOTO 121
132 IF J$<"0" OR J$>"9" THEN 127
133 ATTR3,1:LOCATE 54,21:PRINT
J$;:ATTR 0,0
135 I$=I$+J$
140 K=VAL(I$)
142 IF K=J+1 THEN 300
145 IF K=J+2 THEN RGB:CLS:END
150 IF K<1 OR K>J THEN EXEC
&HE00:ATTR3,1:GOTO 100
155 ATTR0,0
156 CLS
160 LOADM A$(K)
170 EXEC &HE00
175 LOADM"MAINMENU"
180 IF INKEY$="" THEN 180
195 EXEC &HE00
200 GOTO 100
300 REM CONTINUOUS
310 LOADM A$(1)
320 EXEC &HE00
330 FOR X=2 TO J-1
340 LOADM A$(X)
350 IF INKEY$="" THEN 350
360 EXEC &HE00
370 LOADM A$(X+1)
380 NEXT X
390 IF INKEY$="" THEN 390
400 GOTO 170
1000 DATA "ADVSURV.CMP"
1010 DATA "BARRY.CMP"
1015 DATA "VANDERPO.CMP"
1020 DATA "CHRISDEK.CMP"
1030 DATA "CFDM.CMP"
1040 DATA "COCOPRO.CMP"
1050 DATA "DORSETT.CMP"
1060 DATA "ELITE.CMP"
1070 DATA "FARNA.CMP"
1080 DATA "FED.CMP"
1085 DATA "CURTISAD.CMP"
1088 DATA "VAVASOUR.CMP"
1090 DATA "JIMBENNE.CMP"
1095 DATA "MI&CC.CMP"
1100 DATA "NORTHERN.CMP"
1110 DATA "RADICAL.CMP"
1115 DATA "REMCOM.CMP"
1120 DATA "RCE0.CMP"
1130 DATA "RICKFLIP.CMP"
1140 DATA "SMALGRAF.CMP"
1145 DATA "SPECTRO.CMP"
1150 DATA "SUNDOG.CMP"
1160 DATA "COCOTRAD.CMP"
1170 DATA "TRIVIAAD.CMP"
1175 DATA "UP2DATE.CMP"
1180 DATA "END"
```

continued from page 5

me, there are other and very different ways.

For example, I happen to know of a version of BASIC, popular in Macintosh-land, that draws a box like this:

```
BOX LINES 0,1,0,1
```

There is no CoCo-ness in that graphics command at all. Other logical methods clearly could be dreamed up, but of all the possibilities, the CoCo way was chosen for use in the land of MS-DOS machines.

But wait, there is more. To draw a circle on my palmtop I type:

```
CIRCLE (75,25),10,3
```

Again, that reeks of CoCo.

Could the circle command be a coincidence too? The result of drawing circles in the only logical way? Well, consider another Macintosh alternative for comparison — this draws a circle in Macintosh-land:

```
BOX CIRCLE 0,1,0,1
```

The "box" sets the boundaries and the circle is drawn within it. No CoCo-ness there at all. Again, the CoCo way was chosen over the other possibilities.

We could continue in this manner with other graphics commands such as PAINT and SCREEN and COLOR, but you get the point. The common BASIC graphics commands of IBM-land appear to be direct descendants of the CoCo's graphics.

Get away from graphics completely and incriminating similarities continue to pop up. With MS-DOS comes a powerful hacking and debugging program called, appropriately enough, DEBUG. It can be used for a variety of tasks, including things that we CoCoists would use a disk zapper program to do. Interesting things are to be found inside of DEBUG.

In IBM-land disk drives are referred to by letter names. Your first floppy drive is drive A, and your second one is drive B, and so on. This is pretty different nomenclature from the CoCo's drive 0 and drive 1.

However, DEBUG is capable of reading and writing individual sectors of disks, and if you want DEBUG to use drive A, guess how you do that. You refer to drive A as drive 0. Hmmm. Similarly, drive B is called drive 1. Hmmmmmmm. A stretch, maybe, but where have I seen that before?

And while we are on the subject of disk drives, consider this. In IBM-land the following command saves a BASIC program to the first floppy drive:

```
SAVE "A:\GAME.BAS"
```

I wonder where they got the idea of using a colon (of all possible characters) to refer to a hardware device such as a drive? In CoCo-land it is done like this:

```
SAVE "GAME.BAS:0"
```

Beyond the similarity of overall form, there's the colon again, used for the same purpose. Another stretch, perhaps, but interesting nonetheless.

And how do you save a program in ASCII format? Just add a comma and an "A" to the end of the above commands. It is the same in both systems, of course.

But the most astonishing similarities of all pop up when you begin to investigate disk structure. As you may already know, disk data is laid out in rings called tracks. Tracks are broken down into smaller sections called sectors. One CoCo track consists of 18 256-byte sectors, or 4608 bytes. A common way of laying out an MS-DOS track is to have 9 512-byte sectors, or 4608 bytes again. Interestingly enough, my CoCo disk zapper program is capable of reading the first 256 bytes from sectors on a single-sided MS-DOS disk — that's how similar the two different disks are.

The CoCo keeps track of what files are stored on a disk by making 32-byte directory entries in some sectors that are reserved for that purpose on track 17. MS-DOS keeps track of what files are stored on a disk by making 32-byte directory entries in some sectors that are reserved for that purpose on the first track. In both cases you have 8-character file names with 3-character extensions. It does not have to be done that way, as other operating systems will attest, but it is.

The 32-byte directory entries keep track of what the files' names are and where their first sector of data is stored on the disk. All but the smallest files take up more than one sector, and to see where the other sectors are located you must consult the disk's file allocation table (FAT), which is stored immediately preceding the directory on the disk. Am I talking about CoCo disks now, or MS-DOS disks? It doesn't matter because the preceding text would read the same for either case.

There are a lot of sectors on a disk, and most files use up more than one of them. Because of this, the status of every

individual sector is not necessarily recorded in the FAT. The FAT keeps track of groups of sectors, called granules. That is, if you are a CoCo user, they are called granules. If you live in MS-DOS territory, then they are called clusters. Though the name is different, the idea is still the same.

We could continue in this fashion for a long time, but the moral of the story is clear already. The deeper you dig into an IBM clone, the more CoCo you find. They say that imitation is the sincerest form of flattery. We CoCoists should be very flattered, then, and also proud of the fact that we use the original.

Editor's Note:

James is indeed correct! The IBM PC/XT did start out with an extended version of CoCo BASIC, or more correctly, Microsoft BASIC. Bill Gates wrote the code for Tandy and the CoCo before that of the IBM. When IBM contracted witl Microsoft for a version of BASIC (MS BASIC was the most popular at the time), Mr. Gates simply enhanced what he already had to fit the needs of the hardware and IBM.

A complete 6809 system design (schematic and parts list) is available on the Dufield Development Systems BBS (613-256-5820). A DOS for the system is also available. Several applications (editors, assemblers, debuggers, Forth, etc.) are there too, complete with source code! Might be some things that could be ported to the CoCo...

FOR SALE:

Complete Interactive Media Systems 68070 MM1 Package:
- Slim case
- I/O board
- 3 Megabytes RAM
- 3 serial ports, 2 parallel ports
- 1.4 M and 360K floppy drives
- Conner 170 Megabyte SCSI hard drive
- Tandy CM-8 color monitor (w/glare screen)
- 101-key XT/AT switchable keyboard
- Genius Mouse 3 (3-button)
- Labtec CS-150 speakers (w/MM1 sound cable)
- Owner's manual

Software:
- Microware C Compiler
- Write-Right word processor
- Colorsystems Game Pack
- Colorsystems X10 Control Program
- UUCPbb ready to run
- and of course, lots of shareware programs

Price: $950 or best offer

Hugo Bueno
Delphi: MRGOOD
Compuserve: 71211,3662

micro notes

From: aland@hoder.physics.carleton.ca (Alan Dekok)
Subject: NitrOS-9 v1.22 ships!
Date: Wed. 30 Aug 1995

Northern Xposure is pleased to announce that **NitrOS-9 v1.22 has been shipped**. We apologize for the long delay, but we wanted to ensure that we shipped quality products.

As everyone knows, each Coco setup has its own personality. The speed of NitrOS-9 pushes the limits of the Coco, and sometimes pushes a particular Coco too far. The in depth beta testing has found many places where minor tweaks in NitrOS-9 allow it to be run on marginal systems, with no perceptible performance impact.

Much of the delay in shipping has been due to this beta testing. It takes a few weeks for each version to be written, shipped, and tested. We thank our customers for their patience.

FARNA Systems

Software, Books, and Hardware for all OS-9/OSK Systems!
ADD $2.50 S&H
($4.00 Canada, $10.00 Overseas)

Box 321
Warner Robins, GA 31099
Phone 912-328-7859
Internet: dsrtfox@ delphi.com

CoCo DECB Software:

CoCo Family Recorder - $17.50
Genealogy program for CoCo 3. Requires 2 drives, 80 col. monitor.
NEW! OS-9 Version - **$32.50**

DigiTech Pro - $12.50
Record any sound for easy play-back in your BASIC or M/L programs. CoCo 3 512K.

ADOS: Support for double sided drives, 40/80 tracks, faster formatting, much more!
Original (CoCo 1/2) - $15.00
ADOS 3 (CoCo 3) - $25.00
Extended ADOS 3 - $30.00 (ADOS 3 req., RAM drives, support for 512K-2MB)
ADOS 3/Ext. Combo - $50.00

CoCo OS-9 Software:

Patch OS-9 - $7.50
Automated program installs most popular/needed patches for OS-9 Level II. 512K and two 40T/DS (or larger) drives required. (*128K /35T users can install manually- state 35T.*)

Pixel Blaster - $12.50
High speed graphics tools for OS-9 Level II. Easily speed up your game programming with this tool kit and subroutines!

OS-9 Point of Sale - $62.50
Maintain inventory, print invoices, customer catalog, etc. Multi-user capable . Supports ASCII terminals. Menu driven. CoCo3.

Books:

NEW!! **Mastering OS-9 - $30.00**
This is the long awaited update of Paul Ward's "Start OS-9". New format is easier to read, has easier to follow tutorials, and updated information files. Comes complete with disk, which has a few added utilities.

Tandy's Little Wonder - $22.50
History, technical info, schematics, peripherals, upgrades, modifications, repairs, much more- all described in detail for all CoCo

CoCo Hardware:

DigiScan Video Digitizer - $150: Capture images from VCR, camcorder, or TV camera. No MPI required- uses joystick ports. CoCo Max3, Max 10, Color Max 3 compatible. Special order- allow 90 days for delivery. Send $75 deposit, remainder due before delivery.

Puppo Keyboard Adapters - $70: Use IBM PC/XT keyboards with your CoCo! Mounts in your CoCo case with no soldering. 101 key keyboards available for $30 with order.

FARNA Systems Publishing Services

Type Setting and Printing: We can prepare professional typeset manuals, books, booklets, catalogs, and sales flyers for you - we can print or you reproduce as needed from a master set! Very reasonable prices - inquire!

Mailing Service: If you send catalogs or letter correspondence to 200 or more persons at once, we can do all work for you for about the same cost of your materials alone! How much is your time worth???

Contact Frank Swygert at FARNA Systems for quotes

for all your CoCo hardware needs, connect with

CoNect

449 South 90th Street
Milwaukee, WI 53214
E-mail: rickuland @delphi.com

The main problem with OS9 under a CoCo is the serial port. With a one character buffer, it's hard to do much before the serial port needs service. Windows and OS2 have the same problem. Or did, until National released the 16550 uart- 16 bytes of internal fifo buffering gives multitasking systems time to do some.

Announcing Fast232

Tandy with Sacia			**Fast232**	
bps	load	thruput	load	thruput
2400	28.3 sec	237 cps	25.3 sec	235 cps
9600	73.6 sec	938 cps	31.4 sec	950 cps
57600	not available		32.6 sec	5373cps

Local machines withfaster modems can now be connected to properly (or improperly at 115K!). More down to earth, Fast232 is a ROMPak sized case, which will accept a daughterboard (once I get the case to close) to give two ports in a very 'concise' package. OS9 drivers by Randy Wilson. Free bonus software! The pd release of 'SuperComm' (Dave Phillipsen and Randy Wilson). All software includes 6809 and 6309 versions.

Fast232	**$79.95**
Second port	**$45.00**

NEW FOR 1995 FROM DISTO!

1. **"Inside 2-Meg":** A techinical booklet that fully describes how the DISTO 2-Meg Upgrade kit works. Includes schematic, PAL listing, theory and chip by chip circuit explanations. $20 + $2.50 S/H.

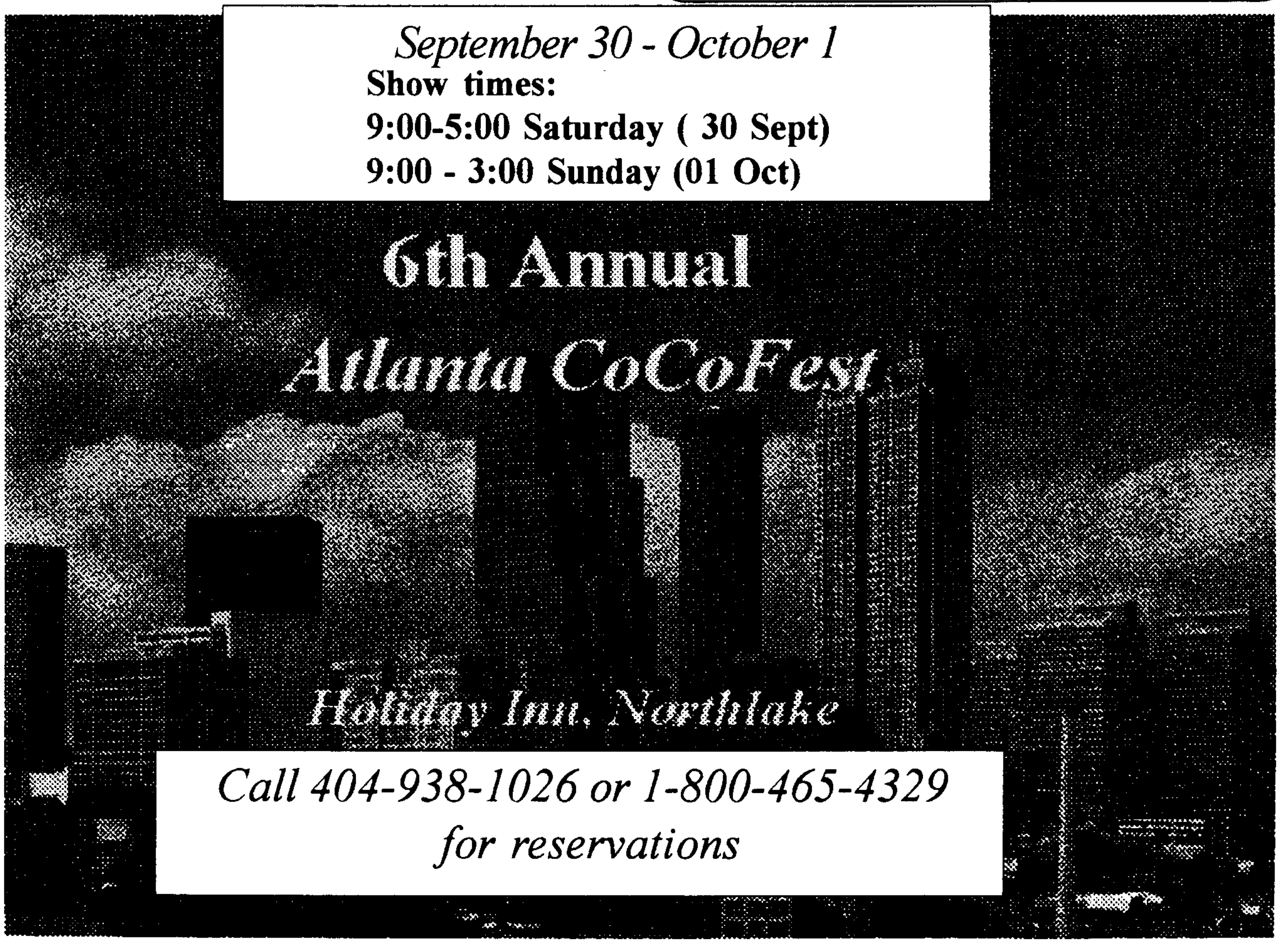
2. **"Blank Board Kit":** Includes blank virgin boards (no components) of the SCII, SCI, 4IN1, MEBII, MPROM and Mini Controller. Collect all the components and make your own! $29.90 + $4.50 S/H.

3. **"Blank Board Plus Kit":** Same as the "Blank Board Kit" except that SCII has parts mounted (but needs repair). Do some trouble shooting and fix, don't have to search for parts! $39.90 + $4.50 S/H.

4. **Call for other DISTO products in stock**
(limited quantities available)

DISTO
1710 DePatie
St. Laurent, QC H4L 4A8
CANADA
Phone 514-747-4851

September 30 - October 1
Show times:
9:00-5:00 Saturday (30 Sept)
9:00 - 3:00 Sunday (01 Oct)



Call 404-938-1026 or 1-800-465-4329
for reservations

Wittman Computer Products

873 Johnson Road
Churchville, NY 14428
Phone 716-494-1506 : Fax 716-293-1207
Internet: ww2150@acsprl.acs.brockport.edu

K-Windows Chess for MM/1
Play chess on your MM/1.........................$24.95
X-10 Master Control for MM/1
Use MM/1 to control you home!.................$29.95
Variations of Solitaire
Pyramid, Klondike, Spider, Poker and Canfield
MM/1...............$29.95 CoCo3..............$19.95
OS-9 Game Pack
Othello, Yahtzee, KnightsBridge, Minefield, and Battleship
MM/1..............$29.95 CoCo3.............$19.95
WPShell
An OS-9 Word Processing Point and Click Interface
CoCo3..$14.95
Using AWK with OS-9
Includes V2.1.14 of GNU AWK for OS-9/68000
MM/1..$14.95

NEW! WCP306 Computer!
MC68306 16.67MHz CPU, code compatible with MC68000. Four SIMM sockets, 512K-16MB memory. IDE hard disk interface, floppy interface, two serial ports, parallel port, real-time clock, all built into motherboard! Designed to use 16 bit AT expansion cards (six slots) and standard AT keyboard, power supply, and case. Comes with "Personal OS-9/68000" and MGR, a graphical user interface with complete documentation. Only $400 for bare board as described above!

Call or write for a free catalog! Demo disks also available.
Owned and operated by William L. Wittman, Jr.

EDTASM6309 Version 2.02 ------------------------------ $35.00
This is a major patch to Tandy's Disk EDTASM to support Hitachi 6309 codes. Supports all CoCo models. CoCo 3 version uses 80 column screen, 2MHz. YOU MUST ALREADY OWN TANDY'S DISK EDTASM TO MAKE USE OF THIS PRODUCT. It WILL NOT work with a disk patched cartridge EDTASM.

CC3FAX ------------------------------ $35.00
Extensive modification to WEFAX (Rainbow, 1985) for 512K CoCo 3. Uses hi-res graphics, holds full 15 min. weather fax image in memory. Large selection of printer drivers. Requires shortwave receiver and cassette cable (described in documentation)

HRSDOS ------------------------------ $25.00
Move programs and data between DECB and OS-9 disks. Supports RGB-DOS for split DECB/OS-9 hard drives. No modifications to system modules (CC3Disk or HDisk) required.

DECB SmartWatch Drivers ------------------------------ $20.00
Access your SmartWatch from DECB! New function added to access date/time from BASIC (DATE$). Only $15.00 with any other purchase!

RGBOOST ------------------------------ $15.00
Make the most of your HD6309 under DECB! Uses new 6309 functions for a small gain in speed. Compatible with all programs tested to date! Only $10.00 with any other purchase!

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
Add $4 shipping & handling per order

CoCoTop version 1.0	$24.95
CoCoTop version 1.1	$19.95
CoCoTop 1.1 + Tools 3	$34.95
OScopy/RScopy	$10.00
TOOLS 3 version 1.1	$29.95
Quickletter version 2.0	$19.95
Accounting level 2	$34.95
Investing level 2	$24.95
Level II graphics 1.2	$34.95

upgrades only $5.00 (return original disk)

Shipping+handling: US/Canada $3.00 all others $5. Prices in US dollars Send cheque or money order NO COD'S. Call or write for Canadian dollar prices. *Mention the name of this magazine in your order and you will receive a free bonus disk!*

C. Dekker •••
RR #4 Centreville, NB
E0J 1H0, CANADA
Phone 506-276-4841

User-friendly Level II Programs!

Mid Iowa and Country CoCo Club

(non-profit)

If you want support, we're here for you! MI&CCC publishes the UPGRADE disk magazine, now in its tenth year, five as a national publication. We've grown to be one of the largest CoCo outreaches in the world! We have subscribers in over 40 states and five provinces of Canada, as well as in Australia and England.

Your MI&CCC membership brings you:
1. A year's subscription to the UPGRADE disk magazine (requires CoCo 3 and one disk drive), 8-10 issues per year. This is a news magazine, not a software disk.
2. Access to our shareware/public domain/orphanware library-- we keep only the best!
3. Optional Christian sub-chapter that gathers Christian oriented software for those interested.
4. ROM burning and other support.

Say you saw this ad in "68 micros" and receive a bonus disk along with your new membership!

Mid Iowa & Country CoCo Club
Terry Simons, Treas./Editor
1328 48th, DesMoines, IA 50311
Please include you phone number and system information

The OS-9 User's Group, Inc.

Working to support OS-9 Users

Membership includes the Users Group newsletter, MOTD, with regular columns from the President, News and Rumors, and "Straight from the Horse's Mouth", about the use of OS-9 in Industrial, Scientific and Educational institutions.

Annual Membership Dues:

United States and Canada	Other Countries
25.00 US	30.00 US

The OS-9 Users Group, Inc.
6158 W. 63d St. Suite 109
Chicago, IL 60638
USA

Northern Xposure

'Quality Products from North of the Border'

OS-9 Level II Color Computer 3 Software

NitrOS-9 v1.20 Call or write for upgrade info or new purchase procedure. Requires Hitachi 6309 CPU	$29.95
Shanghai:OS-9 Introductory price Send manual or RomPak to prove ownership	$25.00
Thexder:OS-9 Send manual or RomPak to prove ownership	$29.95
Smash! Breakout-style arcade game	$29.95
Rusty Launch DECB/ECB programs from OS-9!	$20.00
Matt Thompsons SCSI System v2.2 'It flies!' 256/512 byte sectors, multipak support	$25.00

Disk Basic Software

Color Schematic Designer v3.0 New lower price	$30.00
Oblique Triad Software	*Write for catalogue*

Color Computer 3 Hardware

Hitachi 6309 CPU (normally 'C' model, may be 'B')	$15.00
SIMM Memory Upgrade Runs Cooler! 512k $44.95 0K	$39.95
Sound Digitizing cable	$15.00

OS-9/68000 Software

OSTerm 68K v2.2 External transfer protocol support TTY/ANSI/VT100/K-Windows/Binary Emulation	$50.00
Upgrade from TasCOM (Send TasCOM manual please)	$30.00

7 Greenboro Cres
Ottawa, ON K1T 1W6
CANADA
(613)736-0329

All prices in U.S. funds. Check or MO only. Prices include S&H

Internet mail: cmckay@northx.isis.org

Don't have a subscription to "microdisk"? Don't want to pay the high price for back issues? You can now get the complete Volume 1 of microdisk for $30 (plus $2.50 S&H)! That's an $18 savings over back issues and a $10 savings over the subscription price! Just write and tell us you want the entire volume 1.

All files will be on as few disks as possible, not separate disks for each issue. "microdisk" is not a stand-alone product, but a companion to this magazine. Subscriptions are $20 per year (volume 3 and newer). Single issues are $6 each. Overseas add $10 per year, $1 each for airmail.

ADVERTISER'S INDEX:

Atlanta CoCoFest	*21*
BlackHawk Enterprises	*13*
C. Dekker	*22*
CoNect	*21*
Delmar Company	*BC*
DISTO	*21*
FARNA Systems	*5,9,20*
HawkSoft	*13*
Mid Iowa CoCo Club	*22*
Northern Xposure	*23*
OS-9 User's Group	*23*
Robert Gault	*22*
Small Grafx Etc.	*13*
Wittman Compputer Products	*22*

Don't have a subscription yet?
WHAT ARE YOU WAITING FOR?!
Subscribe today!
(Details inside front cover)

For superior OS-9 performance, the

SYSTEM V

Provides a 68020 running at 25 MHz, up to 128 MBytes of 0 wait-state memory, SCSI and IDE interfaces, 4 serial and 2 parallel ports, 5 16-bit and 2 8-bit ISA slots and much more. The **SYSTEM V** builds on the design concepts proven in the **SYSTEM IV** providing maximum flexibility and inexpensive expandability. Optionally available at 33 MHz.

AN OS-9 FIRST - the MICROPROCESSOR is mounted on a daughter board which plugs onto the motherboard. This will permit inexpensive upgrades in the future when even greater performance is required.

G-WINDOWS benchmark performance index of a **SYSTEM V** running at 25 MHZ with a standard VGA board is 0.15 seconds faster than a 68030 running at 30 MHz with an ACRTC video board (85.90 seconds vs 86.05 seconds).

For less demanding requirements, the

SYSTEM IV

The perfect, low cost, high-quality and high performance OS-9 computer serving customers world-wide. Designed for and accepted by industry. Ideal low-cost work-station, development platform or just plain fun machine. Powerful, flexible and expandable inexpensively. Uses a 68000 microprocessor running at 16 MHz.

G-WINDOWS - a PROVEN WINNER

FOR OS-9

Available for the SYSTEM IV and SYSTEM V computers, the PT68K4 board from Peripheral Technology and the CD68X20 board from Computer Design Services. Resolutions from 640 x 480 x 256 to 1024 x 768 x 256. Support for multiple VGA cards running different processes, different portions of the same process or both.

G-WINDOWS is now AVAILABLE for OS-9000 from *delmar co*

Support for generic VGA boards and SUPER VGA boards including ET4000 (Tseng Labs), OTI067 (Oak), CT452 and CT453 (Chips and Technology), GENOA, WD90C11 (Paradise) and S3 (S3 Inc.) for modes from 640 x 480 to 1280 x 1024 depending on board.

Distributor of MICROWARE SYSTEMS CORPORATION Software

This ad was prepared and printed using QuickEd under OS-9.

delmar co

PO Box 78 - 5238 Summit Bridge Road - Middletown, DE 19709
302-378-2555 FAX 302-378-2556