CoCoNutz!

Christmas 2005

Volume 1 Issue 4 Rev.1

IN THIS ISSUE

A Talk With Roger Taylor of CoCo3.com

by Mary Kramer

This time around I wanted to interview or get to know someone I have talked to quite a bit online.

Roger Taylor, as many of you know, is the creator of CoCo3.com website and many others. He has constantly catered to the public and his fellow CoCoNuts. He loves to come up with new ideas for the website and especially likes to talk to others about his hobby with computers.

Roger is very active in the CoCo community and tries to keep the lines of communication and information open so that the lost CoCo users out there can find one another. If you have a CoCo or 6809-based website he will list it. If you have an article he will host it. In fact, if you have anything CoCo-related he will work something out to post it on his site for all to find and enjoy. Roger has even started a blog section so that others can kind of see what's on his mind from time to time. Well on with the questions:

Where do you live?

I currently live in Kilgore, Texas which is just outside of Longview.

How long have you lived there?

I've lived here for six months now. I've also quickly learned that the weather here is not much different than that in North Louisiana or South Arkansas, the other places I've lived in the past 12 years.

Do you have kids.....pets....?

Yes, I'm a single dad of two healthy kids of ages 5 (boy) and 8 (girl). Both of them are doing well in school, and they like to stay in their dad's hair when they're at home. We don't own any pets at this time, but I'm sure there will be a puppy in the picture one day.

What are your hobbies?

I really just have one hobby; programming. I usually spend some amount of time each day working on my projects; which could be a web site, a web script, or a full-fledged application for various computers and platforms. I guess I was born to create or I wouldn't be having so much fun developing software. Actually, my hobby extends to learning as much as I can about various programming languages and development concepts. The more I learn, the more I want to learn!

What kind of CoCo stuff do you currently have?

I have enough to do some serious CoCoing within a small desk area. My most prized CoCo was nicknamed the 'CoCoTower' because I repacked my 1-meg CoCo 3, 42 meg HD, floppies, and MPI, etc. inside of a PC tower case. Although I've sold most of my CoCo equipment and materials over the years, I still have tons of floppy disks and some hardware gadgets that I couldn't part with.

Which one is your favorite?

Out of the different CoCo models, the CoCo 3 is my favorite. There is still much untapped power in the CoCo 3 that we might not ever see, but because it can run OS-9 Level II and handle so many modern gadgets now (thanks to

Cloud-9 and other dedicated hardware buffs), the power is there for anyone to unleash if !

You're the creator of CoCo3.com; how did that start out and eventually evolve?

My first web site ever was a rather simple page called "The CoCoNut Directory!". I designed it completely using a WebTV set-top internet box. The page was a grid at first, then later became a tower of 200x60 color banners along with contact information for each CoCoNut (CoCo user) who submitted their info to me. After sending out tons of e-mails to people that I knew were active in the CoCo community, I got back a handful of banners and details for me to post. Soon, the tower of banners grew and the web site started to get requests for more features.

CoCo3.com is not really a very large site at this time but it contains what people have asked for at some time or another so there is a chance that you will find something very interesting on your first visit or later. I work on the site all the time looking for ways to improve it or bring in more visitors.

There are also a lot of other nice CoCo web sites out there that I frequent often, so I try to keep their links fresh on the front page of my site.

What other websites or CoCo products do you have a hand in?

Some of my other web sites are client and family sites. Here are a few: waldenhillfarm.com, abminiaturehorses.com, wallersminihorses.com.

I'm not really involved in any outside CoCo projects other than those mentioned on CoCo3.com, but there *could* be a secret project or two in the works. I try not to mention any projects until they are well into development and have a good chance of being released.

Tell me about some of your products you offer on your website.

Well, my flagship product is called Portal-9 which is software used by other developers to create CoCo and 6809 software. It's an integrated development environment (IDE) for Windows and includes a very nice and powerful cross assembler called CCASM. You can tweak Portal-9 to do all sorts of 6809 projects and even see your programs run instantly if you are a M.E.S.S. emulator user (www.mess.org).

Portal-9 is my way of giving back to the CoCo community what it taught me. The software is also my attempt at helping preserve 6809-based computers of all types. By giving 6809 programmers an all-in-one easy way to write, compile, and see results in seconds using a PC, I think development for the older computers will continue for many years to come.

Got any specials coming up for Christmas?

Let's wait and see. Each year I try to come up with something new for the web site, and you never know what I might dish up!

What are some of the future plans for CoCo3.com?

I am working on letting visitors submit their own articles in which I can approve using a control panel and post easily. Submitted articles should blend in with the web site design quite nicely and automatically.

I'm also working on letting visitors submit their info to a new CoCo registry which can be searchable by the public. You'll be able to safely e-mail each person (one at a time) without exposing their e-mail address in any way. People will be able to describe themselves and CoCo interests and also include other optional info.

Ah, a Portal-9 project depot would be nice. So many ideas, so little time. Yes, CoCo3.com will continue to grow and will offer what CoCo users ask for. I always enjoy improving the web site!

What kind of programming do you do?

Most CoCo users know that I can do some serious things with the 6809 and 6309 CPUs, and I've been known to crank out a mean CoCo program in my day. Today I spend more time trying to help preserve the CoCo by using web design and even Pentium machine language programming.

CoCo3.com, Portal-9, and CCASM all three took very serious programming to pull off, and I would have to call its kind "determined". Seriously, I dabble in it all. If it looks like fun and I can produce something quick using a certain language, I will try it.

When it comes time for new languages I've been doing this stuff for so long that I can sit down and "learn" a language in a small amount of time or get by on just enough of what takes to get a job done and then build on it over time to learn more. Today I might hop between such languages as 6809 assembly, Pentium assembly, HLA (Randall Hyde), PHP, MySQL, Perl, HTML, CSS, JavaScript and other means of coding to get the job done. Some other languages just kind of fall into place due to similarities between them and other languages.

What was your first computer?

A 16k CoCo 2! It was right before Christmas and I also grabbed up my first Rainbow magazine that year which was chock full of CoCo programs to type in. This was not really my first computer to play around on but it was my first to have at home to call my own and to have full control of. This period should go down in my history as the beginning of my "night owl" days, where I spent countless hours each night hacking and learning how to write code in various languages for the CoCo. I created more things during the first few years of my first CoCo than I might create in the next 10 years using any other platform.

Why did you choose that one?

If I recall, they ran a $99 special at Radio Shack and I could not resist. Besides, it was the latest model at the time and there were lots of brand new things to go along with the CoCo 2.

What is your fondest memory of the CoCo?

The memory that sticks in my head is when I went through that first big Christmas edition of The Rainbow typing in program after program and watching everything run. I didn't sleep much back then!

Did you ever write for CoCo magazines?

Sadly enough, no. I spent my every waking CoCo minute coming up with something new, but why I couldn't pull something from the surprise hat and send it in to The Rainbow is beyond me. I guess I spent so much time learning that I didn't have time to teach or tell others how it should be done - yet.

Do you ever attend any of the CoCo Fests, past or present?

I was a lot younger back when the CoCo Fests were big so I never could seem to fund such a trip, but I always wanted to. I would read about the fests in the Rainbow magazine and drool over all the CoCo games and hardware they would bring.

I was at a CoCo Fest "in spirit" one year when I sent a bunch of copies of Projector-3 to Carl Boll to put on his booth table to sell. It turned out quite well and I got lots of feedback and requests to actually BE THERE the next year! :)

I also recently hosted a live web cam feed with text chat from a CoCo Fest. This was a last-minute joint effort between me and Dave Kelly that turned out to be quite successful. Quite a few people showed up for chat with video that updated about every 10-15 seconds. Expect things like this to happen more.

What is your favorite CoCo utility?

I used something called V.I.P. back in the day which had a spreadsheet, word processor, database, and several other useful utilities all built into the same system so you could switch between them easily. This package also included a disk "zap" program for manually repairing garbled floppy disks. You had to know the technicalities of a CoCo file allocation table in order to really use the program, but I became a pro at it after a while and fixed many disks that were trashed from time to time from unexplained disk drive errors.

You had a utility that made it easy to view all kinds of graphics for the coco. Tell me about that.

Yes, I originally wrote The Projector which allows the CoCo 3 with 128k to view all sorts of picture formats. Although the color range was low, you could at least load and see some of the modern picture formats at that time, including pictures for the CoCo, Atari, Macintosh, and PC.

The Projector later advanced to Projector-3 which requires 512k and spits out nice dithered pictures of even more formats than The Projector. P-3 can even be extended by letting you write your own format drivers (CODECs) that are automatically recognized when you start the program. My system allows you to load and/or save in whatever format that contains a decoding or encoding routine. So, P-3 was to be the last picture viewer ever needed for the CoCo 3, but because it requires a 6309 CPU, not everybody has gotten to experience the system.

How did you come up with that?

The Projector series viewers derived from my first viewer ever called Super-MAC, a Macintosh picture viewer. This viewer displayed .MAC pictures in super detail by flickering two screens of the image very quickly. One image is the even scan lines and the other image is the odd scan lines. When you scrolled the image, a video effect on some monitors would result in over 500 lines of video being displayed at once. Talk about nice!

Once I realized I had I thing for graphics I started collecting specs for various popular picture formats. Soon a system was born that let me add and remove viewing capabilities to the main program without having to recompile the entire system (EDTASM woes). P-3 later took this to a new level by allowing not only graphics but text files and even music to be decoded, although these features aren't really recognized or used by anyone today.

Did you publish it?

Well, I produced my own copies of The Projector and Projector-3 on floppy disk with printed manuals and shipped it myself to people all over the world. It was a one-man effort.

Can I still get a copy?

You sure can, and like most other programs I've written in the past - it's free. You can download The Projector and Projector-3 from CoCo3.com.

What do you plan to do about the issue of people violating the CoCo Cafe chat?

This is an interesting question because back when I announced that I was writing a live chat page for the web site I asked a bunch of CoCo users their opinion about this. The vote was pretty much to avoid a username/password system until absolutely necessary. I agreed that convenience would bring in more chatters than making everybody go through some sort of hassle. In fact, it's so easy to enter chat that if your browser cookies are enabled, all you have to do after your first visit is just return and click 'ENTER' and there you are talking live to other users. However, this makes it easy for anybody to just come in and post anything. A language filter is in place that catches most bad words, but this doesn't keep non-CoCo related chatters out.

What I will probably do is take another vote to see what most people would want to do about undesired chatters in the CoCo Cafe and then come up with the software solution that is fair to everyone. I have thought about linking the chat system with the CoCo3.com forum accounts and as long as a forum member is logged into the site, the

chat system will use the same username. This way, people are accounted for better than the anonymous system in place now.

Will you bring back to CoCo Store?

Certainly! That effort was put on hold for a bit but the script just needs some tweaking before it is ready to put back online. Actually, it's quite limited so I plan to rewrite the CoCo Store script in PHP/MySQL and maybe call it Vinties again and allow not only CoCo goodies to be listed but stuff for other vintage computers as well. Again, so many ideas, so little time!

What do you like to do most on your CoCos?

Today I like to step back in time by playing around with programs and games that I remember as a kid. Playing some of those old games reminds me of how it used to be and what limitations the developers had to work with. We are spoiled now to having everything at the click of a mouse but there was a day when you actually had to work hard to get a program running. This made us appreciate things better I suppose.

How many personal computers do you own?

Working or non? :) Well, I use two networked PCs every day for most of my work and play. My main tower PC and laptop are connected via Ethernet so I can swap files quick and take my work on the road if I need to, etc. I also have several other PCs in the closet that I'm saving for a rainy week to turn into Linux boxes or something.

What is your favorite brand?

I really don't have a favorite brand, but I do prefer power from the CPU and video card. So my main PC is a Pentium 4 with very nice video and sound capabilities. My main PC can double as a HQ entertainment center, actually. See what I mean by being spoiled?

What is your favorite operating system? Are you a Linux guy or Microsoft guy?

I'm actually a CoCo guy, a Microsoft guy, and a Linux guy all rolled in one.

I use Windows every day so I prefer the convenience to get things done quicker with much less hassle than from any other OS but I also like the control that Linux gives me with far less restrictions than what Windows allows. I truly believe an operating system should be ours to customize to our tastes, especially if we paid lots of money for it.

With the CoCo, my favorite OS would probably have to be NitrOS-9 which is a highly-customized version of OS-9 Level II. Although I've written almost everything for Disk BASIC, I prefer the power of OS-9 when I use the CoCo.

I DARE YE ENTER...
...THE DUNGEONS OF DAGGORATH!!!

An Article by Diego

The sword in your hand feels heavy and cold. Useless. He's closing. You can hear the magic crackling over the pounding of your own heartbeat. As you push your back against the wall in a dark corner, you curse the golden ring in your finger. It's magic for sure, but without the right command word, it's useless.

"Vulcan" The ancient god. The forger, working inside the volcano, making weapons for the Olympians. And then, you know it. The word. Just as the wizard turns the corner, you shout "Incant fire". Now the magic flows from your hand, filling the dungeons with unnatural light. Your are ready for the challenge...

A 3D game in real time for the CoCo 1? Yes, and one of the best ever. In "Dungeons of Daggorath" you are a young warrior who has to defeat a powerful wizard and his evil army.

DOING THINGS

The game takes part in a 3D maze, but, unlike modern games you have to type commands to get things done. "MOVE" to take a step forward, "GET LEFT RING" to pick up a ring with your left hand, and "PULL RIGHT IRON SWORD" to get that sword out of your backpack. If it seems awkward, you can just type "M","G L R" and "P R IR SW", but it's still a real time world.

Even as you type, the creatures are moving around, looking for you. When it's time to fight (ATTACK RIGHT if your weapon is in that hand), you have to type fast, or die trying.

THE WIZARD'S ARMY

To get to the Wizard, you will have to defeat over a hundred creatures, from nasty spiders to armored warriors. Each creature has its own strengths and weakness, and as you will soon learn, a characteristic noise. As you move further down the dungeon, you will find stronger and faster creatures, but be careful... If you decide it's too dangerous, and try to go up again, certain defeat awaits you. When you destroy one of the creatures, the power it holds is transferred to you, making you stronger with each kill. Some of them will also hold items that will help you in the quest.

TOOLS AND STUFF

When you go into the dungeons, you are quite ill-equipped. All you get is just a torch, a leather shield (which is useless due to a bug in the game) and a wooden sword. That's not gonna be enough! You need to get better weapons and a lot of magic items to have any hope of survival.

Maps, potions, magic rings -- the only way to get them is killing creatures, and only the most powerful ones have useful things.

Except from the most basic items, everything has some magic in it. To take advantage of this, you need to "REVEAL" the real name ********

If you are not powerful enough, you still can use the item as if it was a non-magical one.

CAN YOU SEE ME? CAN YOU HEAR ME?

One of the reasons this game was (is) so successful is the way graphics and sounds are used. The game uses the 256x192, 2 colors mode. The creature's graphics are just somewhat simple outlines, without any animation, the graphics are only scaled to indicate distance. There is also a "fading" effect that depends on which torch you are using and how far things out that works very well. One of the reasons for the outlined graphics is that the creatures are actually transparent. This way you can see what's behind them... usually, a lot of other creatures.

And then, there is the sound. Besides the great quality of the noises, the way they are used is impressive. Each creature makes a distinctive noise that starts as a faint whisper and can become a terrifying thunder when something is just around the corner. And instead of background music, there is the sound of your heart beating, which is also your health's indicator. A healthy and rested heart beats slowly, but after a hard fight, or a fast flight, you can hear the heart beating faster and faster.
After playing a few times, you get to know if the creature that you can hear closing in is to strong for you to face. And that's were the game is really different.

JUST A HEARTBEAT AWAY

You can try to run away and loose the creature in the maze, but there is one catch. Almost everything you do, including attacking and walking, makes your heart beat faster. That means that if after a fight, you try to run, you will probably pass out, and while you recover, the creature has a very good chance of catching you and having an easy kill.
Also, the more powerful the weapon you are using, the hardest it is on your heart. You can pass out just by attacking to fast. So you have to be very careful when facing a powerful creature. If you want to flee, the best choice is trying to turn around the corners, and wait for the noises to fade away.

NO COCO? NO EMULATOR? NO PROBLEM!

The game was originally released in a “program pak” dated 1982, and was one of the last games offered in the Radio Shack catalogs. It has been cracked for disk and cassette use, and it's easy to find in DSK files. It doesn't work fine in some emulators, but unlike other CoCo games, that's not needed to play it in a PC. This is because it has been ported to the Wintel platform, with slightly enhanced graphics and sounds (now in useful stereo), but basically, the same exact game. But the best way to play it, is still on a real coco.

There is a “sequel”, actually a totally different game with a similar style, called “Castle of Tharoggad” for the CoCo 3. Stay away from it. It has nothing to do when compared with the original. Maybe somebody will make a really good sequel? I would for sure pay for it.

On a final note, the sound is so well implemented, that even blind people has been able to finish it without any help. How many games can make that claim?

Links

http://members.tripod.com/~Frodpod/index-2.html
http://mspencer.net/daggorath/dodpcp.html
http://nitros9.stg.net/daggorath.html

The Art of BASIC Programming

By Robert Gault

INTRODUCTION

Now there's a phrase to conjure with, "the art of Basic programming". What does it mean if anything? Much better would be to say, the science of Basic programming because to evaluate your programming efforts you must measure them against some objective not subjective scale.

Since we are going to try to evaluate our efforts, what should we measure them against? Here are some possibilities that we can try on for size: style (i.e. matching some instructor's "way to do things"), easy to read source code, maximum speed of operation, easy for customers (and yourself) to use, flexibility of modification, compactness (i.e. more function with less code). You can probably think of other measurements. Now are these objectives or subjective goals?

Style – definitely subjective and almost impossible to quantify. If there is any benefit to having it, it may get you a higher grade in some course. Otherwise this is of no value.

Easy to read – while this is difficult to measure and changes with the person doing the reading, this is an objective goal. You will have no doubts that it is valuable when you revisit your code weeks, months, or years in the future.

Speed of operation – objective. This is very easy to measure with a watch and everyone hates to wait for some code to crank out an answer.

Easy to use – objective. This is also difficult to measure but it is easy to compare A against B and select the better of the two. It will always help to have this evaluated by someone else who does not know how you intended your program to be used. That's what alpha and beta testing is all about!

Flexibility for change – hard to measure but objective. If you have ever attempted to repair or enhance code and found making any changes more difficult than a total rewrite, you will know just how valuable this goal is.

Compactness – objective and easy to measure. This is most important when disk or memory space is at a premium as it is on most Coco systems. As an example, if you are creating a graphics program and it becomes so large that there is no room left in memory for a graphics image, you know you have failed miserably.

EXAMPLES

OK, so we have some goals and you can add your own to the above list; now what?! We can measure the above criterion, but how can we predict that some chunk of code will contribute towards the above goals? How can you tell which of the following snippets of code is "best" based on the above?

Function - wait for a key press

A

…

100 IF INKEY$=""THEN100

…

B

…

100 FOR I=-1 TO I=INKEY$="":NEXT I

…

C

…

100 EXEC &HADFB

…

```
========== SCAN-3.BAS ==========
              by Rodney H.

This two-liner for the COCO 3 lets you scan
the display through the entire 512K address
space.

  "1" selects the text screen
  "2" selects PMODE 0 graphics
  "3" selects PMODE 2 graphics
  "4" selects PMODE 4 graphics
  UP-ARROW scrolls upward
  DN-ARROW scrolls downward

Interesting memory 'pages' to examine include
the top of 32K RAM where you can watch the
6809 stack and BASIC string space, and the
very active default start page with BASIC's
working storage.

1 A$=INKEY$:A=A-(PEEK(&H155)=&HF
7)+(PEEK(&H156)=&HF7)AND&H3FF:PO
KE&HFF9D,A/4:IFM>1THENB=&H7F AND
A:POKE&HBA,B+B:SCREEN1,1ELSEB=1:
FORI=&HFFC6 TO&HFFD2 STEP2:POKEI
+SGN(A ANDB),0:B=B+B:NEXT
2 IFM THENIFA$<"1"ORA$>"4"THEN1E
LSEM=VAL(A$):IFM=1THENSCREEN0:GO
TO1ELSEPMODEM+M-4:GOTO1ELSEWIDTH
32:M=1:A=&H380:GOTO1
```

It will be necessary to learn how Basic works on the Coco to predict which of the above is the "best" code. Basic is an Interpreted language. This means that when a program is run, the interpreter starts at the beginning of the program, and looks for commands or functions that have been predefined in a reserved word list. The program was tokenized as the program was typed into memory but now the tokens must be looked up in a list, their jump addresses found, and the correct code executed. The interpreter continues line by line through the program until finished. If any GOTO or equivalent statements are found, the interpreter must restart at the beginning of the program until the correct line number is found before the command can be executed.

Let's count the tokens in each example to start the process of evaluation. Line **A** has IF, INKEY$, =, THEN, and a line number for a total of 4 tokens. Line **B** has FOR, =, TO, =, INKEY$, = , NEXT for a total of 7 tokens but no line number. Line **C** has EXEC as the only token. Line **C** is certainly the most compact. It will also execute faster as fewer tokens need to be looked up in a jump table.
Both lines **A** and **B** use INKEY$ but line **B** has the function embedded in a For/Next loop, which adds some overhead to the program but prevents the need to search for a line number when looping. Line **B** probably will run faster than line **A** but it really should be tested. The determining factor will be if the line number is near the beginning or end of a large program. Line **C** calls an ml routine in memory that is as small as they come: a@ JSR KEYIN, BEQ a@. Since INKEY$ also uses JSR KEYIN, line **C** is the fastest of the three lines for execution. Looping is done without using Basic.

Which line is the easiest to understand? That will depend on how familiar you are with Coco Basic but lines **B** and **C** are certainly less clear than line **A**. Still line **C** is a winner on two out of three criteria and with a comment added would easily win on all three counts.

SOME THINGS TO REMEMBER

Here are some things to remember about Basic which will help you write the best programs possible. They are not in any particular order and it is hardly a comprehensive list.

1) Document everything you do in a program. If there is any doubt that you might forget why some code is present, put in a comment. If the program is being written as a tutorial, add comments to every line of code.

It is true that comments will slow down the running speed of a program, but not to worry! Keep one copy of the program with a comment for every line, as your source code. Remove all the comments from the copy that you will actually be using. You can make this easier to accomplish by keeping the code and comments on separate lines, say every even numbered line is comment and every odd numbered line is code.

2) Basic will search from the beginning of a program to the end looking for line numbers associated with GOTO, GOSUB, THEN, and ELSE. Therefore the targets of these commands should be as close to the beginning of the program as possible. Put another way, all subroutines should be at the beginning of your program. GOTO should be used as sparingly as possible, as should THEN line# and ELSE line#. This will increase the execution speed of your program. It will also reduce the chances for spaghetti code which jumps all over your program in ways impossible to follow. It will also make it much easier to read and understand your program in the future, when you are trying to understand what you did and why.

3) There are at least three versions of Coco Basic, more if you include Dragon and MC10. So, if you intend to trade programs with someone, send the program in ascii format; CSAVE"name",A or SAVE,"name",A. If you don't do this, the tokens into which your program was converted may not be readable on the other system. You can even run some Coco Basic programs on a PC with GW-Basic if they are imported in ASCII format.

4) Study books like the Basic Unravelled series (yes, I know it's spelled wrong but that's the title used for the series) and look for ml routines that can easily be called from within a Basic program. Your programs will run faster. Give some thought to writing your own short ml routines to be called by your Basic programs.

5) Never define a constant with an equation. If you have a line like X=4:Y=6:A=2:B=SIN(A)*X/Y, you will be wasting valuable time while the interpreter calculates the answer. Just say B=0.6062 and be done with it. If there is some equation which you need to use several time in your program, either place it within a subroutine or use the DEF FN function to define it. That should speed up your program and decrease its size.
6) Don't forget that Basic trig functions use radian notation not degrees.

7) Not all shorthand notations are equivalent. You can place ' in a Basic line as a replacement for REM. You can do the same thing with ? Instead of PRINT. The results are not equivalent in the tokenized versions.

ASCII	Token
REM	$82
'	$3A83
PRINT	$87
?	$87

While you would think that ' saves space over REM by 1/3 it actually takes twice the space when used at the beginning of a line. However if your line is:

10 A=10: REM THIS IS A TEST

then

10 A=10' THIS IS A TEST

is the same size after tokenization and faster to type. That is because there is a required **:** before the REM so the tokens are $3A82 and $3A83. The use of ? for PRINT does not save any space after tokenization although there is less typing to enter the command.

8) Don't use complex code in place of simple Basic commands. For example, if you want an infinite loop or one that will continue until some condition is met, don't forget the STEP command.

```
…
100 FOR L=0TO1STEP0
…
200 IF X=23 THEN L=1:NEXT L
…
```

The above is equivalent to DO/UNTIL.

```
…
100 FOR L=0TO1
…
200 IF X=23 THEN L=-1:NEXT L
…
```

The above is equivalent to WHILE/WEND.

Similar loops can be used to create recursive routines just as can be done with more complex versions of Basic such as Basic09 under OS-9.

9) Well … I'll let you add your own "things of importance" to this list as you get more practiced with programming in Basic.

“Color Artifacing and a CoCo version of the IBM BASIC ‘MOD’ function”

by Retro Rick

You might know of a special function in BASIC for the IBM PC which is called “MOD”. It simply gives you what would be the “remainder” of a variable if it was divided by the number after the keyword “MOD”. Let’s say you gave the command:

```
B = A MOD 8
```

And let’s say that A was equal to 12. When you divide “A” by 8 - the number you see after the word “MOD” - you get the answer “1” with a remainder of 4. The “4” part is all that’s put into the variable B.

There are many ways to imitate this function on the CoCos. The one I’ll give in the program is written like so:

```
MOD=(X1>=96)*96:X1=X1+MOD
```

Here’s how the routine works …

1. The number for the MOD function will be called the “MOD Number”. In the program example, the MOD Number is 96.
2. We’re using the variable X1 to do our MOD imitation with.
3. We’ll use the variable “MOD” here. ECB actually reads it as “MO”, but as long as you don’t use the MO variable in any part of your code, you’ll be okay.
4. The variable “MOD” will equal “-1” if X1 is over or equal to the MOD Number.
5. MOD will be multiplied by the MOD Number.
6. Anytime X1 is less than the MOD Number, MOD will equal Zero. Otherwise, the MOD variable will always equal the MOD Number multiplied by “-1”.
7. Now the MOD variable is put to work. It’s added to the variable X1.

The code used here has a limitation - If X1 is ever greater than or equal to the (MOD Number)*2, the function won’t work right. The code never has X1 go beyond 96, so I left it like this to make the code run faster.

This code example would fix those problems, but it does the calculations more slowly:

```
X1=(INT(X1/96))*96)-X1
```

Now, for the next half of the article. It’s purely graphics-related.

If you’ve tried to do hi-res graphics in PMODE 4, you’ve no doubt run into the “color artifacing” stumbling block. Confound it, sometimes these colors are orange and blue, and sometimes they’re blue and orange!

The most common way to fix this problem is to store “Orange” and “Blue” into variables, and then show an orange or blue screen beforehand and ask the user what color it is. The code then changes the variables accordingly. However, when you make sophisticated use of combining one of those two colors with

the remaining two (Black and White - or “Buff” as the book calls it), the pixels seem “mis-aligned” whenever your code switches the orange and blue around. Argh!

Well, the good news is I’ve developed an alternative to this. The only catch is that you need to design your images so that there’s room to shift the image left or right one pixel in PMODE 4. Not a terrible limitation.

For CoCo 3 users, I’m sure the question is: “Why not just stick to CoCo 3 graphics modes instead, since they have more color and better resolution anyway?” Well, the answer to that question is simple: The more types of computers your program runs on (in this case, all three CoCo’s rather than just the CoCo 3s), the more popular your program can become. Proof of this is an IBM PC game released in the mid 1990s called “Mario!”. It actually ran without problems on all Windows 95 and Windows 98 machines as well as DOS-based machines as old as the 80286s (PCs sold back in the late 1980s). Plus, the file was so small a download, even people with extremely slow modems could download the game.

The program below draws a rotating blue slab. Graphics pages 2 through 5 are displayed, while the images of the rotating slab are on Page 1. There’s a different “angle” of the image every 16 pixels, and the program draws whatever image is one Page 1 from Pixels X1 to X1+15, and the last image goes from Pixels 80 to 95. 16 is added to X1 with each frame, and when the X1 coordinates are greater or equal to 96, X1 is set back to Zero again.

Note: PEEK(275) is the micro-timer of the CoCo, which never goes up to 255 and then back to 0 again. It goes to the next frame when 4 “tics” of this timer go by, thereby keeping the program from running “too fast” when the High Speed Poke is enabled.

```
1 GOTO1000
10 FORI=-1TO0:P=(PEEK(275)/4):FORY0=32TO15STEP-1:PMODE3:
     GET(X1,Y0)-(X1+15,Y0+Y),A:PMODE3,2:PUT(X0,0)-(X0+15,Y),A:GOSUB999:
     Y1=Y1+1:IFINKEY$>""THEN50
20 FORW=-1TO0:W=P=INT(PEEK(275)/4):NEXT:P=INT(PEEK(275)/4):NEXT
30 Y1=Y0+Y:FORY4=0TO191:PMODE3:GET(X1,Y0)-(X1+15,Y1),A:PMODE3,2:
     PUT(X0,Y4)-(X0+15,Y4+Y),A:GOSUB999:IFINKEY$>""THEN50
40 FORW=-1TO0:W=P=(PEEK(275)/4):NEXT:
     P=(PEEK(275)/4):NEXT:I=INKEY$="":NEXT:GOTO10
50 END:RUN
999 X1=X1+16:MOD=(X1>=96)*96:X1=X1+MOD:RETURN
1000 PMODE0:PCLEAR5:PMODE0:PCLS0
1010 DIMA(256):PMODE4:COLOR1:Y0=15:Y1=Y0+Y0-1:
     LINE(6,Y0)-(9,Y1),PSET,BF:PMODE3:COLOR3:
     LINE(22,Y0)-(24,Y1),PSET,B:PMODE4:COLOR1:LINE(25,Y0)-(26,Y1),PSET,B
1020 PMODE3:COLOR3:LINE(34,Y0)-(44,Y1),PSET,BF:COLOR0:
     LINE(45,Y0)-(45,Y1),PSET:COLOR3:LINE(48,Y0)-(63,Y1),PSET,BF
1030 LINE(66,Y0)-(78,Y1),PSET,BF:COLOR2:LINE(64,Y0)-(64,Y1),PSET:COLOR3:
     LINE(88,Y0)-(90,Y1),PSET,B:COLOR0:LINE(86,Y0)-(86,Y1),PSET
1040 PMODE3,2:PCLS2:COLOR0:LINE(0,0)-(254,8),PSET,BF:PMODE4,2
1050 DRAW"BM86,0;C0;R1D6L1R2U6R2D1R1D1L1D1;NL1;D1R1D1L1D1L1;
     BR5;U6R1D6R4;BR3BU1;U5R1D6R3U6R1D5;BD1BR3”
1060 DRAW"NU6;R1;NR4;U3;NR2;U3R4;BR3;BR10;BD6"
1070 DRAW"BR3;L1U1L1U1L1U2R1;ND1;U1R1U1R1;BR3;ND2;
```

```
     R1D3R1D3R1U3R1U3R1D2;BR3BD4"
1080 DRAW"R1U2R1D1U2R1U2R1D1U2R1;BR3;ND6;R1D2;ND4;
     R1D1R1D1R1;NU4;D2R1U6;BR3;R1D1R1D1R1D2L1;NU1;
     D1L1D1L1;BR6;BR2;R1;BU2;L1U1R2U1R1U1L1U1L3D1L1"
1090 SCREEN1,1:PMODE4:FORT=-1TO0:I$=INKEY$:T=I$<>"Y"ANDI$<>"N":NEXT:
     A=-(I$="Y"):FORX=80TO0STEP-16:GET(X,Y0)-(X+15,Y1),A,G:
     PUT(X+A,Y0)-(X+A+15,Y1),A,PSET:NEXT
2000 Y1=Y0+1:Y=Y0+1:Y2=88:Y3=Y2+16:X0=120:X1=0:GOTO10
```

Rom to Ram
By Rodney Hamilton

This one-liner for he CoCo 1 or CoCo 2 copies the ROMs to RAM, patches the RESET vector to return to RAM mode and change the “OK” prompt to “ok”. (requires 64k ram)

For the Dragon 64, you will need to change the address of the “ok” prompt in the 4th line from “ABEE” to “82EC”. If you don’t want to modify the prompt, delete the six bytes encoded as “CC6F6BFDABEE” and reduce the loop count from 52 to 46.

```
8 CLS:FORI=0TO52:POKEI+1017,VAL(
"&H"+MID$("12B7FFDF6E9CF71A508E8
000B7FFDEEC84B7FFDFED818CFF0025F
1CC6F6BFDABEE865597718E03F99C722
706DC72ED1E9F721CAF39",I+I+1,2))
:NEXT:EXEC1024
```

Hello Fellow CoCoists;

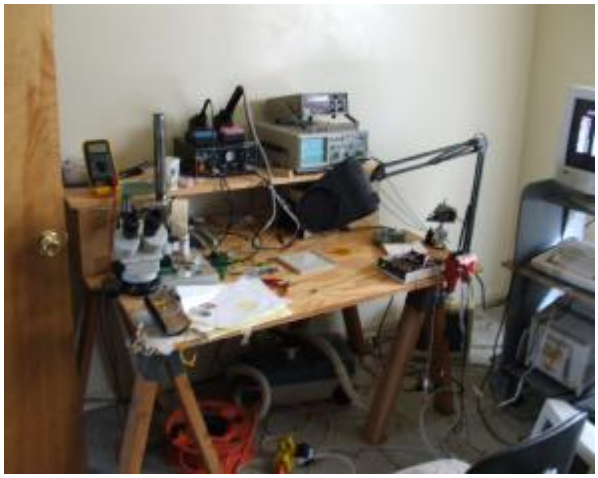
I am writing this short video converter update for Mary's newsletter. The design and schematic of the converter have been finalized pending feedback from all my beta testers. The feedback so far has been very useful. Rodney Hamilton made a great contribution fixing the problem of a weak contrast ratio. I have included several pictures.

First is my "lab". I'm not trying to scare you but if you enter here check to see if you have all your body parts before you leave. :) You can see my binocular microscope, soldering/de-soldering station, oscilloscopes, voltmeter and coco3 of course with SVGA and monochrome sub monitor. I will be spending a lot of time here depending on demand for the unit. I have concentrated on streamlining assembly of PC board and enclosure. I have templates of front and back panels showing where to drill and cut complete with bit sizes and connector positions. Next picture is the converter itself. It measures 6 inches square. Very compact and neat configuration and it works quite well. This particular unit is destined for Mark at cloud9.

The next picture is a screen shot of the working unit. A test pattern is on the SVGA monitor displaying all 64 colors at once by dynamically switching the GIME registers on the fly. It is a very demanding test for the unit which gives a rock solid display.

Before I start taking orders from the general public I am going to contact the people who attended the last fest who indicated their interest in the converter. After that it will be first come first serve. I am also going to try to have a few units to sell at the fest. It all depends on the demand. Also the price is set at exactly $48. Could go up or down slightly in near future depending on availability and price changes of components as time goes by.
Till then:

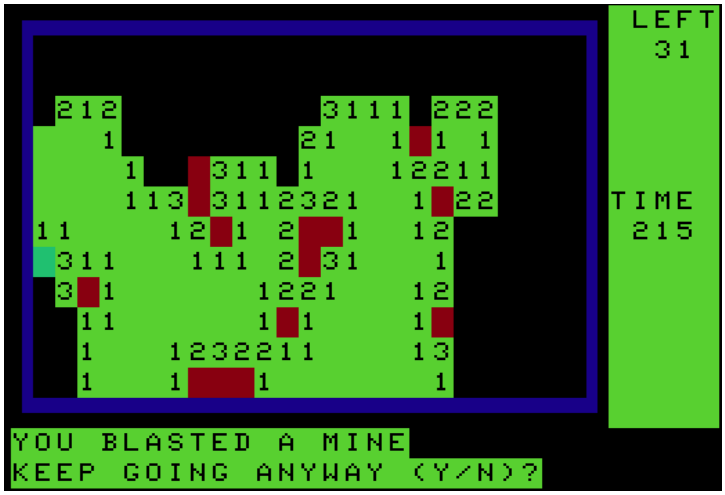
LONG LIVE COCO!!!!
Roy R Justus



Minesweeper
By Diego

This game is based on the popular "minesweeper" that's included with Windows since version 3.1. It should work in any CoCo, but will need a few small changes to run on Standard Basic (at least, the PLAY commands). It also needs a joystick. Sorry, if the coding is a bit messy. It's the way I like to program ;-)

```
10 WIDTH 32:REM NOT NEEDED ON COCO 1&2
20 POKE 65497,0:REM REPLACE FOR 65495,0 ON COCO 1&2
30 GOSUB 3000:REM INTRO & INSTRUCTIONS, NO NEED TO TYPE LINES 3000 TO 3230
40 DIM M(27,14)
50 MR=45:ML=45:REM MR IS MINES FLAGGED, ML MINES REALLY LEFT
60 FOR X=1 TO 27:FOR Y=1 TO 14:M(X,Y)=0:NEXT Y,X:REM CLEAR MINEFIELD
70 BL=0:HS=5000
80 FOR A=1 TO 45
90 X=RND(25)+1:Y=RND(12)+1
100 IF M(X,Y)=0 THEN M(X,Y)=9 ELSE 90
110 NEXT A
120 CLS 0
130 PRINTCHR$(161);STRING$(25,CHR$(163));CHR$(162)
140 FOR A=1 TO 12
150 PRINTCHR$(165);STRING$(25,CHR$(160));CHR$(170)
160 NEXT A
170 PRINT CHR$(164);STRING$(25,CHR$(172));CHR$(168)
180 TIMER=0
200 REM MAIN LOOP
210 H=INT(JOYSTK(0)/2.56)+1
220 V=INT(JOYSTK(1)/5.3)+1
230 S=PEEK(1024+V*32+H)
240 POKE 1024+V*32+H,223
250 FOR LO=1 TO 10
260 BT=PEEK(65280)
270 IF BT=254 OR BT=126 GOTO 500
280 IF BT=251 OR BT=123 THEN GOSUB 900
290 A$=INKEY$:IF A$="" THEN 320
300 IF A$=" " GOTO 500
310 IF ASC(A$)=13 GOSUB 900
320 NEXT LO
330 POKE 1024+V*32+H,S
340 GOTO 210
500 REM IS THERE A MINE?
510 PRINT@219,"TIME";:PRINT@251,INT(TIMER/60);
520 X=H+1:Y=V+1:C=0
530 IF S=191 AND M(X,Y)<>9 THEN MR=MR+1:PRINT@28,"LEFT";:PRINT@60,MR;
540 IF MR=0 AND ML=0 THEN TT=INT(TIMER/60):GOTO 4000
550 IF M(X,Y)=9 THEN GOSUB 2000:GOTO 210
560 FOR A=X-1 TO X+1
570 FOR B=Y-1 TO Y+1
580 IF B<2 THEN NEXT B
590 IF B>13 THEN GOTO 630
600 IF A<2 THEN NEXT A
610 IF A>26 THEN GOTO 640
620 IF M(A,B)=9 THEN C=C+1
630 NEXT B,A
```



```
640 IF C=0 THEN C=31:REM TO GET A SPACE IN THE NEXT POKE
650 IF Y=1 OR Y=14 OR X=1 OR X=27 THEN 670
660 POKE 1024+(Y-1)*32+X-1,C+112
670 IF C=31 AND Z=0 THEN GOTO 750
680 IF Z=1 THEN RETURN
690 GOTO 210
750 REM THE SQUARE WAS CLEAR
760 REM SO, SHOW ALL AROUND
770 Z=1
780 FOR CX=H TO H+2: FOR CY=V TO V+2
790 C=0:X=CX:Y=CY:GOSUB 560
800 NEXT CY,CX
810 Z=0
820 GOTO 210
900 REM PUT A FLAG
910 IF S=191 THEN RETURN
920 PLAY"T32EFE"
930 POKE 1024+V*32+H,191:MR=MR-1
940 S=191
950 IF M(H+1,V+1)=9 THEN ML=ML-1
960 IF MR=0 AND ML=0 THEN TT=INT(TIMER/60):GOTO 4000
970 IF ML=0 THEN PLAY"T12CDEFCDEFCDEF"
980 IF MR=0 THEN PLAY"T8FCFCFC"
990 PRINT@28,"LEFT";:PRINT@60,MR;
1000 RETURN
2000 PLAY"O1V31T8BDV<T>ACV<T>GA"
2010 PRINT@448,"YOU BLASTED A MINE";
2020 PRINT@480,"KEEP GOING ANYWAY (Y/N)?";
2030 A$=INKEY$:IF A$="" THEN 2030
2040 IF A$="N" THEN 2110
2050 IF A$<>"Y" THEN 2030
2060 BL=BL+1
2070 PRINT@448,STRING$(28,CHR$(128));:PRINT@480,STRING$(28,CHR$(128));
2080 PRINT@443,"MINE";:PRINT@474,"BLASTS";:PRINT@507,BL;
2090 GOSUB 900
2100 RETURN
2110 FOR X=2 TO 26:FOR Y=2 TO 13
2120 IF M(X,Y)=9 THEN PRINT@32*(Y-1)+X-1,"M";
2130 NEXT Y,X
2135 PRINT@448,"PRESS ANY KEY TO CONTINUE";
2136 EXEC 44539
2137 CLS 6
2140 PRINT@32,"YOU WERE ABLE TO FIND";
2150 PRINT@64,"A TOTAL OF"45-ML"MINES";
2160 GOTO 4080
3000 REM INTRO
3010 CLS 0:SCREEN 0,2
3020 PRINT@10,"'MINE CAMP'";
3030 PRINT@32,"DO YOU WANT INSTRUCTIONS? (Y/N)";
3040 A$=INKEY$:IF A$="Y" THEN 3060 ELSE IF A$="" THEN 3040
3050 RETURN
3060 PRINT@32,STRING$(32,CHR$(128));
3070 PRINT@64,"LAND MINES ARE ONE OF THE";
3080 PRINT@96,"GREATEST DANGERS CIVILIANS FACE IN AREAS WERE THERE IS, OR WAS  AN
ARMED CONFLICT.";
3090 PRINT@192,"BEFORE BEING DEPLOYED TO HELP UNAND RELIEF CONVOYS, YOU MUST";
3100 PRINT@256,"COMPLETE YOUR BASIC MINE-HUNTINGTECHNIQUES AT THE 'MINE CAMP'";
```

```
3110 PRINT@320,"USE THE RIGHT JOYSTICK TO MOVE";:PRINT@352,"THE MARKER, THE RED
BUTTON OR";
3120 PRINT@384,"THE space KEY TO DIG FOR MINES,";
3130 PRINT @482,"* PRESS A KEY TO CONTINUE *";
3140 A$=INKEY$:IF A$="" THEN 3140
3150 CLS 0
3160 PRINT@0,"AND THE BLACK BUTTON OR THE";:PRINT@32,"enter KEY TO PUT A FLAG AND
MARKA MINE.";
3170 PRINT@96,"EVERY TIME YOU DIG, A NUMBER";:PRINT@128,"WILL INDICATE YOU HOW MANY
MINESARE THERE IN THE SURROUNDING";
3180 PRINT@192,"TERRAIN";
3190 PRINT@258,"'MINE CAMP' V. 1.0, NOV 2005";
3200 PRINT@293,"DO YOU WANT TO SEE THE";
3210 PRINT@323,"INSTRUCTIONS AGAIN? (Y/N)";
3220 A$=INKEY$:IF A$="Y" THEN CLS0:GOTO 3060 ELSE IF A$="" THEN 3220
3230 RETURN
4000 CLS 6
4010 PRINT@9,"CONGRATULATIONS";
4020 PRINT@39,"YOU FOUND THEM ALL";
4030 PRINT@64,"IT TOOK YOU ";TT;"SECONDS"
4040 PRINT@96,"";
4050 IF BL=0 THEN PRINT"AND YOU DIDN'T BLAST ANY MINES"; ELSE PRINT"AND YOU
BLASTED";BL;"MINES";
4060 SC=TT+(BL*25):PRINT@128,"YOUR TOTAL SCORE IS"SC;
4070 IF SC<HS THEN 4110
4080 PRINT@192,"THE HIGH SCORE SO FAR IS"HS;
4090 PRINT"BY "HS$;
4100 GOTO 4140
4110 PRINT@197,"YOU HAVE THE HIGH SCORE... SO FAR"
4120 INPUT"WHAT'S YOUR NAME";HS$
4130 HS=SC
4140 PRINT@416,"DO YOU WANT TO PLAY AGAIN (Y/N)?";
4150 A$=INKEY$:IF A$="N" THEN PRINT:PRINT"BYE":END
4160 IF A$="Y" THEN 50
4170 GOTO 4150
```

Sorting OS-9 Directories
With a Basic09 Program

By Robert Gault

There have been several messages on the Maltedmedia newsgroup asking for programs that sort OS-9 directories. While you can find such programs on RTSI, this offers a perfect opportunity to learn some Basic09 programming.

Basic09 is a good choice of language because it is easy to learn and is almost as fast for this purpose as any other language that runs under OS-9. In addition, it is not that hard for users of Disk Basic to convert to Basic09.

To write a program of this type, it helps to divide it into parts. These parts help to keep things simple and serve as templates for more general use. After all, any mixture of items can be sorted, not just directories. Towards this end, the program has been divided into three parts: 1) general input to place data to be sorted into an array; 2) Shellsort - the sorting routine to be used which is simple to program and middle of the road efficiency; 3) a comparison routine specifically for OS-9 directories.

The study of the sorting process can be a lifetime work and many volumes of textbooks have been written about this subject. I won't attempt to justify the choice of shellsort but merely say that its speed is related to N^1.26 where N equals the number of items to be sorted. By comparison a bubble sort is an N^2 process and both quick and heap sorts are N*log2(N) processes. So for 40 items to be sorted: bubble=1600, shell=104, quick or heap=147. In the context of our project, shellsort holds it's own against the competition and could be the fastest sorting procedure.

To make any progress, we must know the structure of an OS-9 directory. All OS-9 directories have a one sector header (which we can ignore) and as many more sectors as are needed to contain the file entries. Each file entry consists of a file name left justified in a 29 character field and a sector address of 3 bytes. The last character of the name has $80 added to it.

```
directory entry         ASCII value
shell                   $73,$68,$65,$6C,$EC,$00,$00,$00, etc.
                                    $6C+$80
```

Directories can contain traces of entries which have been deleted. These are never removed but have the first character of the name changed to $00. All directories have .. and . as the first two entries.

Our sorting process will skip over the ../. entries so there is no chance of moving them. All upper case entries will move towards the start of the list and all deleted entries will move towards the end of the list. Since the OS-9 convention is to have directory names in upper case and files in lower case, all subdirectory entries will move to the front of the list.

The comparison routine must take two directory entries and compare each character until they can be labeled as smaller or larger. To reduce the comparison time to a minimum, the process should stop immediately if either test case is a deleted entry or if the end of either name has been reached.

The terminated name strings cause the comparisons to be more complex than expected as shown below.

Straight ASCII comparison:

```
arc < arctype
```

```
String terminated comparison:

arc > arctype  because  $61 $72 $E3   is larger than
                        $61 $72 $63 $74 $79 $70 $E5
```

Compenstate string terminated comparison:

```
arc < arctype      when string terminator found MOD(character,128) used
```

The modulus function in effect subtracts out the $80 for a normal comparison but prevents generation of negative values. Modulo math actually divides by the modulo value (mv) and multiplies the remainder by the mv; ex.

```
MOD(231,128)=(1.8047-1)*128=103
MOD(103,128)=.8047*128=103
MOD(359,128)=(2.8047-2)*128=103
```

Quirks of Basic09 and OS-9

The following procedures are entered and saved in Basic09 exactly as written except for the PROCEDURE lines. Use the same names (dirsort, shellsort, and compare) until you understand the language well enough to modify them. Once saved (save* dirsort), pack them into intermediate code (pack* dirsort). The packed modules will become a single merged file in your command directory.

A quirk of Basic09 makes the first module in a packed file the main line. Therefore enter the following procedure in the order given.

Some examples of usage with stock OS-9 are:

```
dirsort (".")
dirsort ("..")
dirsort ("/dd/cmds")
```

If you just enter '`dirsort`' a help message will appear. If you are using Shell+, or NitrOS-9 (which should have Shell+ installed), the ("") is not necessary and the syntax will be:

```
dirsort .          dirsort /dd/cmds
```

!!!! CAVEATS !!!!

Be sure to proof read carefully before running this program on a directory. If something goes wrong, you may lose the contents of the directory or a disk/drive.

It is simple and useful to modify the module dirsort to store the sorted directory list to a file. That will permit proving the program works without altering any directories. Change the following lines:

```
(* Write the sorted names back to the directory. *)
OPEN #path,dname:WRITE+DIR
(* Skip over the .. and . entries. *)
SEEK #path,64
FOR i=1 TO v
PUT #path,directory(i)
NEXT i
CLOSE #path

to these new lines:
(* Write the sorted names back to the file  dirsorted   *)
CREATE #path,"dirsorted"
```

```
FOR i=1 TO v
PUT #path,directory(i)
NEXT i
CLOSE #path
```

The new file dirsorted can be examined with dEd or Dump and compared to the actual directory. When you are sure that the program is working safely, restore the original lines. Delete dirsorted before each new run because attempting to create a file that already exists will give an error.

```
PROCEDURE dirsort
(* This program takes a directory name as a parameter. *)
(* The default directory is the current one. The directory *)
(* will be sorted with Upper case first then Lower case. *)
(* This will put subdirectories at the beginning of the directory. *)
DIM path,char:BYTE; i,v:INTEGER
(* All directory entries are 32 bytes with the name in the first 29, *)
(* followed by the lsn address. *)
TYPE filename=name(29),lsn(3):BYTE
(* This allows for 500 names in a directory. If you have more than *)
(* this amount in one of yours, you need to reorganize your drive. *)
DIM directory(500):filename
(* Trap a missing parameter. *)
ON ERROR GOTO 30
PARAM dname:STRING[80]
IF dname="" THEN
dname="."
ENDIF
(* If the directory does not exist, report an error. *)
ON ERROR GOTO 20
OPEN #path,dname:READ+DIR
(* Remove error trap *)
ON ERROR
(* Skip past the .. and . entries. *)
SEEK #path,64
i=0
(* Loop until entire directory has been read. *)
(* In Basic09, EOF is Boolean so can't be used as *)
(*   FOR I=1 TO EOF(#path)      *)
WHILE NOT(EOF(#path)) DO
i=i+1
GET #path,directory(i)
ENDWHILE
(* Save the number of entries. *)
v=i
IF v=0 THEN
CLOSE #path
PRINT "The directory is empty."
END
ENDIF
(* Sort the array of directory entries. *)
(* Pass the parameters to program shellsort. *)
RUN shellsort(directory,v)
(* Write the sorted names back to the directory. *)
OPEN #path,dname:WRITE+DIR
(* Skip over the .. and . entries. *)
SEEK #path,64
FOR i=1 TO v
PUT #path,directory(i)
NEXT i
```

```
CLOSE #path
END
20 PRINT "Directory does not exist"
END
30 PRINT "Syntax:   either a dot entry or full path is required."
PRINT "Example:  dirsort ."
PRINT "          dirsort /dd/cmds"
PRINT "Usage:    Sorts a directory in ASCII order."
END

PROCEDURE shellsort
(* Sorts an array *)
(* Based on "Sorting Algorithms for Microcomputers" BYTE, May 1983 *)
(* by T.Barron and G.Diehr *)
TYPE file=name(29),lsn(3):BYTE
DIM tx:file
DIM a,b:BYTE; d,i,j,match:INTEGER
(* Get array which may be partially full and number of entries. *)
PARAM entry(500):file; n:INTEGER
(* Basic09 syntax for base2 logs *)
d=2**INT(LOG(n))-1
WHILE d>0 DO
FOR i=1 TO n-d
(* Compare to items to see which is smaller. *)
RUN compare(entry(i),entry(i+d),match)
IF match<=0 THEN 20
tx=entry(i+d)
entry(i+d)=entry(i)
IF i<=d THEN
entry(i)=tx
GOTO 20
ENDIF
FOR j=i-d TO 1 STEP -(d)
RUN compare(tx,entry(j),match)
IF match>=0 THEN 10
entry(j+d)=entry(j)
NEXT j
10 entry(j+d)=tx
20 NEXT i
d=INT(d/2)
ENDWHILE

PROCEDURE compare
(* Compares two directory entries *)
(* Returns -1 if a<b, 1 if a>b, 0 if a=b which can only *)
(* happen with two deleted entries. *)
(* Deleted entries [first byte=0] are considered high *)
(* so that they are pushed to the end of the directory list. *)
TYPE file=name(29),lsn(3):BYTE
PARAM entry1,entry2:file; match:INTEGER
DIM a,b:BYTE; i,j:INTEGER
match=0
(* Scan through a full 29 character name. *)
FOR i=1 TO 29
a=entry1.name(i)
b=entry2.name(i)
(* Quit immediately if either entry was deleted. *)
EXITIF (a=0 OR b=0) AND i=1 THEN
IF a>0 THEN
```

```
(* Only B was deleted so match is Low. *)
match=-1
ENDIF
IF b>0 THEN
(* Only A was deleted so match is High. *)
match=1
ENDIF
ENDEXIT
EXITIF a<b THEN
(* Take into account a string terminated by adding $80 to last *)
(* character. *)
IF MOD(a,128)<MOD(b,128) THEN
match=-1
ELSE
(* False < because of last character in entry2. *)
match=1
ENDIF
ENDEXIT
EXITIF a>b THEN
IF MOD(a,128)>MOD(b,128) THEN
match=1
ELSE
(* False > because of last character in entry1. *)
match=-1
ENDIF
ENDEXIT
NEXT i
END
```

Sorting OS-9 Directories
With a Machine Language Program

By Robert Gault

Previously I presented a Basic09 program that sorted OS-9 directories. It had some limitations as it depended on runB being present. It was not "user friendly" with parameter entry unless Shell+ was installed. The program assumed that directories were as large as 500 entries which meant it required 16K of data memory regardless of directory size.

Converting the program to assembly language makes it simple to correct the above. The size of the directory to be sorted is obtained and data memory requested as needed. Parameter entry, regardless of the version of Shell in use, is now exactly like that for the dir command. No support programs such as runB are required. Of course speed has increased not only because runB does not need to be loaded but because it is machine language.

The source code is heavily commented so little needs to be added here. One interesting problem occurred which caused me much head scratching. The answer is applicable to any OS-9 ml program that uses OS9 F$Mem, a request for more data memory.

When an OS-9 program is started, the registers have the following meanings:

```
------------------------------ <-Y
| Parameter Area               |
------------------------------ <- X,S
| Data Area                    |
------------------------------
| Direct Page                  |
------------------------------ <- U,DP
```

regD= size of parameter area, regPC= module entry point

The minimum data plus DP area is 256 bytes, or zero if no data space is requested. Looking at the source code, you will see that the end of the requested data is indicated by Size equ . and it was assumed that the stack pointer starts at the value of size. However, since data space is allocated in integral pages, the exact location of the stack pointer is not at Size.

Since many programs do not request more data space, the location of the stack is usually safe. However, dirsort does request more data space to hold the sort buffer. When the content of the directory was read into the buffer, the stack pointer was over written causing the program to crash in spectacular fashion. To prevent this, the second instruction in the program relocates the stack to make sure it cannot be in the area of the directory buffer.

You may remember that the Shellsort routine required the use of natural logarithms. I can just hear the wails of despair, "Oh no, not only do I need to learn ml programming but I have to write code for logarithms!" Not to fear, because this is a dedicated program and not a general purpose math program.

It was easy to see that for any reasonable sized directory, there were very few answers to the equation D=2^LN(n) –1. You can see a table of n vs D values at the start of the program. Only eight entries were needed to cover directories of up to 2980 files.

This should be your standard technique for any programming that requires limited values from complex equations. Precalculate the results and use a table lookup. The size and complexity of your program decreases as the speed significantly increases.

I hope you find a one to one comparison of Basic09 vs Assembly code interesting. Examples of this type make learning assembly programming much easier than most other techniques.

```
* This is an asm version of a Basic09 directory sort
* that was published here earlier.
* Original and ml routine by Robert Gault, Nov. 2005

     nam dirsort
* This ensures the assembler knows predefined OS-9 terms
     ifp1
     use /dd/defs/defsfile
     endc

TyLg set Prgrm+Objct    program object code
* Re-entrant means multiple users possible
AtRev set ReEnt+Rev      re-entrant, revision 1
Rev  set 1

* Create module header, needed by all OS-9 modules
     mod Eom,Name,TyLg,AtRev,Start,Size
* Data area
DPptr   rmb 2 pointer to our direct page
dirsize rmb 2 size of the directory less 64 bytes
count   rmb 2 number of directory entries
entryI  rmb 2 pointer to nameI
entryJ  rmb 2 pointer to nameJ
entryID rmb 2 pointer to name(I+D)
Tx      rmb 32 buffer for transfer
i       rmb 2 index
j       rmb 2 index
NminusD rmb 2 holds last value of FOR I/NEXT loop
path    rmb 1 value for path to the directory
D       rmb 2 shellsort constant
        rmb 40
stack   equ .
Size  equ .   This initial data space will be increased as OS-9
* assigns space in pages of 256 bytes. Initially the stack will
* not be here.
buffer equ .  This will be the start of the data array in memory
* to be requested from OS-9 as needed.

Name  equ *
      fcs /dirsort/
      fcb 1   edition number
* Ego stroking :) identifier
      fcc /Written by Robert Gault, 2005/

* Default directory name, dot, or current directory
default fcb C$PERD,C$CR

* Solutions for N,2^INT(LN(N)) -1
* We don't need general logs just specific values so
* they were pre-calculated
Dtable fdb 2,0
       fdb 7,1
       fdb 20,3
       fdb 54,7
```

```
       fdb 148,15
* If your directory has more entries than several hundred, you
* need to learn how to organize your disk/drive.
       fdb 403,31
       fdb 1096,63
* This next could exceed memory limits
       fdb 2980,127
DTEnd  equ *

Start equ *
      stu  <DPptr  save the direct page pointer
      leas stack,u put the stack where we want it or it will be
* inside the directory buffer and crash the program.
      cmpd #0            are there any parameters?
      beq  noprm
l1    lda ,x+            skip over spaces, if any
      cmpa #C$SPAC
      beq l1
      cmpa #C$CR         if only spaces, same as noprm
      bne a1
noprm leax default,pcr   point to default directory, dot
      bra a9
a1    cmpa #'?           if "?" then show syntax
      lbeq syntax
      leax -1,x       backstep to first character of directory
a9    lda #%11000011  directory, single user access, update
      os9 I$Open      attempt to open a path to the directory
      lbcs error
      sta <path       save the path #
      ldb #SS.Size    get the size of the directory
      OS9 I$GetStt    size reported in regs X&U
      cmpx #0         MSB of size
      lbne Tlarge     too big to sort
      tfr u,d         evaluate the size
      cmpd #128       two entries other than .. and .
      lblo getreal    can't sort 1 item or less
      subd #64        reduce size by 64 bytes for .. and .
      std <dirsize    save size in bytes
      addd #Size      we need current data + directory buffer
      os9 F$Mem       request space for the buffer
      lbcs Tlarge     can't get enough memory
      lda <path       recover path to the directory
      ldx #0          MSB position
      ldu #64         LSB, past the entries .. and .
      os9 I$Seek      skip over the entries
      ldu <DPptr      recover DP pointer
      ldy <dirsize    data size in bytes
      leax buffer,u   point to our buffer
      os9 I$Read      transfer the directory information to buffer
* Calculate the number of directory entries
* Divide size by 32 bytes per entry. INT(size/32)must=size/32 or
* the directory is corrupted. So, ignore remainder.
      ldx #0          initialize counter
      ldd <dirsize
l2    leax 1,x        division is multiple subtractions
      subd #32        size of each entry in bytes
```

```
      bne l2
      stx <count      the number of names in directory
      leax Dtable,pcr  precalculated constants
      leay DTEnd,pcr
      pshs y
      ldd <count
l3    cmpd ,x
      bls a4          if fewer or equal # of entries get
D
      leax 4,x        move to next table entry
      cmpx ,s         have we exhausted the table?
      bne l3
      leas 2,s        restore the stack
      lbra Tlarge     should not be possible to get here
in code
a4    leas 2,s        restore the stack
      ldd 2,x         get shellsort D from table
      std <D          save working value
* Sort starts here
* Directory entries can't have duplicate names or the
directory
* is corrupted. That means a<b is as good as a<=b when
testing.
s2    ldd #1         initialize FOR/NEXT loop
      std <i
      ldd <count      same as n in Basic09 program
      subd <D
      std <NminusD    save value
* calculated pointer for entryID
s6    ldd <i          FOR i=1 TO n-D STEP 1
      addd <D         get pointer for entry(i+D)
      lbsr point      get the pointer value
      stx <entryID
      tfr x,y
* calculate pointer for entryI
      ldd <i
      lbsr point
      stx <entryI
* Compare the entry pointed to by regX against that for
regY
      lbsr compare    is name(i) < name(i+D)
      bcs s20
      ldx <entryID
      leay Tx,u       shellsort swap name holder
      lbsr movexy     name(Tx)=name(i+D)
      ldx <entryI
      ldy <entryID
      bsr movexy     name(i+D)=name(i)
      ldd <i
      cmpd <D
      bhi s4          this was a Basic09 IF/THEN
      ldy <entryI     inside the IF/THEN
      leax Tx,u
      bsr movexy   name(i)=name(Tx)
      bra s20        ends the IF/THEN
s4    ldd <i        initialize FOR/NEXT loop
      subd <D
      std <j
s5    bsr point    FOR j=i-D TO 1 STEP -D
```

Xenion by Michael Duncan, Sold by Diecom Inc

Review by Brian Palmer.

Xenion was the first CoCo 3 space game written to showcase the graphics ability of the CoCo 3 and was programmed in 1987.

This game was different to any other game in this genre. This game had a lot of detail done for background scenes. Most other games had sparse scenery, but not this one. From start to finish, there is great detailed scenery; from flying over the Alien Cities, to flying in space, to battling Tanks and force fields on the ground. Two players can play, two types of special bonuses can be found, smart bomb icons, and energy shields.

The only fault I have found in this game is the sound effects could've been better. Then again the graphics make up for this.

This game is a cross between F16 Assault and Zaxxon, but had more Baddies to Kill, and better all round action. This is a must have game for any serious game player. When you get a copy, watch for how many alien ships look like fighter planes from Movies like Battlestar Galatica to StarWars.

```
      stx <entryJ
      tfr x,y
      leax Tx,u
      lbsr compare   is entry(Tx) > entry(j)
      bcc s10
      ldd <j
      addd <D        name(j+D)
      bsr point
      tfr x,y
      ldx <entryJ
      bsr movexy   name(j+D)=name(j)
      ldd <j        NEXT j
      subd <D       STEP -D
      std <j
      cmpd #1       stop if less than 1
      bge s5
s10   leay Tx,u
      ldd <j
      addd <D
      bsr point
      exg x,y
      bsr movexy   name(j+D)=name(Tx)
s20   ldd <i       NEXT i
      addd #1      STEP +1
      std <i
      cmpd <NminusD
      bls s6      stop if i>(n-D)
      ldd <D       D=D/2
      lsra
      rorb
      std <D
      cmpd #1
      lbhs s2      WHILE D>0
      lda <path    rewind to just after .. & .
      ldx #0
      ldu #64
      os9 I$Seek
      lda <path
      ldu <DPptr
      leax buffer,u write out sorted directory
      ldy <dirsize
      os9 I$Write
      clrb
      os9 F$Exit   release memory, close paths, and return to OS-9

movexy ldb #32   move the entry pointed to in regX to
sw1   lda ,x+    that pointed to by regY
      sta ,y+
      decb       if not finished, continue
      bne sw1
      rts

* Converts an index in regD to a memory offset in regX
point pshs y
      leax buffer,u
      ldy #-32   initialize offset, convert base1 to base0
p1    leay 32,y  calculate offset=index*32 -32
      subd #1    update index
      bne p1     continue if not finished
```

```
      tfr y,d
      pshs x
      addd ,s++  add offset to buffer pointer
      tfr  d,x   regX now points to name
      puls y,pc

compare ldb #29  size of name field
      pshs b     save counter
cloop ldb ,y+    get character
      lda ,x+    get character
      beq c1     if deleted go
      tstb
      beq c2     if deleted go
      tsta
      bmi c5     if last character go
      tstb
      bmi c6     if last character go
      pshs b
      cmpa ,s+
      beq  c4    if equal, test next character
      bra cx
c1    clra
      bra cx     return +
c2    coma
      bra cx     return -
c3    anda #$7f
      andb #$7f
      pshs b
      cmpa ,s+
      rts
c5    bsr c3
      beq c2
      bra cx
c6    bsr c3
      beq c1
      bra cx     return result
c4    dec ,s
      bne cloop
      clra       should not be able to get here in code
cx    puls b,pc  return result


error leax nodir,pcr
      ldy #endnd-nodir
write lda #1     screen
      os9 I$Write
      clrb
      os9 F$Exit
nodir equ *
      fcc /Directory does not exist!/
      fcb C$CR,C$LF
endnd equ *

Tlarge leax big,pcr
      ldy #endbig-big
      bra write
big   equ *
      fcc /Either the directory is too large or there is insufficient /
      fcc /memory./
```

```
      fcb C$CR,C$LF
endbig equ *

getreal leax huh,pcr
      ldy #endhuh-huh
      clr ,-s
      bra write
huh   equ *
      fcc /Get real! You can't sort less than 2 items./
      fcb C$CR,C$LF
endhuh equ *

syntax leax usage,pcr
      ldy #enduse-usage
      clr ,-s
      lbra write
usage fcc /USAGE: dirsort will sort any directory. If no directory/
      fcb C$CR,C$LF
      fcc /       name is given, the current directory will be sorted./
      fcb C$CR,C$LF
      fcc /EX:    dirsort    dirsort .    dirsort ../
      fcb C$CR,C$LF
      fcc "       dirsort /dd/cmds"
      fcb C$CR,C$LF
enduse equ *
        EMOD
Eom   equ *
```

“Pegasus and the Phantom Riders”

Game author: David Figge (a.k.a. The Snail)
Publisher: Tandy (Version 1.0) and Snail Corp. (Version 3.0)
Runs in PMODE 4 on 64K+ CoCo’s
Review by Richard Kelly

Introduction: Let’s say you’re looking for a game that plays somewhat like the Joust arcade game, but you want it enhanced a bit. If you have a CoCo, this is basically the only game you have to choose from.

“Pegasus” exists in at least three versions. There’s the RS-DOS version, where you start the game by using the LOADM and EXEC commands. This is the newest version available, and was not sold in the U.S. as far as I know. The version was sold in Brazil, and probably in other areas outside of North America as well.

The second version is the most popular – the U.S. official version which you started up either by entering in the DOS command, or – if you had DECB 1.0, you could use this command as an alternative:

RUN”PEGASUS”

A third version exists which is basically a cracked version. The “cracked” credit message appears along with the name of the game when the program reaches the title screen.

When the title screen does appear, note the color of certain lines of text. They should be either blue or “red” (displaying on all TVs as orange, to be honest). Start the game when the text is blue, and the background of the game screen will have a blue sky.
If you select a two player game, you have the option of playing “Duel” or “Cooperate” Mode. Duel Mode is like Joust – The two players can kill each other as well as the enemies. Cooperate Mode allows the players to pass through each other, so Player 1 and Player 2 can’t kill each other by accident.

You start the game flying a Pegasus, and the “Phantom Riders” fly into the scene from a mountain to do battle. Your opponents are killed by making contact with you when your head is above theirs. If the opposite is true, then you’re the one who dies.

The level starts with two halves of a hill on the bottom left and right, and a deeper level of ground in between the two. There’s quite enough room to get your Pegasus a full running start before you start flying.

Pass two levels, and you enter the second phase. There’s a closer view of the water and the mountain, and you now see sharks swimming around. The ground below has a “gap” in it where you can fall in the water, although it’s a little too easy for you to avoid. Turns out your enemies can fall in the water themselves, and often do.

Pass two more levels, and you enter Phase 3. Now there’s an even more detailed mountain view, plus there’s a water beast that breathes fire into the air to kill your Pegasus. The “gap” is still exactly the same size as before, and it evens appears in the same place.

If you manage to pass a few more levels after that, you’ll get a sort of “Challenge” stage, where you come in full view of the waterfall. There’s no ground where you can run around. In fact, you appear on a small platform that disappears only seconds after your Pegasus comes into view. If you manage to pass this stage without losing any lives, you get a bonus.

If you have a Sound-Speech Cartridge, then the game will talk to you. Supposedly the sound effects are more realistic with the cartridge as well, but I personally noticed no difference at all in the sound when I played the game.

Graphics: C+. The graphics always do the job fairly well, but for a game with this sort of color setting, the graphics look rather blocky. The background is always blue (or “red”), which means you’ll never get those ultra-sharp images you’d see if there was white on a black background, or vice versa. It sort of defeats the purpose of giving the game a PMODE 4-based color scheme in the first place. It would probably look just a good with the screen in PMODE 3, and you wouldn’t have that color artifacing that makes the first two CoCo’s so infamous.

Sound: C+. No sounds seem to be out of place here, but they’re nowhere near as impressive as ones I’ve heard in many other CoCo releases. There’s just no sound in this game that really grabs your attention. Also, Steve Bjork has proven (with the game Megabug) that the CoCo’s PCM sound can “speak” better than the Sound-Speech cartridge can.

Animation: B+. This has some nicely animated sequences. Take the Pegasus running off the hill and jumping down, or turning very smoothly from one direction to the opposite one. Let's not forget those swimming sharks I mentioned earlier. Not to mention the monster popping its head out of the water, then breathing a few fireballs in your general area. But my personal favorite bit of the animation here would be the view of the waterfall in the challenge stage. *Very* nicely done.

So why not give the animation an A+? Well, the flying "bats" from the mountain will abruptly change into Phantom Riders, rather than zoom in the image a little more, then smoothly turn left or right, like Player 1 and Player 2 do. The Riders' entering of the battle seem kind of patchy this way, since no other sprite in the game is animated like this.

Game engine: A. There's no obvious bugs of any kind, and the frame rate never decreases. It does show its limits in higher levels though, when you realize that there's never more than four Phantom Riders on the screen at one time.

And here's another problem: *Very* stupid enemies. In a nutshell, they pretty much fly up and down, and rarely do little else. They never change their attack patterns … not even to avoid drowning. Given this, it's a small wonder that the "challenge stage" wasn't much of a challenge to me, since there's no ground in the level to save these airheads from drowning out of their own stupidity.

Gameplay: C. The best part about the gameplay is that the animation enhances it. Since the Pegasus takes time to turn around, it forces you to adjust your gameplay a little bit, and plan ahead a bit more, lest you find yourself in an impossible situation. This is the way that the truly best game animation works.

Another highlight is that you get extra points if you catch the falling crosses *before* they hit the ground. It encourages you to enhance your strategy, so if simple survival isn't enough for you, that's something extra you can focus on.
However, the variety of enemies in Pegasus is very low. And although Phase 3 *looks* different, the layout of the level is exactly the same; there's a fire-breathing monster in the water added in, and enemies that haven't come in to attack you yet are still flying around the mountain in zoomed-outform. But these things are just here to disguise the level for what it truly is – Phase 2 with a different background.

Even worse – there's only three phases. After that, you'll get a challenge stage every once in a while (starting at Wave 11), but otherwise you're just playing the same levels in Phase 3 all over again. The enemies don't speed up after a certain wave, they never attack in a larger group than four, they don't get smarter … there's just nothing more to see.

At least in the Joust arcade game, there were structures that crumbled away, adding at least some variety. Joust also had more types of enemies. The fact that "Pegasus" has neither of these features really hurts the entertainment value. When the player makes progress, he isn't rewarded with new

types of experiences in the game. Instead, he finds himself brought through an endless set of levels that never vary.

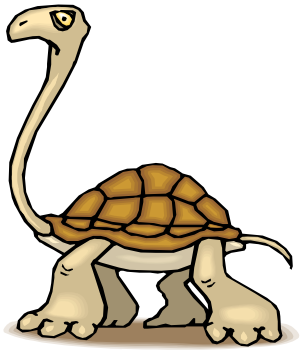
Overall: C+. This game seems to be based upon the idea of enhancing the concept of the Joust arcade game, and I think the enhancement concept was a pretty good idea, even though I've always enjoyed the original Joust as-is. Tragically though, it's my impression that "Pegasus" kind of had its development cut short, and was released before the author finished what he really wanted to do with the game.

The first tell-tale sign of this is where you see the Phantom Riders flying in - how they abruptly change from "bats" to flying black Pegasus. The second piece of evidence would be Phase 3, where the level layout replicates Phase 2 rather than enhances upon it. The third and final bit of proof is the poor variety of enemies, and lack of increased difficulty in the higher levels.

I suspect that Tandy said to the author somewhere in the middle of production: "Hey, this product is taking too much time and money to develop. Just finish it up so we can get it out the door." So the author did just that. Wasn't his fault if that's the case – he was just doing what he was paid to do.

So as far as "Pegasus and the Phantom Riders" is concerned, what we've got here is pretty much an average game. I say, borrow a copy for a day just so you can get a few kicks out of it. Then, dig out a copy of MAME and try a better enhancement of Joust – the arcade game "Joust 2: Survival of the Fittest". You'll be glad you did.

Show Off Your CoCo!



Boisy Pitre

Guys,

I have an elaborate CoCo setup that is hard to capture in pictures, but I'll try to give you some idea of how it looks and works.

This is officially "Lab South," the southern extension of the Cloud-9 enterprise. Overall, I think that I have over 10 different CoCos in this little area. For product development and testing, I keep a CoCo 2 and a CoCo 3 up and running; the rest are spares, and also come in handy for other projects from time to time.

The first picture is of my development CoCo 3. I have a CM-8 as the active display with a Commodore 1084 on the side for backup purposes. The shelf, built my my wife's uncle, works well to suspend various components above the CoCo system. You can see a 3.5" and 5.25" drive pair in an FD-502 case, as well as a hard drive, LS-120 and a CD-ROM drive sitting atop the bare power supply. The CoCo 3 is underneath. Since its a development machine, the top stays off. It has a Cloud-9 512K memory board with Protector+ and 6309, as well as a Cloud-9 PS/2 keyboard adapter. The Multi-Pak contains a Sardis floppy controller and a SuperIDE.

The second picture is my 64K CoCo 2 setup. LIke the CoCo 3, it has a shelf with two displays. One is a 14" RCA TV with S-Video and composite connections, the other is a green monochrome monitor with composite input, useful for Word-Pak displays. The CoCo 2 has a Multi-Pak with TC^3 SCSI Controller, Disto HD2 controller and Disto Super Controller (floppy). A single 5.25" floppy drive is connected
to the system.

Also on the shelf are four hard drives and a CD-ROM drive that connect via the TC^3. The drives range in size from 200MB to 1GB and are all formatted for use on the CoCo. And yes, the CoCo 2 is connected to ALL FIVE of these drives. This actually serves as a testbed for SuperDrivers, HDB-DOS and other software that Cloud-9 sells.

To give you an idea of where these CoCos sit relative to my shop, I've taken a third picture. The pegboard allows me to conveniently hang tools, parts and other things that I need for testing and development. A PC system running Linux sits in the corner, and holds all sources for the NitrOS-9 Project, ToolShed and Cloud-9 products. I can easily build a bootable floppy disk (3.5" or 5.25") on this Linux system, put the floppy in a CoCo and boot up to test drivers and other software. I think that Linux is a great system for doing CoCo development.

At the very top I've placed shelves to hold spare CoCos in the box. I also have a Dragon computer up there. Speakers are on either side and I have a Sony receiver (hidden by the metal cabinet door at the bottom right of the picture) so I can listen to music while I work. Speaking of that, I have two metal cabinets full of CoCo parts, software and other collectibles.

On the floor under the bench I have a 25" Sony Professional Monitor that I've modified to hook up to a CoCo 3's RGB output. It is VERY heavy, but it puts out a GREAT and HUGE picture. 80 columns on a CoCo 3 never looked so good!

See that little box on the top right side mounted on the stud with the little black antenna protruding from the top? That's a 1900MHz digital packet repeater. I had to purchase one of those in order to use my Sprint PCS phone in the building, since the metal was killing signal inside the shop. This little gadget sprays signal all over the shop area, allowing me to talk to Mark back at Lab North while working. We collaborate on the phone quite a bit (thanks to free Sprint-to-Sprint calls).

Cloud-9 'Lab South'

All of this is in the back of a 40'x50' shop that I built behind our house. This is where the CoCo magic happens! I keep busy in here quite a bit, and always am testing new ideas for the CoCo. Of course, the SuperBoard will be party born here over the next few months. It's the perfect place to tinker, try new things, and have fun with a great machine.

Richard Ivey

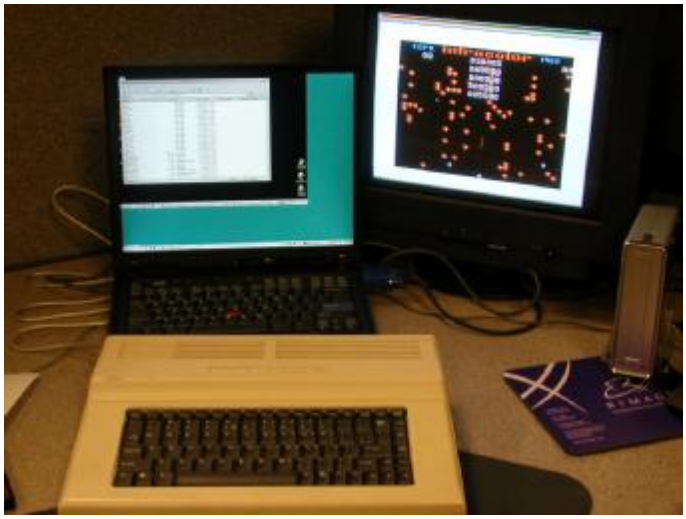
A CoCo - *Texas Style!*

I got my first Color Computer 2 for Christmas 1984 with a CCR-81 tape player. We had just moved to a much larger town and I was pretty bummed out, so my parents got me this to help make up for moving me. I was hooked. At that point, my NES and Atari 2600 pretty much went into hibernation. I got a 5 1/4 drive for my birthday 5 months later and a Multi-Pak Interface and Speech Pack the following Christmas.

My first games were Popcorn and Polaris (not the *worst* Missle Command clone). I never got that good at programming. I could program in BASIC and somewhere in a Rainbow magazine issue is a submission from me. As I got older and started chasing girls and went to college, the computer got packed up. About 5 years ago, I unpacked it and started playing with it.

I had bought a Color Computer 3 in 1987 but had sold it in a garage sale, so I bought one off ebay for $5. It got packed up again and recently unboxed since I bought a house and have more room. I have since bought another floppy drive, some games, and a Silver Case CoCo 1 still in the box. I have 3 deluxe joysticks and about 4 of the crappy black kind. I hope to upgrade the keyboard and memory in the CoCo 1 at some point in the future. I have also gutted a dead CoCo and put in a ps2 keyboard and it is now hooked to my docking station at work.

I am currently adding a 3 ½ inch drive into my 501 enclosure to transfer my old stuff off aging 5 ¼ media.



John Schuster

One photo won't really do it, but here is my main CoCo/emulator setup. The second picture is my main pc and my original CoCo 1 where I waste incredible amounts of time playing Mega-Bug. LOL.
John

Mary Kramer's CoCo

I have the basics, the CM-8 monitor, Multi-Pak interface, 2 floppy drives(5 ¼ and 3 1/5) and of course my CoCo 3. I have got to have the laptop for M.E.S.S in there too.

The second picture is of my second CoCo. He is my baby; see how he likes to sleep under his blankie! This CoCo is much younger than my machine CoCo. Baby CoCo is only 6 months old right now. Make sure you take careful notice of my budding artists' work hung nicely on the wall behind my coco. Jacob loves to show off his paintings, who knows maybe I will sell them on ebay when he is older!

Bob Devries aka: Aussie Bob

This is a CoCo - *Aussie Style!*

Diego’s System

My first CoCo was a 16 Kb CoCo 2 (ca 1985). I had it for about a week, until I realized that very few games would work on it. I swiftly changed it for a 64 Kb model. About a year and a half later, I upgraded to a CoCo 3. I used them with a CCR-81 and standard joysticks. In the early 90s I was able to buy the Orchestra-90 and the Sound/Speech pack.

When I moved to the USA, I had to leave them all behind, but as soon as I found e-bay, I was getting CoCos. The one I'm using all the time is a 512 Kb CoCo 3 from Cloud9. For storage, I have a FD-502 controller, and 2x360 drives in a 501 case, plus a trusty CCR-81 (mostly to save games). In the MPI there is an Orchestra-90 and a S/S, leaving me an empty slot for games (backgammon anyone?).

The printer is a DMP-106, which is running short on ink, but still enough to print my listings. I have the 4 styles of joystick that AFAIK were available from RS, plus a 1 button mouse and a koala touch pad. This is a problem ,since all of them use the joystick ports. That means a lot of switching plugs. I have to build some kind of multi-port adapter.

The saddest part of the setup is the monitor. A Commodore that only does composite and distorts the image at the bottom. I'm hoping for a SVGA adapter. The DC-3 Modem and the "plug 'n power" are just for display. I don't have a phone line, nor the data cable for the p'n'p. Most of our time together is using BASIC, but sometimes we go for Nitros/OS-9. I'm looking for a hard drive to really get NitrOS-9 going.

Whew! This issue was cutting it close to the deadline. I just want to say thanks to all that contributed. Thanks to you guys, this issue has once again been a big success. I will keep this going as long as there is a reader and people that contribute for it. Remember this is a community event! People from all around the world contribute to this. I think my dad would be very proud of me for providing a means for most of you to start actually using your CoCos again and taking interest in rescuing others off of ebay!

Some of you fellars are beginners with all this like I am! I would like to see more from you guys as you progress with your skills and learning of programming, etc. I do not write this newsletter... I simply compile a bunch of works of many others into one simple format. (although there wasn't much simple about this issue format!) J

So thanks again and let's get more people involved in this, and pass on the love of our machines! So enjoy this issue and please give me both negative and positive feedback.

Your Editor,
Mary Kramer