March/April1998 Vol. 5 Number 5 $4.00US (North America)

the world of 68' micros

Support for Motorola based computer systems and microcontrollers, and the OS-9 operating system

Rediscovering ... or discovering ... the Tandy Color Computer. A guide for old users returning and new users just learning.

CONTENTS

YOU CAN NOW GET NITRO FOR FREE IF YOU HAVE INTERNET ACCESS! IF YOU DON'T, CHECK OUT THE NEW LOW PRICES ON NITRO PRODUCTS IN THE FARNA SYSTEMS AD ON PAGE 9.
See editorial in this issue for details

Don't forget to make plans for the Chicago CoCoFest!!! PennFest coming again in August!!

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

If your address is incorrect, send me a postcard!

The Editor's Page

WANTED: EDITOR FOR "68' micros" MAGAZINE. Must have good general knowledge of the CoCo and all hardware, interest in OS-9 and OS-9/68K machines. Must have at least a 486 clone with 8MB RAM and 40MB free harddrive space for necessary software. Should have some experience with DTP software (software will be provided). Must have reliable internet connection or ability to get one. Token payment will be made to editor -- negotiable. Editor will be required to gather information and layout pages on provided template. Publisher will maintain mailing database, advertising, shipping, and production. Write or call Frank Swygert c/o FARNA Systems (mailing and e-mail address below), 912-328-7859 5-10pm EST weekdays, 9am-10pm EST weekends. All applicants considered.

Does this mean I'm getting out of the CoCo magazine "business"? Not at all! It is just that job, famliy, and other hobby considerations are putting a crunch on my time! I simply don't have time to do everything, and when I change jobs in the next year time will be tighter. **So I'm looking for an enthusiast to share the work while receiving some "hobby money" compensation.**

I'll still write a few articles and go over final editing of the magazine. I expect the editor to do 75% of the layout, I'll be doing the rest. At least in the beginning. I'll be willing to let them have complete editorial and creative control after a few issues if the work is agreeable to me. I'm not abandoning the community! So if your'e interested, give me a call! I'll accept e-mail and written inquiries, but will want to talk to everyone before a decision is made. If you've done flyers and such before, send a sample along also.

If I don't get any inquiries, the magazine will continue. There will be more times that it will be late (like this one is!) and quality may show more rush in production (more gramatical and spelling errors). And I may be forced to go to a quarterly printing schedule. I'd much rather retain quality and reduce quantity than keep the magazine bi-monthly while overall quality degrades.

I won't be at Chicago this year, though I am sending some items along for Carl Boll, who will be representing FARNA Systems in Chicago. My military unit had to deploy some people to help support a base in Saudi Arabia (routine rotation). I didn't have to go, but we are left with a few people handling a large workload! Our people deployed in two sections of 100, each going for two months rather than one section staying four. About the time of the Chicago fest almost everyone is gone, either in route to or from Saudi Arabia, so we are left with a few people covering base maintenance. Naturally, we need all available hands, so I couldn't take off for the show (this also contributed to lateness of this issue -- couldn't take any time off to finish the magazine and some other tasks). First time this has happened in about six years of Chicago fests, so I can't complain much!

For those who do make the fest, I hope you have a great time. And note that **Ron Bull will be hosting a fest in Pennsylvania this August** too! It was looking bleak for having another in PA, but Ron got some assistance recently and it is a go! Timing is bad for me to personally attend, but FARNA Systems will be represented.

NitrOS-9 is now a free-ware product. Sales have been low since the last CoCoFest in Chicago. This was one reason Alan Dekok decided to make it freely available on his ftp site (ftp.striker.ottawa.on.ca/pub/nitros9/). He will release source code to those who express an interest in assisting with future development and has the ability to do so (the source is not freely released). FARNA Systems will continue to distribute disk copies with known to work modules and a simple installation file at a reasonable cost. This will still be the best way for the novice user to get Nitro.

the world of 68' micros

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

Editor:
Francis (Frank) G. Swygert

Subscriptions:
US/Mexico: $24 per year
Canada: $30 per year
Overseas: $50 per year (airmail)
Back and single issues are cover price. Overseas add $3.00 one issue, $5.00 two or more for airmail delivery.

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

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

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

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

All trademarks/names property of their respective owners

ENTIRE CONTENTS COPYRIGHT 1998, FARNA Systems

Reader's Write

Do you still support the TRS-80 Color Computer? If so, may I have some information?

Can you provide the pin-out of the 512K expansion connectors inside the CoCo 3? I want to design a 512K static RAM upgrade with a battery backup. Actually, preliminary design is complee -- just need connector pin-outs.

Robert Turner
Box 62
Riverside, AL 35135-0062

I received this letter a few weeks ago. Robert was sent a sample issue and the pin-outs he requested. He was also asked to submit his design for publication. Hopefully we'll hear from him again!

I own a TRS-80 Color Computer 2 with 64K and a TRS-80 Line Printer VII. I have no idea how to use them. I intend on restoring them with all new equipment, hardware and software. I would like a catalog if you have one. I will keep in contact for all my needs.

I need:

Printer ribbons
Printer cable
RF connector
Cartridges
Disk Drive
Tape Drive
Modem
Users manuals (computer and printer)
Any other items that might be useful and/or fun

Jason Mills
823 Main Street
Monmouth, ME 04259

There are still young users discovering the CoCo! Jason will be sent this issue. Always encourage young computer explorers! There just isn't enough challenge in the new "appliance" boxes.

I enjoy the magazine very much and look forward to another year or more! I would also like to know if microdisk is still available. How about a contents listing? The lst microdisk I received was volume 3 issue 5 and 6.

Best wishes and good luck for you and your wife. I do hope you make it to Chicago.

Vern F. Larson
598 Riverton Avenue
Winnipeg, MB R2L 0P1
CANADA

Microdisk was discontinued after volume 3 due to lack of interest. All those who had subscriptions to microdisk had their magazine subscriptions extended by an appropriate amount. If you need a program from the magazine that you can't get any other way, send a check for $6 with the name(s) of the program(s) you need and the issue. There just weren't enough subscribers to microdisk to make it worthwhile compiling disks for every issue.

What do I need to do to get my 3.5" disk drive to work as drive 2 on my 512K CoCo3? I have been unsuccessful so far. My 5.25" drives (DS/DD) work fine. Do I need to do some pokes?

Earle D. Eason
2612 NW Pitkin Lane
Huntsville, AL 35810-3510

3.5" 720K disk drives are electrically the same as 5.25" drives even though the connectors are different. There should be a jumper designating the drive as 0 or 1 (or A or B, or 1 or 2). That jumper needs to be set as the second drive. (1, B, or 2). If there are no jumpers, the drive was made for an IBM clone that uses a twisted cable. For you this obviously isn't the case, as IBM set all drives as the second drive then twisted the cable to make one the first drive. Make sure you have the power lines connected properly. The yellow is +5V, red +12V, and both blacks are ground.

The only other possibility I can think of is that you have a 1.4M drive and not a 720K. 1.4M drives spin faster than 720K's and have a faster data transfer rate. A very few have a jumper that slows them down to the standard 720K speed, but for the most part spin rate is not adjustable. A few automatically slow down, others compensate for the faster spin rate in the electronics and stay at the higher speed. The location and marking of the jumper (if used) varies with manufacturer, so I can't really help much there. The only thing you can do is call the manufacturer or see if they have a web site.

You might want to take the drive back if it isn't a 720K or isn't jumperable to the lower speed. 720K drives are still available used and remanufactured through several parts dealers. Try All Electronics Corp. (P.O. Box 567, Van Nuys, CA 91408 -- 1-800-826-5432). They usually have remanufactured or good used 720K drives available at a low price.

Color Computer Users Guide

By Lee Deuell

Originally written June 10, 1991

Edited and revised for 268'm March 1998

Editor: Lee originally wrote the following as a "continuity manual" for a possible inheritor of his CoCo3. I edited it to include ALL CoCo models. Lee suggests that everyone make such a report on their particular setup. This is a great idea! Many people get CoCo's at yard sales and thrift shops and have no idea what to do with them. Some find 268'm through contacts with Radio Shack, others find the CoCo List on the internet. We eventually get them help, but a short letter or info folder sent along with the CoCo sure would be nice! Who knows, maybe your grandkids or even great grandkids will find a box in the attic one day with your old CoCo! Wouldn't it be nice if they would also find in that box a simple set of instructions on how to get it up and running? Not a complete manual, but a "quick reference guide" to get it up fast without damaging it. So what are you waitning for? Get to your word processor and print up a few pages on your system or make a copy of this article. Better yet, make a copy of the entire magazine and put it with your CoCo along with a page or two describing particulars to your machine. And if you know someone with a CoCo that doesn't subscribe -- I, Frank Swygert, owner of FARNA Systems, Publisher of "the world of 68' micros", hereby grant permission to copy and distribute this issue (Volume 5 Number 5) of "the world of 68' micros" free of cahrge. The only restrictions are that the magazine must be copied and distributed in full on paper only. No disk or electronic copies are allowed in full or part, and no articles may be posted to web pages as they appear in this magazine without written permission from FARNA Systems or its agents. No charges except for the exact copy charge can be made to the receiver, including shipping and handling charges. The intent is to allow this issue to be passed along freely. Make as many copies as you wish! This permission is for the stated issue only, and does not expire. THIS DOES NOT MEAN THIS ISSUE IS PUBLIC DOMAIN! FARNA Systems retains all copyrights.

This computer is a Tandy Color Computer 3, also known as a "CoCo3" or "CC3" for short. Previously (obviously) there was a Color Computer 1 and a Color Computer 2 (both known as "CoCo"s). The original CoCo1 was released in 1980 and produced through 1983. The CoCo2 was a slightly improved model with easier memory expansion and an easier to produce motherboard. It was produced through 1986. The long awaited, much improved CoCo3 finally came along in 1986. The last CoCo3 was made for the 1991 Christmas season. It did not appear in the 1992 Tandy catalogs. Eleven years of continuous production -- quite a longevity in the computer world!

The CoCo3 has 57 keys. Uppercase and lowercase are toggled by holding down the SHIFT and 0 keys. The F1, F2, and ALT keys are not used by most programs. The CTRL (control) key is used under ADOS (explained later), for fast entry of commands. The BREAK key is used to halt operation of some programs (most programs automatically end). The CoCo 1 and 2 only have 53 keys. They don't have a CTRL, ALT, F1, or F2 key. The CLEAR key is used as a CTRL key in some programs. See the CoCo manuals for more information (if you have them!).

On the rear of the CoCo3 are 10 connections and buttons. From left to right (as viewed from the rear) are the following: Reset, Audio, Video, Channel selector, RF out (for TV), cassette, Serial I/O (input/output), Right joystick, Left joystick, and Power. CoCo's 1 and 2 have eight connections -- they are missing the Audio and Video connectors, limiting them to a TV for viewing without a special adapter for a aomposite monitor (see "Composite Monitor Circuit" in this issue!). Always use care when plugging and unplugging cables! The computer should be OFF when anything is plugged in or removed.

The Reset button is used to end some programs. Also, after some machine language (ML) programs (ones that have a ".BIN" after their names on the disk) cause strange things to happen to CoCo when you exit them. Pressing reset should remove the program completely from memory. If not you'll have to turn the power off, count to ten, then turn it back on. The CoCo 3 has another way to totally clear memory -- hold ALT and CTRL down, then press reset. Let go of all the keys. A screen with three people on it will appear. These men helped design the CoCo 3. Finally, press reset again, and you will get the start-up message.

The Audio and Video ports are used to connect the CoCo3 to a composite monitor. It can also use an RGB monitor, the Tandy CM-8 or one of several other brands (the most common alternates are the Magnavox 8CM515 and 1CM135 or the Commodore 1084 and 1084S -- these require custom cables). The RGB port is located underneath the computer. Be careful when moving the CoCo if the RGB cable is still connected! It's fragile, so don't plug/unplug it often.

The Channel Selector switch is used to select either Channel 3 or Channel 4 when using a TV with the computer. The RF Out port is used to connect the CoCo with a television set. Its cable I usually leave connected. Some games written for a CoCo 1 or 2, not a CoCo 3, use colors not displayable on an RGB monitor. The programs use a method of creating multiple colors on the screen called "artifacting" that fools the TV display into creating more colors. These programming tricks don't work with RGB monitors, so they display black and white instead.

The Cassette port allows you to connect a cassette recorder to the CoCo. Most current CoCo users no longer use the cassette recorder, as the disk drives are much faster for storage/retrieval, plus files (programs) are easier to find on disks. The recorder is perfectly usable as a regular audio tape recorder. As you may guess, almost any mono cassette recorder can be used, even a micro recorder. It just needs a microphone and earphone jack. The third connector on the cable is for a remote control, so the computer can turn the cassette motor on and off. This isn't absolutely necessary, but if you're going to try using casettes

for a while, it is highly recommended. You may have to fool around with the playback volume for the computer to read cassettes. With a volume scale of 1-10, 8 is usually right. The cassette port is also used to connect the hi-res interface.

The Serial port is used to plug in the printer or modem cable. Only one of these can be attached at a time. However, I use a 2-position interface, which allows me to connect both peripherals to the CoCo 3 at the same time. The switch lets me choose which I want. Position A is for the printer, while B is for the modem. A cable with two connectors won't work. Some printer interfaces designed for the CoCo have a built-in pass through port.

Some games and other programs use the joysticks (or a mouse, which is identical electrically to the joystick -- sticks and mice can be used interchangeably). They plug into the left and right joystick ports. A very few CoCo 3 specific programs require the hi-res joystick interface. This plugs into the right port and the cassette port, with the joystick plugging into it. The hi-res interface cannot be used with anything not specifically programmed for it. It has to be removed for normal games and programs. Some hi-res interfaces have been modified with a switch to turn the hi-res circuits off so it doesn't have to be reomved for normal operation.

Use the Power button to turn the CoCo on and off. If the computer, monitor, disk drives, and modem (all described below) are connected to a switched surge protector or power strip, you may leave them on all the time, turning all on and off with the surge protector. Otherwise, always turn the computer on after other peripherals, and off before them.

The cartridge slot on the right side of the computer is used for program packs, such as "Gin Champion", or for the disk controller. Always make sure the system is off before inserting/removing anything from this port! You could seriously damage the computer.

The Multi-Pack Interface (or MPI, for short) also uses the cartridge port. This is an expansion device which turns the single cartridge slot into four. When using the MPI the disk drive controller must be in Slot 4. Any other packs can reside in any slot. Slots can be selected by either setting the switch and pressing reset, or by typing POKE 65407, followed by 0 (for Slot 1), 17 (for Slot 2), 34 (for Slot 3), or 51 (for Slot 4), as in POKE 65407,34 (the numbers POKEd in are hexadecimal codes, that's why they aren't just the slot numbers).

The printer made for the CoCo was a serial printer. These were the Tandy DMP 105, DMP 106, and DMP 130. Those models (and a few other, less common Tandy printers) have a round connector on them specifically for the CoCo. Any serial interface printer can be used but will require a special cable to be made.

The most common, least expensive printer is a "parallel" printer. These can be used on a CoCo with a serial-to-parallel interface (another interface!). They can run at a variety of speeds. I usually keep the interface set at 9600, which is the fastest a CoCo can connect to any serial device reliably. Such devices were common with CoCo users, but may be hard to find now. A common PC type can be used, but a special cable will have to be made to connect it to the CoCo serial port.

The CoCo, however, must also be set at the same speed as the printer or serial-to-parallel interface. This is accomplished with a POKE 150,x either entered from the keyboard, or put in a program (x=: 180 for 300 baud, 87 for 600, 41 for 1200, 18 for 2400, 4 for 4800, and 1 for 9600). You must enter this before trying to print or you'll just get garbage. Some programs that expect you to print will have settings for the printer speed that are saved with the program so you don't have to change it every time you want to print. Commands to print to the printer are issued like this: PRINT #-2, followed by quotes, and then your message, as PRINT #-2, "Lee Deuell".

The controller for the disk drives is inserted into the Color Computer's cartridge slot. Disk BASIC is available immediately upon starting (booting) up the computer when the disk controller is plugged in. Many people call this "RS-DOS", but that is a mis-nomer! It is not a true Disk Operating System (DOS), but simply enhancements to the BASIC programming language in the CoCo that allow it to read and write to floppy disks (thus the true name -- Disk BASIC).

The CoCo will control up to four single sided or thre double sided disk drives (the line for the fourth drive is used as the side select for double sided drives). They are connected to the disk controller by a large ribbon cable. Older cables are flat, the newest ones round. The round cable is actually a flat cable rolled into a casing. The flat cable is fine! Cable and connectors are available at computer stores and Radio Shack. Old cables will have "teeth" pulled from the cable to select the drive number. Later ones use a set of jumpers on the drives. The old type should all be worn out by now!

The standard CoCo drive is a 35 track single sided drive. This is programmed into Disk BASIC. Any IBM type 360K or 720K drive can be used. Disk BASIC will only use one side of the drive and 35 tracks (about 156K) unless it has been modified or OS-9 is being used. Those are options you can consider after learning more about the computer! Drives larger than 720K cannot be used with the CoCo controller.

Drives are numbered 0, 1, 2, and 3. The jumpers on the drive itself may be labled DS0, DS 1, etc. (for Drive Select x), or DSA, DSB, etc. Some companies start numbering with 0, some with 1 (0-3 or 1-4). The jumpers have to be set on the drives for the number you want the drive to be. IBM drives are usually all set as the second drive (1, 2, or B). A cable with a twist is then used to make one the first drive (end of cable) and the other the second drive (middle connector). You can leave the drives set this way and use a standard IBM type twisted cable, but you'll only be able to connect two drives (which is usually all you'd want anyway).

Always make sure there are no disks in the drives when you turn the computer on or off. The controller sometimes sends a stray signal through the drive head when powered on or off which could cause damage to the data on the disk. Be sure the handle on a drive is closed before accessing, and that the disk is inserted all the way before using. Some drives don't have handles, they use a "door"

or an eject button instead. If you accidentally try to access a drive that doesn't have a disk in it, press reset. It's not good for the drives to run much without a disk in them. Press reset as quickly as you can!

The modem connects the computer to the telephone line, so I can access on-line bulletin boards and information services such as Delphi. I use the "terminal" program "Ultimaterm" to connect to Delphi or to local bulletin board systems (BBS). Unfortunately there are few BBS's left since the advent of the Internet. Local computer clubs should still have listings of any local boards. It is doubtful there will be any CoCo specific content on local boards, but at least you can communicate with other general computer enthusiasts. Delphi is a large communications company. They are one of the few left with a text interface that the CoCo can use. To find out more, call them at 1-800. Prices are very reasonable and there is still a CoCo area with conversation and software freely available.

The CoCo cannot directly link with the Internet. The software isn't available and isn't likely to be written. There is a way to connect to the internet though. Electronic mail (e-mail) can be sent to (and received from) internet users through Delphi. That is the easiest way. The other way is through a "shell" accont from a local Internet Service Provider (ISP). This is a text interface into the providers system. Very few providers offer shell accounts, but it won't hurt to ask. What you do is log into the providers system with a terminal program then run an e-mail program on the providers system. This is basically how Delphi works.

The "World Wide Web" is the graphical portion of the Internet. The CoCo doesn't have the software or graphics format to access the graphics, which are primarily PC type s. There are text only "browsers" (the program used to access www "home pages") though. The most common one is called "Lynx". This is run on the host computer (Delphi or a shell account's host), not on the CoCo. Once you log onto the host, you run Lynx then access "the web". Most pages require graphics capability to get much from them, but there are still many that can be made out without graphics. If the graphic files have a ".gif" extension, you can download them and use a GIF viewer to look at some of them later. The CoCo only displays 64 colors, so 256 color GIFs may not look very good. Resolution is also limited, but most web pages use low resolution graphics.

When I want to run the computer, the first thing I do is uncover the monitor, keyboard, and printer (if I will be using it). The switch on the surge protector turns everything on. If you don't have a surge protector or switched power strip, turn the compuuter on last. Generally, the rule is turn everything on with the disk drive then the computer being the last two turned on. Next I type POKE 65345,2: DLOAD. This brings up the A-DOS screen, which is what I usually use. It asks for the date. You can enter the date or just press ENTER.

Next, put a disk in the drive. If it's in Drive 1, type DRIVE 1. This tells the computer to default to drive 1 unless you tell it to use another drive. If you type in "DIR" instead of "DIR1" the computer will assume drive 1 since you've already told it that's what you want the defualt drive to be. It's not necessary to enter DRIVE 0 if you are using it. The CoCo automatically defaults to drive 0. Some programs are hard coded to use drive 0. After running some of these, the computer will default to drive 0 even if you did type in "DRIVE 1".

Editor: ADOS is the most common enhancement for the CoCo. The following will tell you why -- it makes the CoCo easier to use. It is still available from FARNA Systems and is supported by the author, Art Flexser.

If you are using ADOS, hold CTRL and press @ to get the menu for that disk. Use the up and down arrow keys to highlight the program you want to run. Then press ENTER to run it. Until you get used to all of the programs, it's best to hold the SHIFT key down as you press ENTER. This is in case the program was written for the 32-column screen.

Without ADOS (or with if you prefer) use the commands RUN"filename" and LOAD"filename" to run programs. Most DECB commands also work under ADOS. ML programs can be executed with either the menu screen, or by entering RUNM "filename" or LOADM"filename" followed by EXEC. And you're off and running!!!

If a program doesn't work with ADOS, first try entering DISABLE:DLOAD and running it again. If it still doesn't run, type SLOW:POKE 65345,0:DLOAD to go to the DECB screen. Then RUN"filename" again. A few programs aren't compatible with A-DOS, and if you are using a CoCo 3, some of the older CoCo 1 and 2 programs won't run with or without ADOS

Here are the preprogrammed keys under A-DOS. Hold CTRL, then press:

A=SCREEN	B=CHR$(	C=CIRCLE(
D=DATA	E=EXEC 44539	F=FOR
G=GOTO	H=GOSUB	I=INPUT
J=	K=LEFT$(	L=LOAD"
M=MID$(	N=NEXT	O=OPEN
P=PLAY"	Q=TAB(	R=RIGHT$(
S=SOUND	T=THEN	U=POKE
V=PEEK(	W=WIDTH	X=RETURN
Y=CLOSE	Z=CAT	
1=LINE(	2=DRAW"	3=PAINT(
4=DATE	5=SCAN"	6=EXEC
7=CONFIG	8=LIST	9=SAVE"
0=RUN	:=FAST	-=DIR
@=MENU	;=SLOW	,=DRIVE
/=COLUMNS	UP ARROW=PRINT #-2,	
BREAK (then ENTER)=COLD START		

As you can see, these are "shortcut" keys to enter most BASIC commands. When writing a program they cut time dramatically. The command is saved, not the ADOS keystrokes, so programs written using shortcut keys will run on other CoCos without ADOS.

The CoCo disk controller is capable of using 4 "physical" drives. If all drives are single sided they are connected with the jumpers set as four individual drives. The drive select jumpers are usually located close the the drive cable connector. They are labeled DS0, DS1, etc. Some manufacturers use the labels DSA, DSB, etc., and some simply use a number or letter. Most start letters with 0 (0-3), but some start with 1 (1-4). A few of the newer drives will only have jumpers for two drive numbers/letters since that is

how most IBM/PC compatibles are setup. And some will have no jumpers, or the jumper to drive 1 (or 2 or B) soldered closed. This is because IBM chose to set both the supported floppy drives as the second drive then use a twist in the cable to select the first and second drives. The CoCo normally uses an untwisted ribbon cable for the drives (even the round cable on later drives is a rolled up ribbon cable in a plastic sheath), but if you only intend to use two drives an IBM style twisted cable can be used -- just remember to set both drives as the second one! The drive on the end of the cable will be the first drive.

The CoCo controller will support any drive up to 720K capacity no matter the size. The common sizes now are 5.25" and 3.5". The CoCo WILL NOT use the newest 1.4 megabyte HD drives. Most spin at a faster speed than the older types and confuse the controller. A few have jumpers to force them to be 720K only drives, but your best bet is to get a used, genuine 720K 3.5" drive.

The 5.25" 360K drive works fine with the CoCo even though it is a double sided drive. Single and double sided drives are electrically identical, the controller actually determines drive capacity and if both sides are used. With standard DECB, only 35 tracks on one side of a drive will be used, about 157K. Using ADOS, all 40 tracks can be used on each side (180K). The CoCo controller uses the side select line as the select line for the fourth drive, so if you want to use both sides of double sided drives only three drives can be used. DECB and ADOS only allow four drive numbers, so only two double sided drives (or any combination of single and double sided up to four drive numbers) can be used. The CoCo uses the "back" side of a drive as anther drive number, so one single sided drive becomes drive 0 and drive 1, not one large drive. The common practice is to use the "back" of drive 0 as drive 2 and the "back" of drive 1 as drive 3. This allows drive 0 to 1 backups easily (the CoCo defaults to copying from 0 to 1) and works like single sided drive systems. The next paragraph will offer some insight to the method of designating drives even if you don't have ADOS.

The Config command is used under ADOS 3 (only!) to name/use the various disk drives in a system. Extended ADOS 3 adds two "RAM" drives. The "physical" drives may include both sides of the disks in the drives. The two RAM drives are "invisible", created from some of the CoCo's memory. You can have a small RAM drive with 128K, but 512K is recommended. This is a good way to use the extra memory if you are programming in BASIC or running BASIC programs.

ADOS allows up to four drives to operate at one time. Please read the A-DOS manual for more information on using the CONFIG command. I have the CONFIG command set up as follows: CONFIG A gives me Drive 0, Drive 1, RAM Drive 0, and the back of Drive 1. CONFIG B gives me Drive 1, RAM Drive 0, the back of Drive 1, and Drive 0. CONFIG C gives me RAM Drive 0, Drive 1, the back of Drive 1, and Drive 0. CONFIG D gives me Drive 0, Drive 1, the back of Drive 0, and the back of Drive 1. You can also type CONFIG followed by whatever combination you would like, such as CONFIG 0,B0,R0,1 which would give you Drive 0, the back of Drive 0, RAM Drive 0, and Drive 1.

To run programs under DECB type RUN"filename" to load and automatically run them, or type LOAD"filename" followed by RUN. Use LOADM"filename" followed by EXEC to execute ML files

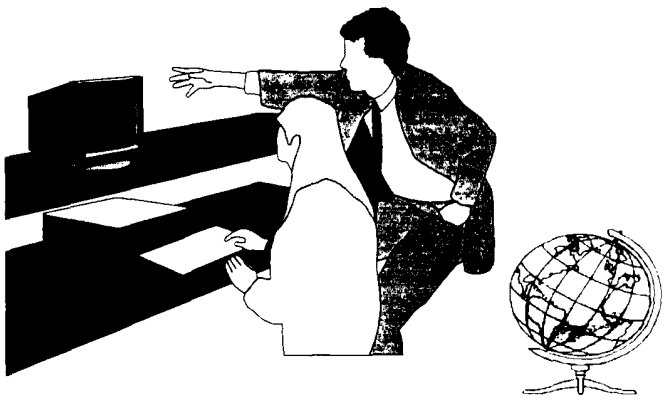
Most programs include a way for them to automatically end. One way to stop a program that is running is to press the BREAK key. As a last resort you can press the Reset key. Be careful when using Reset, however, as it can erase a program that is in memory. If a program asks you on the screen if you wish to quit or end the program, just press the appropriate key (usually "Y" for yes, "N" for no). If you have ADOS setup for lowercase, or changed to lowercase while running a program, make sure the CAPs are on by holding SHIFT and pressing the 0 key.

When you are done using the computer first make sure there are no diskettes in the drives. The CoCo and its peripherals can be turned off using a surge protector or power strip.

There's one aspect of the CoCo that I have, but which I don't use very much. That is OS-9. OS-9 is an operating system for the Color Computer, as opposed to a program, just as MS-DOS is an operating systemfor IBM and compatible computers. OS-9 turns the CoCo into a totally different machine. If interested in OS-9, get the books and get in touch with other users. "68' micros" runs articles covering OS-9, and a beginners article is in this issue.

One thing I will point out about OS-9 -- you can use the full capacity of 360K and 720K drives as a single drive. This is why the information on using three or four drives is handy. If a double sided drive is used for OS-9, only three drives can be used. It is a very good idea to make sure drive 0 is a double sided capable drive if using OS-9. If you still use a single sided drive, use it as a higher number. his way you can boot OS-9 from the larger drive. The more capacity for OS-9 the better! The recommended setup for DECB and OS-9 use is two double sided 360K drives and one 720K 3.5" drive as the last drive. The 3.5" won't be useable under DECB, but that is no problem as the standard drive size for DECB is 5.25". Very few users have 3.5" drives in use under DECB.

I think this is enough information to get you started using the Color Computer. I don't want to tell you everything! If you have any questions about any part of the computer system, try to get in touch with other users and vendors. This magazine is a good place to start, a subscription will pay for itself many times over. Good luck!



Chicago CoCoFest 98!

THE GLENSIDE COLOR COMPUTER CLUB OF ILLINOIS PRESENTS
THE SEVENTH ANNUAL "LAST" CHICAGO COCOFEST
April 18th & 19th, 1998 (Sat. 10am-5pm; Sun. 10am-3:30pm)

Elgin Holiday Inn

(A Holidome Indoor Recreation Center)
345 W. River Road
Right off intersection of I-90 & IL-31, Same location as past years!
Overnight room rate: $65.00 (plus 10% tax)

Call 1-847-695-5000 for reservations. Be sure to ask for the "Glenside" or "CoCoFEST!" rate. There is a limited number of rooms set aside for the CoCoFest. *These rooms will be released on March 31. They will not be available at the 'Fest rate after that date,* ***so make your reservations early!***

General Admission: $10.00, whole show (Children 10 and under are free)

For further information, general or exhibitor, contact:

Tony Podraza, VP, Spcl Evnts, GCCCI
847-428-3576, VOICE
847-428-0436, BBS
Tonypodraza@juno.com

Mike Knudsen, President, GCCCI
630-665-1394, VOICE
Mknudsen@lucent.com

Brian Schubring, Ast., Fest Coordinator, GCCCI
E-MAil theschu2@juno.com OR theschu3@aol.com

CoCoFEAST!

That's right a CoCo FAMILY DINNER at the HoliDay Inn. Why? So You don't have to drive 'here or there' or try to decide which group you want to spend time with. We can be all together to enjoy the food, and best of all, each others company. There my be a Keynote speaker present. This is planned for Saturday Night about 6:00pm. We need is a MINIMUM of 50 people to reserve a dining room. The tickets will only be available in advance, AT THIS TIME! People to conntact are listed below. The cost will be only $15 U.S. PER Person. We will be able to take paid reservations only up to March 28th, 1998. Please contact one of us for further details.

NOTE: THE CLUB IS NOT MAKING ANY MONEY FOR THE DINNER, NOR PLANS TO. THIS FUNCTION IS ONLY TO PROMOTE A COCO FAMILY GATHERING AT ONE GREAT LOCATION. . . THE COCOFEST!

If by the specified date we are lacking the number of attendees required, the dinner will be scrapped and a refund will be issued at the Fest.

Afterwards there may be a Musical Monk'O Rama Jam-Session like we had last year with Brother Jeramy, Allen Huffman, and anyone else who wants to bring and instrument and join in!

See Ya'll there in '98!!!

ADDITIONAL COCOFEST VENDOR INFORMATION

The Glenside Color Computer Club of Illinois is a not-for-profit computer club established to assist its members in the learning and better understanding of Tandy's Color Computer. In the pursuit of that goal, we have been priveleged to host, under the auspices of Falsoft Publishing, five Chicago Rainbowfests and one CoCoFEST! sponsored by CoCoPro!. Last year, we organized AND sponsored the Sixth Annual "Last" Chicago CoCoFEST, which was in the Chicagoland area.

This year, Glenside will again sponsor the show as noted above. We would like to continue our tradition of giving door prizes, but we need your help to do so. All vendors are asked to donate items to be distributed through a drawing of admission pass stubs. If you wish your donation to be part of the Grand Prize given on Sunday, please state so. We are also open to donations for the auction held during the show. Proceeds go to Glenside to help offset costs for this and future CoCoFEST's.

Donations MUST be received by April 1, 1998, in order that all prizes and vendors can be properly credited. You will receive a receipt for your donations

EXHIBITOR INFORMATION:

Tables: $35 each (excludes vendor passes)

Full booth exhibits shall be no less than 8'x8'. including one 6' table (draped and skirted), booth sign bearing Exhibitor's company name and booth number, two chairs, and one electrical outlet. Other exhibition needs can be provided at extra cost. Booths may not be shared without the knowledge and written consent of the FEST! Coordinator (inquire!).

Vendor Passes: $5.00 each (max of 2 passes per table)

REGISTRATION MUST TAKE PLACE WITHIN THE MONTHS OF JANUARY AND FEBRUARY, 1998! The FEST! Coordinator MUST have your deposit of $25 by February 28, 1998. The remainder is due by April 1, 1998. Payments received AFTER April 1 will be subject to a 20% LATE FEE based on the outstanding balance. Deposits are NON-REFUNDABLE after April 1, 1998. Send a check for the deposit made payable to "Glenside Color Computer Club" to:

CoCoFest! Glenside Color Computer Club
c/o Tony Podraza, FEST! Coordinator
119 Adobe Circle
Carpentersville, IL 60110-1101

A registration packet will be returned to you. You may, of course, write for the packet first, but the deposit MUST be received by the FEST! Coordinator no later than February 28,1998.

Liabilities and Restrictions:

Sponsor reserves the right to change exhibit hours or cancel the exhibition of its own volition, in which case all deposits and prepayments will be refunded. Should the exhibit be cancelled due to circumstances beyond the exhibitors control, deposits and payments may not be refunded. Neither the sponsors nor its representatives shall be liable for any injuries, loss, or damages in any form. Exhibitor is at all times responsible for its own goods and materials regardless of location.

FARNA Systems

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

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

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

BOOKS:

Mastering OS-9 - $30.00
Completely steps one through learning all aspects of OS-9 on the Color Computer. Easy to follow instructions and tutorials. With a disk full of added utilities and software!

Tandy's Little Wonder - $25.00
History, tech info, hacks, schematics, repairs,... almost EVERYTHING available for the Color Computer! A MUST HAVE for ALL CoCo aficionados, both new and old!!! This is an invaluable resource for those trying to keep the CoCo alive or get back into using it.

Quick Reference Guides
Handy little books contain the most referenced info in easy to find format. Size makes them unobtrusive on your desk. Command syntax, error codes, system calls, etc.
CoCo OS-9 Level II : $5.00
OS-9/68000 : $7.00

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

CHECK OUT THE NEW LOW PRICES ON NITRO PRODUCTS!
See editorial in this issue for details

SOFTWARE:

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

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

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

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

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

TuneUp : $10.00
Don't have a 6309? You can still take advantage of Nitro software technology! Many OS-9 Level II modules rewritten for improved speed with the stock 6809!

Thexder OS-9
Shanghai OS-9 : $10.00 each
Transfer your ROM Pack game code to an OS-9 disk! Requires ownership of original ROM pack.

Rusty : $10.00
Launch DECB programs from OS-9! Load DECB programs from OS-9 hard drive!

NitrOS-9:

Nitro speeds up OS-9 from 20-50% depending on the system calls used. This is accomplished by completely rewriting OS-9 to use all the added features of the Hitachi 6309 processor. Many routines were streamlined on top of the added functions! The fastest thing for the CoCo3! Easy install script! 6309 required.
Level 3 adds even more versatility to Nitro! RBF and SCF file managers are given separate blocks of memory then switched in and out as needed. Adds 16K to system RAM... great for adding many devices!
NitrOS-9 V.2.0: $10.00
NitrOS-9 Level 3: $10.00

The AT306 OS-9 Single Board Computer

AT306 Motherboard Specs:

16 bit PC/AT I/O Bus (three slots)
MC68306 CPU at 16.67MHz
Four 30 Pin SIMM Sockets
IDE Hard Drive Interface
Floppy Drive Interface (180K-2.88M)
Two 16 byte Fast Serial Ports (up to 115K baud)
Two "Terminal" Serial Ports (no modem)
Bidirectional Parallel Port
Real-time clock
PC/AT Keyboard Controller (five pin DIN)

Included Software Package:

"Personal" OS-9/68000 Vr 3.0
(Industrial with RBF)
MGR Graphical Windowing Environment
with full documentation
Drivers for Tseng W32i
and Trident 8900 VGA cards
Drivers for Future Domain 1680
and Adaptec AAH15xx SCSI cards
Many PD and customized utilities and tools

The AT306 is a fully integrated single board computer. It is designed to use standard PC/AT type components. Sized the same as a "Baby AT" board (approximately 8" square). Compact and inexpensive enough to be used as an embedded controller! Use with a terminal (or terminal emulation software on another computer) or with a video card as a console system. Basic OS-9 drivers are in ROM, making the system easy to get started with.

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

Call for a quote on turn-key systems and quantity pricing.
Warranty is 90 days for labor & setup, components limited to manufacturers warranty.

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

operating system nine

Jim LaLone & Rick Ulland

Beginners guide to OS-9

Introduction

This is a limited peek into a powerful, complex system. It is not intended as a "how to" manual, rather to help you "get the feel". Some specifics are given and every attempt has been made to be accurate in these cases. To make things easier, I have divided this guide into three sections. Section I and II was written by myself (Jim LaLone), section III by Rick Ulland. You may want to read them one at a time then take a little time to experiment or think on the presented ideas before going on to the next section.

Section I:
Getting the OS-9 Attitude

Some limitations...

From the very outset I will make it clear that, as it comes, OS9 for CoCo is a rather crippled system because of its packaging and lack of support from the producer. There are several things you can do, as soon as you are able, that will make it vastly superior to what it is in its original form.

First, have Level 2 on a CoCo3. There is no comparison between a 32 or 40 column screen and an 80 column screen, but for strange reasons OS9 arrives in a low resolution package and on single sided, 35 track disks. Personally, I wouldn't even consider using it that way. It is like having a powerful new car with special full time brakes installed and all but one window painted black.

Editor: If you have a CoCo 1 or 2, you are limited to Level I. Other than screen size, the most noticeable difference between Level I and Level II is memory support. Level I only supports 64K, Level II up to 2MB. Still, Level I is a good learning tool, but memory and screen limitations make it less useable for applications

80 Columns and Enhancements

So you have an 80 column computer -- how do you get OS9 to use it? There is an article by RICKULAND entitled "The First Step" included in this collection. It tells you how to switch to 80 columns, double sided 40 track disks and high speed drive stepping. It is a fairly long tedious procedure, but when it's done, you'll be able to see what you're doing and you'll have more than double the disk space.

Another real aid in using OS9 is in the System Modules database on Delphi and can be found on the RTSI site. It is written by OS9 programmer Kevin Darling and is called SCF EDITOR PLUS - LEVEL TWO. Again, it is a task and a half to install, but is WELL worth it. It is a command line editor that makes life in the OS9 world of long command lines greatly easier, especially if you're not an expert typist.

There are many other improvements and additions to OS9 that you will get as you go along, but these are tops in my opinion. Next in line would be the new shell, SHELL+, with several improvements. As of this writing the highest version is v2.1. The sooner you can arrive at an updated system the better off you'll be.

The easiest way to get the most used enhancements is to purchase "Patch OS-9" or "Tune-Up" from FARNA Systems. "Patch OS-9" has all the patches individually with complete documentation on disk. One can pick and choose which updates to install. A lot of utilities are also included. "Tune-Up" is an integrated enhancement that has all the patches pre-installed. You cna't choose what you want or don't want. The advantages are that some of the patches and modules have been rewritten to be a little faster than the originals and it takes a little less time to install. Documentation for all the enhancements is not included -- it is intended for an experienced OS-9 user. "Patch OS-9" was written specifically for novice users. It has an auto install feature for CoCos with 512K and two double sided drives. Installation instructions are good for both programs.

Changing Your Attitude...

Coming to OS9 from CoCo Disk Extended Color BASIC (DECB) involves something besides the software and dreams of owning a hard drive - if you don't already have one. No doubt you've heard about the power and versatility of OS9. It is, indeed, several steps ahead of DECB in many respects. But the change in environments can be almost overwhelming if you don't grasp a few essentials - a few of the differences between DECB and OS9. You can't maintain a DECB attitude while learning OS9.

DECB is a BASIC language with some of the power of a DOS (disk operating system) thrown in. OS9 is an operating system with some of the power of a language thrown in.This much is mundane. But getting into the actual differences is both interesting and essential.

A simple concept that sheds light on the move to OS9 can be illustrated this way. If you had all the money you wanted, but all audio/video systems were alike, it would take you minutes to buy a complete system. But with all the hundreds or thousands of TVs, CDs, amps, tuners, speakers, stereos, tape players, recorders, multi media systems, etc., in the equation, it is a major job deciding on an intelligent system purchase.

The more options an operating system has, the more decisions you must make, the more you must remember, and the more occasions you have to make mistakes. What might involve three factors in a simple system might involve ten in a more powerful system. The three would give you nine paths to choose from. The ten would give you one hundred choices - for starters.

The importance of this point can't be over estimated. And it directly relates to one of the things that causes the most trouble and consternation to new OS9 users - multiple directory disk organization. That will be our starting point and our prime consideration.

Directory Organization

For all practical purposes an OS9 system might as well have six or eight (or more) different drives. You can imagine how much you could do by organizing them all in creative order! But you can also imagine the headaches of remembering which drive had what on it if they weren't organized. This dilemma would ideally be solved

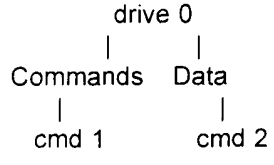
by arranging them in a logical heirarchical order.

First you would divide everything up into files that do work and files that are worked on - executable files and data files. If you have a great many executable files you might have to divide them into two or more sets. The data files must be divided into many more sets and subsets. There are data files like messages, there are tables, lists, subroutines, logs, graphics, music, controls, etc. Some files even defy catagorizing!

Which drives would get which files? Remember, we're talking about much power and large numbers of files! And, to make it interesting most of the commands you would use like "deldir" aren't usually in memory, but on disk! To get all the power, THERE ARE SO MANY commands they'd choke the computer if you put them all in memory at the same time, even with a 2MB system! And you'd go crazy looking for the right one on the disk - except for the organization (Pay close attention to the use of words like "might" and "could". This is not a "how to" article. I will go out of my way to avoid setting down "how to" rules while hypothesizing). A level of organization is USUALLY, but not always (another variable) indicated by a "/" mark. Let's look at some levels.

Levels of Organization

Within the disk in a drive there might be a section of commands. Listed under commands (besides scores of things we usually think of as commands) might be WP, a word processor, and SS a spread sheet . Let's see what this starts to look like. Then I'll back up and make things better (and relieve those who just KNOW I'm doing this wrong).



You start with a "/" for the top level.
/

You add a drive number.
/d0

You add another "/" for a level within d0. /d0/

You add the commands identifier.
/d0/cmds

You add another level indicator.
/d0/cmds/

Finally the name of the word processor. /d0/cmds/wp

In BASIC you enter RUN"WP". In OS9 you enter /d0/cmds/wp, a three level command (Nothing magic about the "three". There can be more levels).

Let's do a tiny bit of computing -- deleting WP from the disk. That will add another dimension to remember. If there were only one level you could enter "del wp." But the computer wants something like this:

del /d0/cmds/wp

The real beginners OS9 system usually starts with two drives at the top level of organization. So let's put a backup copy of WP on the other drive - drive 1. Let's see how complicated this can turn out:

copy /d0/cmds/wp /d1/cmds/wp

Already you can see something that is similar to DECB - spaces separate items within a statement. Something different is that both the word "copy" and the word "WP" are executable and considered commands on the disk. Some people separate these kinds of files making yet another thing to remember.

Making it Easier -- CHD and CHX

Now remember that you have tons of stuff on your disk - commands and data. The authors of OS9 came up with a way to cut down on a lot of the typing you need to do to slice through all the levels of organization. You can set the system to know what drive and/or level you're working with and whether it is executable or data. Voila, the commands CHD (data) and CHX (executable).

Let's switch to computer talk for these levels of organization. What they are called is directories and sub-directories. Like a general index for an encyclopedia then a different index for each book.

If you enter "chx /d0/cmds" you set the executable commands pointer to that drive and directory. Then you can leave off that part from a typed command! Since WP is in the cmds directory, to get it you just enter "wp"! At this point you could delete it by entering "del wp." Simplifies things, doesn't it?

Let's do some more computerese. After the command itself, a line like we've been using is called a pathlist. It indicates the path down through the levels of directories to a given item you're after. Remember - the effective name [pathlist] of a file includes the names of the drives and directories above the filename itself.

These helpful commands (CHD and CHX) are perhaps even more time saving if you're dealing with data. For example, if you have to decompress several files that are located in the ARC directory, you could enter "chd / d1/arc". Then you could call all the files without the full command line and all the files you decompress (dearchive) would be stored in that same directory. When you set chx to a directory it is then called the "execution" directory and the system will look there for any command you use that doesn't have a pathlist preceding it.

CHD and CHX are a lot like connecting to two single directory drives, one for data and one for commands. That simplifies things doesn't it? But are you thinking what I'm thinking? Chd and chx are also two more things to remember!

So what happens if you forget? You get an error!! If you set "chx /d1/cmds" and then call for a command that is on drive 0 you will get an "error 216" - pathname not found. That is like an NE error in DECB. But you can't just blame things on CHD or CHX, because the system will have to look in SOME specific place for what you command. And if you ask the wrong thing, you get the wrong thing.

Errors!

Note that the error message uses "pathname", not "pathlist". Consider a pathname a segment (between slashes) of a pathlist. Let's look closer at errors. Forwarned is indeed forarmed.

Usually errors are the bane of the OS9 beginners existance. With so many things to remember it is easy to make multiple mistakes. And as with other systems the actual error number doesn't always tell you what's wrong - even if you look up the number.

But there is something you can keep

in mind that will ease your recovery from such things with the least frustration and time. Knowing it in advance will at least allow you to analyze problems in an intelligent way. Remember that the file name is called "pathlist" because it includes the path of drives or directories through which the file is accessed.

Any part of the pathlist can give you the error - not just the file name at the end of it.

Let me describe just one error problem. If the command you start a line with is not in the directory of executable commands that you indicate, you can get a 216 error there. If the drive you name, has the wrong disk in it, you can get an error from that part of the pathlist. If the directory you name is not on the indicated disk, that will give the same error, 216 - pathname not found. And, of course, if the file you want turns out not to be in the indicated directory, same thing. Of course, misspelling can effect any of these things. At least you now know to look in different places for trouble.

All this really involves a state of mind. To get at all that power you have to think differently (and more) with OS9 than you did in DECB. It's like moving into a house with 10 times as many rooms and 10 times the stuff. At first you'll have trouble remembering where you put everything.

You will find, as you progress, that there are alternative ways to do a given task. The third section will involve a little less attitude and a little more technique. There I will go a little deeper into some points I have made so far.

Section II:

Beyond the System

The Philosophy of OS-9

A sort of philosophy is also involved when you begin to deal with OS9 software. If you used only smoothly functioning, shrink wrapped commercial software, you probably wouldn't be reading this. You've probably been on Delphi or the Internet to try downloaded software, software disks that came with a used system, or to ask for help with something that is not so smooth running.

There are beginners who frequent Delphi, the CoCo List, and other Internet sources. Often they're as full of questions as you are, but they have answers. At the other end of the spectrum there are some very expert and brilliant programmers who know "everything there is to know" about OS9. Too often, you will find yourself simply talking a different language than these "tech types."

It isn't that they don't want to help. They're helping each other every day! If you want to know something about the exact syntax of an obscure part of a brand new update to the latest XYZ language enhancement, you're in luck. But if you want to know how to get the menu on yesterdays spreadsheet it might take longer.

Some very bright programmers dash off experimental programs for fun. Sometimes they like what they end up with and post it on Delphi or an ftp site. Maybe they wrote two pages of documentations for a complicated communications program. Maybe they didn't finish it. Maybe they forgot they even wrote it! Grabbing the first thing in sight may or may not be a good idea.

If you're looking to OS9 to be the cure for the common cold, the end all, be all, you need to rethink things. It is a tough thing to learn. But some of those brilliant programmers I mentioned have worked long and hard at getting rid of bugs in the original package and adding yet more power to it.

Making the Most of Space

I guess the irony is that this system sits in such a tiny box wishing it had some place to go. Users require more and more function. That calls for more and bigger software, and that calls for more memory and/or data storage space. When that space is limited, you cut something. What should get cut?

A very appealing aspect of OS9 is the ability to snap from one program to another with the push of a button. But with space and CPU speed limited, how many programs of what size, power and function can you squeeze into a little CoCo? Working Delphi is nice. You run a terminal program. Maybe you have an editor to generate messages which are stored. You download a file to disk or ramdisk. Maybe you print out documentation in the background.Fun!

But what are you downloading? A spreadsheet that will hold all your small business bookkeeping for a year? Will you run that at the same time you have a memory gobbling graphics program in place? And the graphics user interface you use? And the big game you didn't finish last time you played?

Multi-Tasking

Then, perhaps more importantly, there is the 6809 CPU a terrific little chip that could blow the doors off everything when it first came out (and all later eight bit processors as well). DECB users who've been around a while remember when everyone was scrambling to get things running at double speed - (still less than 2MHz, actually 1.78MHz). If one program can need that, what about several, running at once? Some programs take little CPU time. Some take a lot more, like when there is much disk I\O. Special no-halt disk controllers can help a lot, but that is only one of the angles.

How successful multi-tasking will be for you will depend on what you'll be running. Don't plan on having three action games running on three different terminals.

With applications growing to accomodate users demands for computing power, more memory and speed is needed. If you're a purist of the 68xx genre, 68xxx computers (MM/1 and AT306) may be interesting. They have many times more memory and speed than the CoCo. You'll find programs right on the RTSI site for these computers and some discussion on the OS-9 and even CoCo mailing lists and USENET groups.

Look around...

Take a look at things. If you're a budding programmer, learning OS9 can be very educational. If you're a hobbiest you can do a lot of experimenting with OS9 software and utilities. You can even run some of those very good applications programs. Or power a rock concert via Ultimuse and MIDI!

A little Body For the Attitude

So far, I've given you just a peek at how life can be in the OS9 world. Some things I've described may be very unlike what you ultimately find to

be your specific experience. But I have been more interested in the general idea of things. Now I'll touch on a few points that will probably effect everyday computing more. Still, this isn't an OS9 "how to", just a whiff of its perfume.

So Many Levels!

Why is it so important to have so many levels of directories? Firstly, as I mentioned, there is a lot of stuff to organize. With people writing constantly there are new "tools" coming out almost every month (I have 16 clocks on file and that isn't all of them). By "tools" I mean utilities to do jobs, etc. Earlier I used "del" in an illustration. That is a very simple tool. "List" is a fairly simple tool. "Free" is like free in DECB. "Mfree" is like "MEM" (there ARE differences between such similar DECB and OS9 commands but the general idea is the same). Improved versions of various commands are written, often somewhat larger. Most veteran OS9 users have an entirely new shell - SHELL +.

Other commands are replaced if the user prefers something different (and if an alternative is available). But you really can't just toss out the original commands. There WILL be a time when you need at least some of them if you stick with OS9 long! So you keep copies of them. Probably on a separate disk. But if you happen to want to use both versions you might want them on one disk. You can't do that with DECB if you use the same name, but it is easy with OS9, because you can put them in different directories.

You might want to put them on the same level in differently named directories. You could have, for example, CMDS1 and CMDS2. Then it would be possible to have one version of (perhaps) ATTR in one and another version in the other - (/d0/cmds1/attr or /d0/cmds2/attr). How much of this you can do might depend on your own memory.

Or you might want to put them on different levels. You could actually have a CMDS directory listed under the regular CMDS directory (/d0/cmds/cmds/attr), although it would probably not be a very good idea. A "dir" could confuse you if you didn't remember which level you were looking at.

Another reason for various directories is the wide variety of data you can (sometimes must) have on disk. Frequently you will see directories named SYS, DOC, MODULES or ARC. SYS might have data essential to the function of an assembler. It might have "help" or "error" messages. "DOC" would probably have documentation for programs or commands. "MODULES" would likely have descriptions of the parameters of your various I/O devices such as your printer or RS-232 pak (40 isn't unusual). ARC would probably have archived (compressed) copies of programs, data, or other files.

As mentioned, directories CAN be "stacked" many levels deep. One communications program stores dialing information like this:

/d0/sys/dial/filename

A MultiVue screen icon might be named like this:

/d0/cmds/icon/icon.app.

I saw an example in which Basic09 commands were on the same disk with others. Under the regular CMDS directory was BASIC09, and under that was another CMDS. To copy it you could end up with this line:

copy /d0/cmds/basic09/cmds/filename /d1/cmds/basic09/cmds/filename.

Note that the "standard" naming convention for OS9 is that directories are named in all caps (such as CMDS) with file/program names in lowercase of mixed case. There is no set rules, as OS9 isn't case sensitive (In most cases, OS9 accepts lowercase keyboard input for the system. Programs run under OS9 make their own decision on that), but names will display as types. Using this convention makes things easier to decipher later -- you automatically know that "CMDS" is a directory and "list" is a runnable program. Names can be up to 29 characters long, so they can be very descriptive, like:

"Business.letter.to.Microware"

This uses 28 characters, including periods. An underscore character can also be used as a separater:

"Business_letter_to_Microware"

Since mixed case is used, we know this is a file, not a directory. The last characters are usually an indicator as to the type of file or program the file uses, so a better name mighr be something like:

"BS_letter_to_Microware.wp"

which would indicate a word processing (wp) file.

If you see the reason for multiple directories, we can go on.

Unified I/O System

At some point you will probably come across the phrase "Unified I/O System". So far I've referred to pathlists only as long paths through the maze of directories to files on your disks. But paths can access other things.

Via pathlists you can list a text file to the screen or you can list it to your printer or to a screen in another window to be seen later! Via a pathlist you can read input from your RS-232 pak (and therefore whatever it is connected to). You can run a program in a window you don't see. These things can happen because of this "unified" system.

In section one I mentioned that most commands are usually kept on disk. When you need, for example, the command "format" it will probably be on disk. When you enter "format" the computer loads it into memory and executes it (but not by firing squad) then unloads it when it is done with it. But you can also load commands into memory manually, then unload them when you wish (OS9 computerese for "unload" is "unlink".)

Editing the Command Line

In section one I mentioned that misspelling something in a command line could cause an error such as 216 (pathname not found). Something that you can do in OS9 does make life easier when this and other things happen -- the command line buffer.

What you enter as a command line is saved in memory. You can then repeat it with the appropriate keypresses. As it comes, OS9 has an command repeat that is activated by CTRL-A. Before you press <ENTER> you can do some overstrike type editing.

With one very popular new version of this you use right arrow or shifted right arrow instead of CTRL-A. You can easily key over to the offending characters, change, insert or delete them and repeat the command. This is the "SCF EDITOR" I mentioned at the beginning of this article. This saves

MUCH time over typing in whole lines again, especially when dealing with long full pathlists. If you're not the best typist, it can be a life saver.

It also comes in handy if, for example, you have several similar files to copy. By changing only the actual filename part of the pathlist you could move several files without retyping much. On the other hand, when in doubt about where you are, you can usually enter a full pathlist to "feed the hungry computer" (sometimes, specific programs might not allow that).

The Startup File and Scripts

There are other things that can really frustrate the beginner. One thing is the time it takes to do certain tasks according to the manual. For example, at some point you will wish to rewrite your startup file. Beginners haul out the "build" command and start a new startup file from scratch as instructed by the manual. One tiny error and you're back to deleting, renaming, rewriting or what all, from the top down.

Startup is actually a text file called a script (this may be in the manual somewhere, but I've never found it. It is definitely not in the indices). Each command is on a separate line. You can a use a line editor such as the "Macro Text Editor" (EDIT -- discussed later) that comes with OS-9 Level II, SLED, or a word processor to create and load such a script as "startup" and add or subtract at will. Just loading an existing startup file in an editor (or listing it) and looking at it will take away much of the mystery.

Always press <ENTER> after each command as you would on the keyboard. Just as in regular word processing, this will start a new line for the next command.

If you wish to keep a copy of the original startup file you can rename it the way the manual suggests -- before starting on a new one. Then when you are finished with the new one you save it to disk as "startup" just like the original. Once you get comfortable with the system you will probably do this.

The "edit" utility that comes with OS9 can do this job. If you find it easy to use, by all means do so. But with a regular text editor you can see it all at once and use the arrow keys to edit.

Since a script is just a series of commands in a text file, you can write different scripts to do different things and call them just as you would any other command. This is a good way to set up a series of commands you use repeatedly. If a particular program calls for doing several things to prepare for it, you can probably do them in a script. You could put your preparatory commands in it, save it in the "cmds" directory as "pstart", for program start, and do the job by just entering "pstart" (assuming your chx was set to "cmds").

Scripts are really very easy to write. You'll see. But all is not rose petals here either. Sometimes, even when the script lines appear to be perfect, a line will just not work. Swapping lines sometimes makes a difference. Script lines do not give error messages. Shell+ will automatically get a script from the execution directory (set by chx). You can NOT load a script into memory like other commands unless you upgrade to Shell+ v2.0 or higher.

I have tried to say (perhaps warn), in a simple way, how complex OS9 can be (even the explanation has become somewhat complex). I have tried to offer some preparation for dealing with that by covering an area or two that has been especially problematic for the new OS9 user -- the manuals can be very frustrating sometimes. Perhaps the best preparation is to tell you that this is just the "tip of the iceberg."

The Firing Line

Since directories and the CHD and CHX commands are at the center of a lot of beginners problems here is still more to help make things clear.

When you type in a line like:

list /d1/modules/bootlist

the system looks for those names on the disk. It looks first for "list" in the execution directory then looks for a descriptor named "d0", a directory named "modules" and a file named "bootlist". It must have the numbers that these names represent, and it scours one or more disks looking for them in logical ways. But when you use CHD and CHX those numbers are stored for quick use.

If the "CMDS" directory is, for example, on line 1 to drive 1 and on sector 872, the CHX command stores that information. It is as if a big arrow were pointing to that sector. After that, any time the system "sees" a command it jumps to those numbers - that sector. That is why it cannot look for the "CMDS" directory by name.

Here are some typical command line problems:

Say you have put a system disk in drive 0 and issued a "CHX /d0/cmds" command. Then you enter:

copy /d0/sys/stdptrs /d1/sys/stdptrs

Things will probably be okay. But if you were in a hurry it may be:

cpy /d0/sys/stdptrs /d1/sys/strptrs

The system cannot find "cpy" and you get a 216 error.

If you put the disk in but forgot the CHX command, a perfect command line will probably give you a 215 error - bad pathname - because the system jumped to sector so-and-so and there was no "CMDS" directory there! See?? Right there is where a lot of frustration comes in (sometimes this may work, because if the command is in memory [where the system first looks] it will work anyway.)

"copy /d0/sys/stdptrs /d1/sys.stdptrs" isn't a hard mistake to make if you're not a good typist. This will cause the system to store a file in the directory last indicated by CHD and name it "sys.stdptrs."

"copy /d0/sys/strptrs /d0/sys/stdptrs" will probably give you a 218 error - file already exists.

"copy /d0/sys/stdptrs /d1/sys/stdprts" will not give you an immediate error. But when something looks for "stdptrs" and there is nothing but "strprts" it is 216 time again.

Believe it or not, if you have set CHX to /d1/cmds, for instance, and you decide to use a command (let's say "list") which is in your /d0/cmds directory you can enter something like:

/d0/cmds/list /d0/txt/message.

Lastly let me touch on redirection again. While making up a new bootfile (OS9Boot) I had some problems. I decided to use the "ident" utility to look at what was in the file. With 30 or 40 or more modules in OS9Boot you just can't get the idea on the screen. So I redirected the output of the ident. A normal ident on a single module might look like this:

ident /d0/modules/cc3disk.dr

By redirecting output like this:

ident /d0/OS9Boot >/p

I had all of the modules information sets printed out in hardcopy instead of the screen.

You don't have to be typing command lines to get errors. Some utilities REQUIRE certain CHD settings, and if you just forgot to do one the utility could be getting the wrong information or none at all.

To really bring your system up to date, FARNA Systems is distributing some very cheap sets of all the things you need to make a new system out of what you started out with (Patch OS-9 and TuneUp).

This is really a good idea, because you can spend days, weeks or even months acquiring all the patches and getting them flawlessly into place. AND you will find that some of the very good public domain software available does not work well unless the original bugs are eliminated from the operating system itself.

A Neat Trick --

I'm going to mention something Rick Ulland suggested, so you can "chew" on it. It comes with a sort of catch 22 built in so be careful how you size it up.

Although the OS9Boot file must be in one contiguous file, not stored in various parts of the disk, it does not need to start at the beginning of the disk. So you could put your commands directory on the disk first - on all your system disks - and the pointer set up by the CHX command would be the same for every disk. Therefore you would not have to reset it except for times when you used /d1.

The trick here is that by the time you might feel you have the expertise to do this you would probably have several system disks set up the other way. Those disks would require "CHXing" as usual and the mix might not be worth the effort.

Section III:

OS9 - The First Step (for CoCo3)

The main problem with OS9 as issued by Tandy is their stupid insistance on adhering to the CoCo 'standard' of 35 track, single sided drives and VDG (very dumb green) 32 column video. Fixing it is a little bit of a pain -- you'll need three blank disks and an hour of work. At least it only has to be done once!

After enduring this procedure, your CoCo should boot to two 80 column windows in RGB mode, with two (or three) 40 track

continued on page 22

HawkSoft

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

MM/1 Products (OS-9/68000)

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

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

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

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

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

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

CoCo Products (DECB)

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

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

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

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

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

SMALL GRAFX ETC.

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

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

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

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

The Embedded Programmer

Paul K. McKneely

More About the Boot Process

In the last article we had an overall view of what happens during the life of the Boot Process. In the early phases of this sequence, a lot of work has to be done just to set up a working environment. Few, if any, of the kernel services are available and the environment is very primitive. Complicating the matter, is the problem of updating an operating system under development.

Boot Images

When power is first applied to the processor there has to be some sort of non-volatile memory to provide the Boot Process with enough software to get the processor on the road to getting the system up and ready for use. In most cases this is an EPROM (Erasable Programmable Read-Only Memory). These are nice because they are low-cost and they are re-programmable. The down-side of using an EPROM is that you must have two pieces of equipment before you can recycle the part: an Ultra-Violet (UV) Eraser and an EPROM Programmer. These are devices that the average computer owner does not have. When you are first developing an operating system or a standalone embedded system, you will be faced with the problem of changing the software every time you come up with a new version to test. If your software is contained in the EPROM then you have to re-program it. This usually means performing the following procedure:

1. Open the cabinet
2. Pry out the EPROM
3. Erase it in the UV Eraser (about 40 min. for my unit)
4. Reprogram it in the EPROM Programmer
5. Replace the EPROM in its socket
6. Close up the cabinet

If I add up all of the time, it takes me close to an hour to do this whole thing. If you have several EPROMs you can shorten this cycle somewhat because you can erase more than one chip at a time. Still, every upgrade requires removal of the EPROM from its socket. Over a period of time your socket pins will pay the toll. This routine is very costly and should be minimized if possible. In the introductory article I talked about another device called an EPROM Emulator that eliminates much of this work. But these devices are even more expensive than an EPROM Eraser and Programmer. A cheaper route for most people is to use a boot ROM containing a program called PREBOOT. This can even eliminate the need to have an EPROM Eraser and Programmer if you know of someone who can do this service for you.

The purpose of PREBOOT is to provide a way to update your ROM software without having to go through the process of erasing and re-programming the EPROM every time you have a version change. But at the same time, you want to keep your BOOT program in a form that can eventually be burned directly into the EPROM and run standalone when it is ready for release. To make this work you need to have some conventions.

Every BOOT image contains two values in its first 8 bytes. These can be best defined using the following 68K assembly source:

```
RESET:  DC.L   ROM_SIZE    ;Length of ROM
        DC.L   COLD        ;Cold Boot Location
```

At reset, the boot ROM is mapped to location 0 so that the SP (Stack Pointer) is loaded from the first four bytes (long word at location 0) and the PC (Program Counter) is loaded from the next four bytes (long word at location 4). After these fetches the ROM socket is typically remapped to some high address. The first four bytes contain a value that represents the total size of the BOOT image. This number includes itself in the count.

You might now be asking the question "Why would you want to load the size of the boot image into the stack pointer?". The answer is that it is built-in to the 68K to load the SP from this location after system reset. But the SP can be loaded later any time the programmer wants and the first few instructions don't depend on the existence of a valid stack pointer anyway.

It is more important to define the first four bytes as a size so that a loader can determine how many bytes are expected in the boot image and so that it can determine where to place it into memory. The second long word in the listing above determines where program execution actually begins. This is a physical location that points to where the first instruction to be executed will reside when the image is itself burned into the EPROM.

This should emphysize a point: boot images are always linked to the location where the EPROM socket is mapped. This is because the PC requires an absolute address before it can begin execution.

Remember that PREBOOT will be loading the boot image during the development phase while testing is under way. It is only when the image is ready for release that we will actually burn it into the EPROM. Until then, PREBOOT resides in the EPROM and it exists merely to load the BOOT image from your development host via some kind of interface such as a serial port of even a floppy disk. PREBOOT is always burned into the EPROM so it does not have to handle the problem of finding out whether it is being loaded or is resident. The BOOT image, on the other hand, must be prepared for either case. For both EPROM images, the first one to come up must determine how much memory is in the system. We discussed how that is done in the last article when using the Peripheral Technology CD68X20 computer.

When PREBOOT comes up, scanning memory is one of the first things to happen. But when BOOT begins to run, before doing any explicit accesses to RAM, it must first find out whether it is running in EPROM or in RAM. If in EPROM, it will have to do the memory scan because it hasn't been done yet. If it is running from RAM then it must assume that it has been loaded into RAM by PREBOOT (or another program) and it must skip the memory scan.

One common way to determine whether you are running in the EPROM versus RAM is to see if you can write into your own image. If you are running in EPROM, the device will not be writable. It will be writable if it is in RAM.

But we have a serious problem here. Many computers have bus fault logic that causes an exception when the processor attempts to write to read-only memory. Since a usable Stack Pointer has not yet been set up before the memory scan has been performed, this method is out. A better way to determine whether you are running in EPROM or RAM is to test to see if you are actually located at the address where the EPROM is mapped. You can do this by executing the following instructions:

```
        LEA    (RESET,PC),A0
```

```
CMP.L  #RESET,A0
BNE.B  SkipSizeMem
```

The Load Effective Address instruction computes the actually run-time location of the beginning of the BOOT image and places it into A0. The next instruction compares it to its linked address. If the two addresses match then the program can assume that it resides in EPROM. If they do not match, then BOOT assumes that it has been loaded by someone else and it skips the memory scan. If BOOT has been loaded then it must assume that the memory scan has already been done and the end of RAM has been loaded into A6. At this point, BOOT can start using RAM.

Partitioning Memory

Once a memory scan has been done, BOOT knows how much memory there is. RAM always begins at location 0 and it ends at the location just prior to the address stored in A6. The first step in setting up main memory for use is to divide memory into a number of partitions. After this has been done, partitions are fixed and cannot be changed without a reset. As discussed last time, our operating system uses up to five partitions:

Absolute
Boot Image
Dynamic Memory
Interrupt Stack
Paged Memory

The first partition (Absolute) is different from the others in two ways. It always begins at the same address (location 0) and its contents are determined at link time. Because its location is fixed, the kernel uses absolute addressing to access it. If you can guarantee that it will be no larger than 32K in size then you can always use short addressing which will save you two bytes of program size with every access.

PREBOOT is not aware of any partitioning done by the BOOT image. Because of this, it must decide by itself where to load the image in RAM. It will then be the responsability of BOOT to relocate itself into the partition which was created to hold itself (the Boot Image Partition).

The process of a program moving itself can be a little tricky. If you were to over-write the code where the move is done before it is finished, the system will crash. You also have to jump into your new location after you are finished. Because the move operation is done early during the boot process, it would be logical to have PREBOOT load the image into location 0 before turning processor control over to the newly loaded image. This would mean that the image will always have to be moved to a higher address location and it wouldn't have to move itself very far before the new location would no longer overlap the relocation routine. If the BOOT Image Partition follows the Absolute Partion in memory then this address is determined by the linker and is known to BOOT before the program begins to run. Here we need to emphysize a point: Since BOOT must run at potentially any address, it must be written using position independent coding techniques.

Let me state an observation. If BOOT finds that it is running from ROM, it is up to the programmer to decide whether BOOT will relocate itself into RAM or not. There are at least two good reason why someone might want to move the BOOT image into RAM. Both of them have to do with speed. Often times RAM (especially synchronous RAM) runs with fewer wait states than EPROM. Secondly, in many systems (the CD68X20 is a good example), the EPROM is only 8-bits wide while the RAM is 16 bits or more in width. Processors with Dynamic Bus Sizing can have a narrowed data path to devices such as an EPROM. In such systems, the processor will require more bus cycles to fetch its instructions and data than if it were located in RAM. The image will run significantly faster if it is moved to RAM.

After BOOT is relocated to its new home, the remaining partitions need to be created. These are done from the end of memory backward. The first of these to be allocated is the last in memory: the Page Memory Partition. Many embedded systems will have no need for page memory because they are diskless. In general, page memory is used to store images of blocks that are buffered from volume-oriented devices. This partition is always last in memory to guarantee that it lies on an even 4K page boundary for the benefit of MMU's that often require block alignment.

After this is done, a partition is created for the Interrupt Stack. The ISP can finally be assigned after this is done. Note that the initial value of the ISP points to the end of this partition. When pointing to the end of the Interrupt Stack Partition, the interrupt stack is said to be empty. As items are pushed onto the stack, the ISP grows downwards toward the beginning of memory.

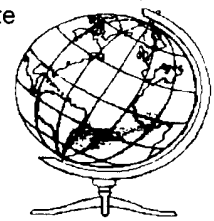
Finally, we have the Dynamic Memory Partition. This partition absorbs the rest of available memory. This partition is divided into 16-byte chunks (called paragraphs) that begin and end on 16-byte boundaries. This memory is allocated/deallocated in contiguous extents in a way very much like the heap in C language environments. This partition is managed by one of the kernel objects called the Dynamic Memory Manager (dm). The dm object maintains a singularly-linked list of free extents of memory. The first four bytes of each extent stores the length (in bytes) of the extent. The next for bytes contains a pointer to the next free extent in the chain. The last extent contains a pointer with the value of 0. This list of free extents is always kept in sorted order.

There are two ways to allocate memory from the pool of free memory. If you request memory with the intension of returning it later, then it is allocated off the beginning of the first extent that is large enough to satisfy the request. If you request memory with the intension of never returning it (until shut-down), then it is taken off the end of the last extent. When memory is returned, dm places it back into its proper position, making sure to merge it with free extents that happen to be adjacent to the one being returned.

This "first fit" alogrithm works very well although you tend to get some fragmentation over time. It might seem a little inefficient at first glance but it is one of the best for keeping fragmentation to a minimum. It tends to keep small fragments toward the front and large fragments toward the back. When fragments of similar sizes are clustered near each other, the system tends to make more use of available memory while keeping large unbroken extents toward the end. Since permanently allocated extents are always taken off the end, they do not contribute to further fragmentation.

Well, we never got around to talking about the Scheduler so I guess we'll put that off until next time. If you have any comments or requests, please feel free to write me at either <gecko@onramp.net> or at the address given below:

Paul K. McKneely
technoVenture, Inc.
P. O. Box 5641
Pasadena, Texas 77508-5641



CoCo3 Extended Memory Secrets Part 6

Herbert Enzman

Final installment!! Loading everything into memory on start-up.

Now that your head is full of knowledge, you have your super program ready and a great title page, it is now time to load it all upon startup. This installment will show you how to get started, and just by typing only "DOS". This is nothing new, there have been magazine articles on using the "DOS" command before; but I had to figure out how to swap blocks to load all my routines at once, something that was not covered in articles.

I kind of like just typing "DOS", because sometimes I am switching disks so often that I start to forget filenames of disks inserted. Besides, typing 'LOADM "XXXXXXXX.XXX"' is a pain. For the novice that doesn't know about the "DOS" routine; a little explanation on how it works.

When you type "DOS", BASIC will load the entire TRACK 34 into memory, at $2600. It then checks the 1st two bytes to see if they are 'OS'. If not, then BASIC returns to it's command mode; but if it is 'OS', then it begins to execute the code at $2602. So all you need is to put your TRACK 34 routine onto track 34, make sure that the first two bytes are 'OS', and you have fooled BASIC into thinking that an 'OS-9' program is present. Your TRACK 34 routine will then load everything that you need, and then execute the MAIN program.

LISTING 16 is the TRACK 34 program that I have been using. The first part writes everything needed onto half of TRACK 34, then flags GRAN 66 as being used to keep it from being written over. It does not give it a filename, so you won't see it in a 'DIR'. I did this as sort of an insurance policy, so that it can't be erased by accident.

The only way to remove it is with a disk editor, a DSKINI, or by writing over it with a new TRACK 34 routine. Just remember that the disk has 1 less GRAN than is displayed in a GRAN count. The 2nd half of TRACK 34 (GRAN 67) is still useable for files. It is only setup for DRIVE 0, so when you write this to disk, make sure the disk in inserted into D-0. Also, a disk editor might give you a "file structure" error due to GRAN 66 not having a file name, but flagged as used. DO NOT answer "YES" if it asks if you want "IT" to fix the problem, because it will just flag the GRAN into use.

I've been using this "DOS" routine for several years, with NO problems. You can make it as simple or complicated as you like, as long as it doen't run over 9 SECTORS (1 GRAN). Otherwise you will have to modify the routine to flag that GRAN 67 is in use; but 9 sectors should be plenty of room.

Now is the time for a disclaimer. This program DOES work, but DO NOT run it as it is; because it is set up for my system. (You will just get NE ERRORS due to the filenames not being there). It is just an idea of HOW the DOS routine can be done. Feel free to modify this routine for your system, and use any or all of it. I just wanted you to know that it won't work for you as it is.

To see what is going on in the routine, just use LISTING 16 to follow along with the explanation.

**** WARNING to standard Disk EDTASM users..... This routine DOES use 6309 code, so make the changes where needed for 6809 code!!!! ****

The "AUTORUN" routine is set up so that the first 256 bytes of data can be reserved for 'DEFAULT DATA' storage. This can be usefull for a very simple 'disk write' routine for changing the 'default data', because it is located on the 1st sector. For the moment, I just use this area mainly for the printer and palette data; but it can grow as the routines grow. You can see this in the listing marked as "printer storage" and "palette storage" areas.

The next section of the routine sets the default data to where it belongs, and then moves on to the TITLE page. It sets the registers where the picture is to be loaded; sets the MMU for GFX mode; sets the screen start; LOADS the picture and runs a timer for the picture display time.

After the TITLE page has been displayed, the routine then sets the MMU up for an 80 column screen. Now we have to move the routine OUT of the $2600 address area, because we want to use the 80 column screen to display what is being loaded; and the 80 column screen uses the address range of $2000 - $3FFF (we don't want to write over the top of our code).

A safe place to put it is at $4000, since it is away from the screen memory and it won't be in the block that will be swapped-out for loading the files (remember from part 1 about planning?). We will use $FFA3 ($6000 - $7FFF) to swap the blocks for loading in the files. After the routine is moved to it's new location, the stack is set and control is turned over to the part of the program labeled "START".

The routine first sets up the added PIA that is the seral/parallel port; sets up the 80 column screen, sets the MMU and swaps block $2D into $FFA3. It now loads the first file (utilities #1) into block $2D, and prints this action to the screen. Next it loads utilities #2 into block $2E, prints this action to the screen, and so on.

As you can see by the listing, the "XFER" routine is used to load the file, after you give it the filename to load. It just calls two DECB routines to do the job. It is important to swap in the block where you want to load the file FIRST.

After it loads the RAMDISK, it formats it. I just used a quick and dirty routine to format the RAMDISK. To save time, the routine just checks to see if the RAMDISK has been formatted (checks two bytes for the letters FM), and if it finds them, then the RAMDISK is already formatted. If not, it just formats track 17 only (sets all bytes to $FF). It really doesn't matter if the other tracks have data in them, just track 17 needs to be formatted to $FF, so that the GRAN TABLE and the directory are free.

The next thing that it does, is load the DOS routine that EDTASM uses. It prints the text "Loading EDTASM 6309", but EDTASM 6309 is not loaded just yet (the DOS routine that was just loaded will do that).

Next some patches are performed to fix some bugs that were in the first version of EDTASM 6309, some patches added for my DSKINI routine

are put where they are needed; and the final action to be taken is to execute the DOS routine so it can load and run EDTASM. All of the subroutines that this main routine uses are located after the text.

A word of caution is now in order. When you are going to swap a block to load a file, then you have to use the range that your program to be loaded uses. For example, I chose to swap the blocks into $FFA3 ($6000 - $7FFF) because the files to be loaded were 'ORGed' at $6000 when written. If I would have swapped in the blocks at $FFA2 ($4000 - $5FFF), DECB still would have loaded them into the $6000 range, because that is their "load" address.

So here again comes that "PLANNING" statement. The $6000 range works out great for EDTASM and EDTASM 6309, due to the fact that the EDIT buffer is in that range, and can easily be switched out for most of the utilities that I have added; without causing any problems. Other programs will use different areas of memory for different routines, so you must find out as much about the programs that you will be using this routine with BEFORE you add it. You can load your routines into any of the $FFAx registers that you like if the above warnings have been met.

Try out your Autoload routine on a COPY of your original disk that you want to add it to first. If any problems occur (I had a few bugs to work out) then your good disk is not ruined!!

LISTING 17 is a short program that changes the LOAD address of the .PIC graphics file that you have saved with Z-BUG during the GFX part of this tutorial. The screen start of that program was located at $8000, so since I needed to display the graphics screen at $6000; I wrote this routine to change the load address of the file on the disk. If you have a disk editor, you can use that to change the load address instead of this program. I just included it for those that don't have a disk editor.

You can modify the first part of LISTING 16 to change the default data on the first sector of TRACK 34; in case you choose to use that option. It is just a simple matter to fix it to load, modify and write the sector back to the disk. You don't even have to worry about the GRAN table, just load sector 1 into a buffer, move the new data to it and then just write that sector back to the disk.

This is the end of this tutorial series. I hope that you have found this information as usefull as I have. I have learned quite a bit more about the COCO 3 and its capabilities since I started on this quest for information. I am glad to share it with you in the hope that you have some use for it; and don't have to wonder how some of these things have been done by other programmers.

It would have been nice if someone would have shared this information long ago; the COCO 3 might not have languished in closets and might have been still going strong today. A severe lack of information to programmers is probably what killed the COCO 3 so early in it's life. I didn't know that it could do most of what I found until I started some of these experiments. Maybe this will breath a few breaths back into the COCO 3, and those that still use it will find some new uses for programs in the future.

LISTING 16 - The Autolaod Routine for TRACK 34 (DOS routine)

```
*** WARNING to Standard Disk EDTASM
users..... This routine DOES use 6309 code.
Make changes where needed to 6809 code! ***


    TITLE     AUTORN.ASC

 ** This portion writes the AUTOLOAD routine to
track 34, then protects GRAN 66 from use by other
files. (it has been now MODIFIED to write 9 sectors
of data)

 ** ASSEMBLE TO MEMORY AND RUN FROM
Z-BUG **

GO  LD    #$00EA
              Point to 'DSKCON' temp start
    LDU   #$07C8
              Point to GRAN TABLE in RAM
    LDQ   #$02001102 = Read / D-0 / T-17 / S-2
    STD   ,X   store   OPCODE / DRIVE #
    STW   2,X  set  TRACK / SECTOR
    STU   4,X  set BUFFER pointer
    JSR   [$C004]   read in track 17 / sector 2
    LDX   #$00EA
              Point to 'DSKCON' temp start
    LEAU  AUTORN,PCR
              Point to DATA to write to the disk
    PSHS  U,X
    LDQ   #$03002201 = Write / D-0 / T-34 / S-1
    STD   ,X   Store OPCODE & DRIVE #
NEXSEC STU    4,X
            Save buffer pointer for 'DSKCON'
    STW     2,X  Set TRACK & SECTOR
    JSR  [$C004]  DO 'DSKCON' ROUTINE
    PULS  U,X
    LEAU  $100,U
          Bump data pointer for next SECTOR
    PSHS  U,X
    INCF        Bump SECTOR count
    CMPF  #9    9 sectors written??
    BLS   NEXSEC NO, loop until all are written
    LEAS  4,S     RESET STACK
    BSR   SECTOR     count sectors used
    ORA   #$C0       Flag that GRAN is in use
    STA   $080A   Put into disk buffer (gran 66)
    LDX   #$07C8   Point to GRAN TABLE
    LDU   #$0635   POINT TO DCB
    CLR   33,U     Set DRIVE '0'
    JSR   $0E7E    Re-write FAT to disk
    CLR   $FF40    Shut off drive motor
    SWI            STOP-FINISHED

**** This part counts the # of SECTORS used (to
flag GRAN table with last SECTOR code)

SECTOR LDD  LENGTH Get program LENGTH
    CLRE           Clear counter
NXSEC INCE         Count sector
    SUBD    #$FF   Subtract 1 sector
    BPL NXSEC  Loop until all are counted
    DECE           Adjust count
    TFR   E,A      Save count
    RTS            RETURN
LENGTH FDB    SAVE-AUTORN+1


THIS IS THE 'AUTORUN' PROGRAM
**************
* DECB will load track 34 into mem. location
$2600. If it finds 'OS' in the first 2 bytes, it will then
begin execution of the program with the 3rd byte.

AUTORN FDB $4F53  = 'OS'
          need to fool DOS to think this is OS-9
    LBRA  JUMP
              Skip over DEFAULT data storage

*****************************************************************

* This section is the DEFAULT data storage area.
MEMAXER 6309 will change this area with the
"SAVE TO DISK" OPTION.  This location is not to
be used by anything else !!
** Printer data storage area

PRNFLG  FCB    $A0
MARGIN  FCB    0
WIDTH   FCB    $50
LPAGE   FCB    $3F
TOP     FCB    0
BAUD    FDB    $0028
RDFLAG  FCB    $FF
               00= no ramdisk  FF= ramdisk

** Palette data storage area

BORDER FCB $00
PALTAB  FDB $1537
    FDB    $2820
    FDB    $000A
    FDB    $1C38
    FDB    $3E2C
    FDB    $2D3F
    FDB    $3632
    FDB    $3400
    FCB    $FF      END OF TABLE

*****************************************************************
```

```
*** PROGRAM execution begins here ***
**************************************************
*** This Part LOADS the palette registers with the
default colors
JUMP  LEAX  PALTAB,PCR
                point to default color table
      PSHS  X
      LDY   #$FFB0   point to palette registers
      BSR   NXTPAL   move table
      PULS  X    point to default color table again
      LDY   #$E678
              point to SUPER ECB palette table
      BSR   NXTPAL  set it up for reset
      BRA   CONTIN  move on to next section
NXTPAL LDA  ,X+          Get default color
      BMI   NOPAL       quit if end of table
      STA   ,Y+      store into SUPER ECB table
      BRA   NXTPAL  loop until all moved
NOPAL  RTS        done, return

*** This part sets up the PRINTER variables and
border register

CONTIN LDD    RDFLAG,PCR
                GET ramdisk flag / border color
      STB   $FF9A    SET border color
      STD   >$0076 SET ramdisk flag/border color
     LDQ   PRNFLG,PCR  GET printer defaults
      STQ   $007C    SET defaults
      STW   $01DF
            SET 'CHARNM & NXLINE' for spooler
      LDA   TOP,PCR   GET 'TOP MARGIN'
      STA   $0080     SET it
      LDD   BAUD,PCR    GET 'BAUD RATE'
      STD   $0095     SET it

 ** This part sets-up the title page, loads and
displays it; when it times-out, the 80 column screen
is reset and the program continues to load files.

      LDD   #$3031 set (2) blocks for GRAPHICS
      STD   $FFA2    put them here
      LDA   $FFB0      ###
      LDB   $FFB3      ###
              get palette colors and save for later
      PSHS  D         ###
      LDD   #$0034    ***
                set colors for graphics screen
      STA   $FFB0      ***
      STB   $FFB3      ***
      LEAX  NAME99,PCR
                filename for graphics screen
      LBSR  XFER     load the file
      LDD   #$8011   graphics data
      STD   $FF98
            set BIT PLANE bit and video resolution
      LDA   #$C0
      STA   $FF9D       set screen start
      LDA   #$4C
      STA   $FF90       set MMU
      LDX   #$FFFF      ***
      LDY   #$0007      *** set timer
LOOP2  LEAX  -1,X        ####
      BNE   LOOP2       ####
      LEAY  -1,Y        ####
                Display picture until timer is done
      BNE   LOOP2       ####
      LDD   #$0315
      STD   $FF98    reset for text
      LDA   #$D8
      STA   $FF9D    reset for text
      LDD   #$3A3B   Get origional blocks
      STD   $FFA2    put them back

*** This part moves the program to the $4000 area so
that MMU blocks can be switched in and out for
loading of various files.

      LEAX  START,PCR   = source
      LDY   #$4000      = destination
      LDW   #END-START  = byte count to move
      TFRP  X,Y     Move program
      PULS  D       get saved colors
      STA   $FFB0       ****
      STB   $FFB3       **** restore colors
      LDS   #$5FFF      set stack here
      JMP   $4000
              execute program at NEW location

*** This is the MAIN BODY of the program that is
executed after it has been moved. It will load all
the files that MEMAXER 6309 needs. It will set
the 80 column screen and clear it; set the MMU
and display the filenames as they are being loaded.

START ORCC #$50
      LEAY  SETAB,PCR  point to PIA setup table
      LDX   #$FF4C
              point to new PIA (parallel port)
      LDD   ,Y++    get setup data
      LBSR  PIA     set it
      LEAX  2,X     bump pointer
      LDD   ,Y      get next data
      LBSR  PIA     set it
      JSR   $F679  Set 80 column screen width
      LDA   $FFA3  Get CURRENT block #
      PSHS  A      Save it for later
      LDD   #$2D4C
        $2D= segment to change; $4C= set MMU
      STB   $FF90     Set MMU
      STA   $FFA3
              Set segment for loading of data
      LBSR  CLEAR    Clear 80 column screen
      LEAX  TXT4,PCR
                "MEMAXER 6309 LOADER"
      LDY   #$203E   where on screen to put it
      LBSR  STRING  Go print text onto screen
      LEAX  TXT5,PCR  "COPYRIGHT....."
      LDY   #$2164   where on screen to put it
      LBSR   STRING print text onto screen

***   This loads the 'UTILITIES #1' Program into
block # $2D

      LDY   #$23FC
                Where on screen to put text
      LEAX  TXT1,PC "LOADING UTILITIES #1"
      LBSR  STRING     Print text onto screen
      LEAX  NAME1,PCR
                Point to FILENAME to load
      LBSR  XFER
 Put it into FILENAME BUFFER for DECB to load

*** This part loads the 'UTILITIES #2' program

      LDA   #$2E   Put program into this block
      STA   $FFA3
      set the block where it is to be loaded
      LDY   #$253C      screen text location
      LEAX  TXT8,PCR
              "LOADING UTILITIES #2"
      LBSR  STRING       Print it
      LEAX  NAME4,PCR    Filename to load
      LBSR  XFER         Setup for load

***    This part loads the 'XCHANG' Routine

      LDA   #$2A        BLOCK to put it
      STA   $FFA3       Set block
      LEAX  NAME5,PCR    point to filename
      LBSR  XFER   setup for DECB to load it

*** This part loads the PRINTER SPOOLER
program

      LDY   #$267C   screen location for text
      LEAX  TXT9,PCR
              "LOADING PRINT SPOOLER"
      LBSR  STRING      print it to screen
      LDA   #$17  BLOCK # where it is to be loaded
      STA   $FFA3       set it
      LEAX  NAME6,PCR    point to filename
      LBSR  XFER      let DECB load it now
      PULS  B       GET ORIGIONAL BLOCK #
      STB   $FFA3       RESTORE IT

*** This part loads the "SWAP BLOCK" program at
$0408

      LDY   #$27BC      screen location for text
      LEAX  TXT10,PCR
                "LOADING BLOCK SWAP"
      LBSR  STRING      print it
      LEAX  NAME7,PCR    Point to filename
      LBSR  XFER        let DECB load it

*** This part loads the RAMDISK program. We will
only use 1 RAMDISK as DRIVE -2, If ramdisk is not
wanted, then no text will be displayed on the
screen, and drive 2 wil be active. Ramdisk can be
activated later.

      LEAX  NAME3,PCR  filename to load
      LBSR  XFER     let DECB load it
      LBSR  FORMAT   now, go format the ramdisk
      TST   >$0076      Ramdisk wanted?
      BPL   SKIP     NO, then no text on screen
      LDY   #$28FC      Screen location
      LEAX  TXT3,PCR   "LOADING RAMDISK"
      LBSR  STRING      print it

** This part loads the 'LOGON' routinne at $5600

SKIP   LEAX  NAME8,PCR  = Filename to load
      LBSR  XFER     Let DECB load it

***  This part loads the 'DOS' program that
EDTASM uses

      LDY   #$2ADC      screen location
      LEAX  TXT2,PCR     "LOADING  DOS"
      LBSR  STRING       print it
      LEAX  TXT7,PCR
                "LOADING EDTASM 6309"
      LDY   #$2C1C      screen location
      LBSR   STRING      print it
      LEAX  NAME2,PCR     Filename to load
      LBSR  XFER        let DECB load it

***  Set up INTERCEPTS for 6309 'BUG' repair

      LDQ   #$004E1212
      STA   $CD22
      STA   $C7A6
      STB   $C75A
      STW   $C79F
      STW   $C7A1
      STW   $C7A3
      LDQ   #$21BD1FDB
      STQ   $CCFC
      LDQ   #$007E0EF4
      STB   $AE29       store 'JMP' opcode
      STW   $AE2A  Intercept 'return to basic' here

****  Make changes for the DSKINI routine here
```

```
      LDD   #$1212    = NOP NOP
      STD   $D5DC      **
      STD   $D5DE ** stop TOP OF RAM test
      STD   $D5E0      ##
      STA   $D5E2    ## stop CLOSE ALL FILES
      STD   $D5E6      &&
      STD   $D5E8      && don't set stack here
      LDD   #$4000     = track fill buffer
      STD   $D667      change it here
      LDD   #$4100     = skew table
      STD   $D66C      change it here
      STD   $D69D      and here
      STD   $D725      and here too
      LDD   #$4113     = track data buffer
      STD   $D632
      STD   $D692
      LDD   #$4112     = end of skew buffer
      STD   $D5B5
      LDA   #$39 = RTS
      STA   $D685      set return from DSKINI

*** Set locations for 'RAMDISK' intercepts

      LDD   #$7EE2
      LDW   #$8B12
      STD   $D75F
      STW   $D761

********** EXECUTE "DOS" ROUTINE **********

      ANDCC #$AF       enable interupts
      JMP   [$009D]    EXECUTE PROGRAM

********** TEXT **********
*** These are the FILENAMES of programs to be
loaded. NOTE the spacing between filename and
extension !

NAME1  FCS   /OPTION BIN/
NAME2  FCS   /DOS    BIN/
NAME3  FCS   /RAMDSK BIN/
NAME4  FCS   /KILL   BIN/
NAME5  FCS   /XCHANG BIN/
NAME6  FCS   /SPOOLER BIN/
NAME7  FCS   /SWAP   BIN/
NAME8  FCS   /LOGON  BIN/
NAME99 FCS   /GFX    PIC/

* This is the text for putting onto the screen. The
COLOR attribute is BEFORE the text (FCB $xx). It
is used by the 'STRING' routine to set the color of
the text when printing to the screen.

      FCB   $24      screen attribute
TXT1  FCS   /Loading UTILITIES #1/
      FCB   $24
TXT2  FCS   /Loading  DOS/
      FCB   $24
TXT3  FCS   /Loading  RAMDISK/
      FCB   $1C
TXT4  FCS   /MEMAXER - 6309 Loader/
      FCB   $1C
TXT5  FCS   /Copyright (c) 1994 MMIT
Technologies, Ltd./
      FCB   $24
TXT6  FCS   /RAMDISK FORMATTED =
DRIVE 2/
      FCB   $24
TXT7  FCS   ?Loading  EDT/ASM - 6309?
      FCB   $24
TXT8  FCS   /Loading UTILITIES #2/
      FCB   $24
TXT9  FCS   /Loading PRINTER SPOOLER/
      FCB   $24
TXT10 FCS   /Loading BLOCK SWAP/

SETAB FDB   $0204
      FDB   $FF2C

********** SUBROUTINES **********

*** Move TEXT to DECB filename buffer

XFER  LDY #$094C  = FILENAME buffe location
NEXFER LDA  ,X+      Get character to move
      PSHS  A        save for 'stop' test
      ANDA  #$7F       drop MSB
      STA   ,Y+      store it into buffer
      TST   ,S+      Last character?
      BPL   NEXFER   Loop until all printed
      JSR   $CA07    Locate filename on disk
      JSR   $CFE3
            LOAD the PROGRAM from the disk
      RTS            RETURN
*
**** Clear the 80 column screen
*
CLEAR LDY   #$2000 = Screen start
      LDA   #$20   = SPACE
      LDE   #$3C   = BACKGROUND COLOR
CLR LBSR  SCREEN   send it to the screen
      CMPY  #$2EFF     End of screen?
      BLO   CLR
              Loop until entire screen is cleared
      LDA   >$0077   get default BORDER color
      STA   $FF9A
          Set BORDER color to the default value
      RTS            return — done
*
*** SET 80 COLUMN SCREEN AT $FFA1 ($2000-
$2EFF) *** This routine sets MMU segment $36 at
$2000 - $2EFF so that we can write to the 80
column screen.
*
FIX   ANDA    #$7F       drop MSB
SCREEN PSHS   B
      LDF   #$36     = MMU segment for screen
      LDB   $FFA1
              GET block # that is there NOW
      STB   SAVE   SAVE it for return
      STF   $FFA1  SET $FFA1 for screen use
      STA   ,Y+      PUT character onto screen
      STE   ,Y+      PUT it's attribute next
      LDB   SAVE     GET origional MMU block #
      STB   $FFA1      RESTORE IT
      PULS  B,PC       RETURN
*
****     Write text to 80 column screen
*
STRING LDE    -1,X     Get attribute for text
NXCHAR LDA    ,X+    get text character to print
      PSHS  A        save for 'STOP' test
      BSR   FIX      Go print character to screen
      TST   ,S+      last character ?
      BPL   NXCHAR   NO, loop until all printed
      RTS            return — done
*
*** This part first checks to see if the RAMDISK
has been formatted; if not, then it formats it. Only
track 17 needs formatting to set the FAT and DIR
areas to $FF (unused). It really formats track 17/1
thru track 18/14 (the entire block) BUT it gets the
job done easily without extra code.
*
FORMAT NOP
      LDE   $FFA3
              Get the origional block # for return
      LDF   #9       ** SET TRACK 17
      STF   $FFA3
            ** INTO THIS BLOCK ($6000-$7FFF)
      LDX   #$6000   = TRACK 17 SECTOR 1
      LDD   #$464D     = "FM" (format code)
      CMPD  ,X++
              Has it been previously formatted?
      BEQ   FOREND     YES, THEN EXIT
      STD   -2,X
          Flag that RAMDISK is now formatted
      LDD   #$FFFF     = FREE
FORNXT STD  ,X++       Format TRACK 17
      CMPX  #$7FFF     Entire track formatted ?
      BLO   FORNXT     NO, loop until all done
FOREND STE $FFA3   RESET origional block #
      TST   >$0076     ramdisk wanted?
      BPL   SKIP2      no, then no text
      LEAX  TXT6,PCR
        ="RAMDISK FORMATTED = DRIVE 2"
      LDY   #$2E96     Where on screen to put it
      LBSR  STRING     Print it to screen
SKIP2 RTS              return
*
PIA   CLR   1,X
      STA   ,X
      STB   1,X
      RTS
*
*
SAVE  RMB   1
          STORAGE FOR MMU SEGMENT
END   EQU   *          PROGRAM END
MARKER for calculation of program length.
      END
*
```

```
***** LISTING 17
*
** This is where YOU put the NEW LOAD address
and the filename of the picture. (Your filename goes
where the GFX.PIC is). (NOTE spaces between
filename and extension if filename is not 8
characters)

START  FDB   $XXXX
        Address where you want picture to load
FILNAM FCC   /GFX    PIC/
        filename of picture

*** This is the MAIN body of the program **

GO    CLR   FLAG       flag directory routine
      LDD   #$1102     = track 17  sector 2
      STD   TRACK    start with track 17, sector 2
      LBSR  READ
                  go read directory into memory
      LEAU  BUFFER,PCR  point to file buffer
LOOK  LEAX  FILNAM,PCR
                  point to filename for compare
      PSHS  U          save pointer
      LDD   #$0A20
          = 10 characters / 32 in directory format
      PSHS    D          save for later
LOOK2 LDA,X+       get character from buffer
      DEC   1,S    count directory format number
      DEC   ,S       count filename character
      BEQ   YES      match found - now move on
      CMPA  ,U+        character match?
      BNE   NO   NO, then this is not the filename
      BRA   LOOK2      check next character
NO    PULS  D          get count
      LEAU  B,U
          adjust directory pointer to next filename
      LEAS  2,S        purge 'U' from stack
      CMPU #BUFFER+$0C00   end of directory?
      BLO   LOOK
              no, then continue to look for match
```

```
EXIT   LDB   #4       "file not found" error
       LBRA  ERRTXT
                  EXIT with error code in 'B'
YES    LEAS  2,S      purge 'D' from stack
       PULS  U
             get buffer pointer for filename location
       LDB   $0D,U
             get start GRAN from directory entry
       LBSR  CONVRT
          go convert GRAN to TRACK / SECTOR
       COM   FLAG     flag 'change address' routine
       LBSR  READ     go put 1st sector into buffer
       LDD   START    get NEW LOAD address
       LEAU  BUFFER,PCR
                  point to start of 1st sector
       STD   3,U      change LOAD address of file
       LBSR  WRITE
             now, write the sector back to the disk
CHANGE SWI
             change complete - EXIT to Z-BUG

** This part will read / write to the disk **

READ  LDA #2        = read opcode
      FCB  $8C      = CMPX instruction
WRITE LDA #3        = read opcode
      LDX  #$00EA   DSKCON buffer pointer
      STA  ,X       set opcode for DSKCON
      LDA  TRACK    get track number
      STA  2,X      save for DSKCON
      LDA  SECTOR   get sector number
      LEAU BUFFER,PCR  point to I/O buffer
NEXT  STU 4,X  save buffer pointer for DSKCON
      STA  3,X save sector number for DSKCON
      PSHS U,X      save pointers
      CLR  6,X      clear status
      CLR  $FF91    set TR=0 to access ROMs
      JSR  [$C004]  do DSKCON routine
      LDB  #1       ##
      STB  $FF91    ## set TR=1
      PULS U,X      get saved pointers
      TST  6,X      check DSKCON status
      BNE  DSKERR   EXIT if error
      TST  FLAG
          which routine is using READ / WRITE?
      BMI  DONE
       EXIT if address change;continue if directory
      INC  SECTOR   bump sector number
      LDA  SECTOR   get sector number
      LEAU $100,U   adjust buffer pointer
      CMPA #12      last directory sector?
      BLS  NEXT
             NO, loop until all sectors in buffer
DONE  RTS   exit back to calling routine

** this routine will convert GRAN number to
TRACK / SECTOR **

CONVRT PSHS  B    save GRAN number
       LSRB       divide by 2
       STB  TRACK    save track number
       CMPB #17      directory track?
       BLO  NOBMP    no, ignore next instruction
       INC  TRACK    bump track number by 1
NOBMP  ASLB          multiply by 2
       NEGB          negate result
       ADDB ,S+      add GRAN number
       TSTB
          B=00= even GRAN  B=1= odd GRAN
       BEQ  SAVE     jump if EVEN
       ADDB #7       adjust for ODD GRAN
SAVE   INCB          set sector for odd / even
       STB  SECTOR   save sector start
       RTS           RETURN
```

```
** disk error handler routine - will exit with error in
'B' register

DSKERR LDA 6,X     get DSKCON 'status' bit
       LDB  #8     = "write protect" error code
       BITA #$40   write protected?
       BNE  ERRTXT    yes, exit with error in 'B'
       LDB  #1     "drive not ready" error code
       BITA #$80   drive ready?
       BNE  ERRTXT
                  no, exit with error code in 'B'
       INCB        = other disk errors
ERRTXT SWI
          exit to Z-BUG with error in 'B' register

** program TEMPS ***

TRACK  RMB  1       track storage
SECTOR RMB  1       sector storage
FLAG   RMB  1
00= directory routine FF= address change routine
BUFFER RMB  $0C00   = I/O buffer
       END
```

ERROR codes in 'B' if error occured: (read 'B' register with Z-BUGs 'R' command)

04 = FILE NOT FOUND
08 = DISK WRITE PROTECTED
01 = DRIVE NOT READY
02 = OTHER ERRORS

It looks like an adventure in assembly language programming is next. Art Flexser has given permission to use a series he wrote on some assembly language techniques. This series assumes some assembly knowledge, but does lean toward the beginning or inexperienced assembly programmer. See you next issue!

operating system 9

continued from page 15

double sided drives @ 6ms step rate, with your choice of serial port and printer baud rates.

1) Make a copy of the basic09 disk using backup (you can even do this from DECB if you want), then put the new disk in drive 1, and the system master in drive 0.

2) Type: chx /d0/cmds;chd /d1/modules;edit bootlist [when you see "type:", you always press "ENTER" at the end of a line.]

Bootlist is just that, a list of the modules to be used when making a custom version of OS9Boot. Since the version Tandy included is pretty lame, we are going to change it with edit to use 40 track drives and the windowing system. Since you entered "edit bootlist" you should now be seeing E:, the edit prompt.

3) Type:

c*/35s.dd/40d.dd/ [c* = change all occurances]
c/vdg.dt/win.dt/
q

Q saves the changes and returns to OS9: prompt.

4) Now to load some stuff into RAM to free up the drive the system disk is in, type: load format;load os9gen

5) Remove the system disk from drive 0 and insert a blank one in it's place. Type: format /d0

6) When the formatting is done, type:
os9gen /d0 </d1/modules/bootlist
[makes OS9Boot using bootlist]

7) Now, remove the BASIC09 backup disk from drive 1 and replace it with the System Disk! Type:
chd /d1; chx /d1/cmds; dsave /d1 / d0 ! shell
[copies rest of disk]

We have to use dsave here because backup would clobber the new boot just made. This is a good time to take a coffee break. Dsave will take something like 5 minutes to copy the entire system disk file by file. That's version 1 - a 40 track capable boot stuck on a 35 track disk.

8) Press reset once to reboot the system. It should come up in a 40 column term screen.

9) Now there are a few more things to change before making version 2 - the true 40 track boot disk.

10) First to change term to 80 columns and set the 6ms stepping rate for the disk drives. To do this, we'll use modpatch.

Modpatch uses a file that lists the changes to be made. To create this file, we'll use edit again. Type the following - (Note each line between edit and q starts with a space).

```
edit patch
I term
c 002c 28 50
c 0030 01 02
v
I d0
c 14 00 03
v
I d1
c 14 00 03
v
```

q

Now type: modpatch patch

11) There are a few last changes -- permanently setting printer and modem baud rates. In the following lines, replace # with 3 for 1200, 4 for 2400, 5 for 4800, 6 for 9600, or 7 for 19.2K baud.

```
xmode /t2 baud=#     [makes changes
xmode /p baud=#       in memory]
```

12) Now everything is set up in RAM. All thats left is to save the final version to disk. Format one last blank disk in drive 1. This time it should verify all they way to 4F and probably show $5A0 sectors, signifying a 40 track double sided disk. Now type:

```
cobbler /d1   [transfers OS9Boot from memory]
chd /d0
dsave /d0 /d1 ! shell  [rest of disk]
```

You'll have a short wait between commands.

13) When dsave gets done, reboot with the new disk. This disk has more than twice the room the old one did; so the Basic09 stuff will fit on it. Copy it over as follows.

Assuming new Master in /d0, Basic09 in /d1, type:

```
chd /d1/cmds
copy Basic09 /d0/cmds/Basic09
copy Runb /d0/cmds/Runb
copy gfx2 /d0/cmds/gfx2
copy inkey /d0/cmds/inkey
copy syscall /d0/cmds/syscall
```

You'll note I didn't include gfx -- it's only for CoCo1/2 compatable Basic programs. Copy it if also if needed.

14) One last thing you'll like is a second window started automatically (and I guess RGB mode would be nice [assuming you are using an RGB monitor]).Once again, time to edit! (Assumes Master in /d0) Type:

```
chd /d0
edit startup
+*
montype r
iniz /w7
shell i=/w7&
q
```

(Finally) Done!

P.S. No attempt was made to explain what every line given above does, since a reasonable explanation would swell this article to unmanageable size. The idea was to just get the job done, and figure it out later.

Near The End...

This last section is a selection of common problems:

OS9 has the equivalent of two DRIVE(x) commands- chd and chx. The first points to current data directory and the second to current execution dir (usually a CMDS dir).

They are a little more complex than DRIVE(x), due to the directory tree system used by OS9. Not only do you have to repoint to use a different drive, but also when using a different disk in the same drive, since its directories may be in a different place on the disk. A 216 or 214 erroroften means you forgotten to do a chx or chd.

To avoid the need to chx or chd all the time, you can use a full pathlist to get a file. Say you're pointed to a disk in drive 1, and need to run something off of the system disk in drive 0. /d0/CMDS/something will find it.

Another reason for a 214 error is that the file execution attributes are not set- you'll notice practically everything in the os9 sig and other file sites is an .ar file. When you burst these files with Ar, it doesn't always set the execute attributes, so the file won't run. Use "attr filename e pe" to make it runable.

OS9 doesn't have an llist command, instead you have to redirect to the printer. list file >/p will do that. Most commands may be redirected. To get a printed directory dir >/p. And so on. If you have a long file to print, you can do it in the background by adding an ampersand: list file >/p&

One other neat redirection -- if you are in conference and want to show somebody your module directory (or

RGBoost - $15.00

If you want to speed up DECB easily, install an Hitachi 6309 and get RGBoost. This patch for DECB uses the extra 6309 functions for up to a 15% gain in overall speed. It is compatible with all programs tested to date! Save an additional $5 by purchasing RGBoost along with one of my other products listed below!

EDTASM6309 v2.02 - $35.00

Patches Tandy's Disk EDTASM to support Hitachi 6309 codes! Supports all CoCo models, including stock 6809 models. CoCo 3 version uses 80 column screen, runs at 2MHz. YOU MUST HAVE A COPY OF DISK EDTASM. This is a PATCH ONLY! It will not work with "disk patched" cartridge EDTASM

CC3FAX - $35.00

Receive and print weather fascimile maps from shortwave! The US weather service sends them all the time! Requires 512K CoCo3 and shortwave receiver. Instructions for simple cable included.

HRSDOS - $25.00

Move programs and data between DECB and OS-9 disks! Supports RGB-DOS - move files easily between DECB and OS-9 partitions! No modifications to OS-9 modules required.

DECB SmartWatch Drivers - $20.00

Access your SmartWatch from DECB! Adds function to BASIC (DATE$) for accessing date and time. *Only $15.00 with any other purchase!*

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
Please add $4 S&H per order

VIDEO GAME MAGAZINES WANTED!

Video Games Player, Atari Age, Electronic Fun, Electronic Games, Replay, Playmeter, Blip, and Atarian. Write with what you have and what condition they are in and I'll send back a quote. Video Games, Box 9542, Pittsburg, PA 15223

whatever), clear key to another window and redirect to /t2, as in: mdir >/t2. It will look a little goofy on your end, but everyone else sees a pretty formatted mdir.

Something apparently missing is CLS. It's not really missing, just hidden. Use display c.

Many of the 3rd-party utilities contain a built in help file. To see it, type:

utilityname -?

Printing these help files often stumps folks, since they are output on the error path. Use utilityname -? >>/p if the normal method doesn't work.

Starting another window isn't that difficult. To form a full screen 80 column window with a shell in it, type:

```
iniz /w#
display 1b 20 02 0 0 50 18 1 0 0 > /w#
shell i=/w#&
```

Where # is the window you want to start (notice we've already used 7), and the last 3 numbers in the display command (1 0 0) are foreground, background, and border color. These are hex numbers, so 50 18 is 80x24. There is a utility (wcreate) that uses decimal numbers, but display is faster, since it's always in ram.

I wish you the best of luck in your explorations of the CoCo and OS-9!

-Rick Ulland

Rick can be reached at:

CoNect

1629 South 61st Street
West Allis, WI 53214
(pulland@omnifest.uwm.edu)
414-328-4043

It should be pointed out that Rick also carries the "Patch OS-9" disk set.

Jim LaLone can be reached via e-mail as: termite@delphi.com

Jim wrote this file several years ago and may not be available for comment.

OS9:

NEW Hardware coming from

Cloud Nine

c/o Mark Marlette
3749 County Road 30
Delano, MN 55328
email : mmarlett@isd.net
voice: 612-972-3261

512k - 2048k upgrade board

Just install SIMM memory in 512k increments (2x256K 8 or 9 chip SIMMs). Three chip SIMMs WILL NOT work! This is a timing requirement, as the 8/9 chip SIMMs use the same timing as the CoCo DRAM upgrades.

SCSI Host adapter interface

- Comes with OS9 Drivers, 6x09. 63b09e 1.78MHZ system "megaread" times are ~11 seconds with 512 byte sectors (Nitros 2.00 Level3).
- 256/512/1024 Sector size selection
- FULL SCSI ID supported
- Parity generation, enable/disable. Can use with parity devices such as ZIP drives!
- Gold plated card edge connector
- 50 pin SCSI header port
- Installation/Operation Manual
- Schematic package
- OS9 Utilities SCSItools, SCSIdesc, ZIP/JAZ Tools
- SCSItools - A BASIC09 utility that will do low level SCSI commands.
- SCSIdesc - A BASIC09 utility program that will create the SCSI descriptor for you based upon the menu drive options inputted.
- ZIPJAZtools - This utility will allow the features of the Iomega ZIP/JAZ drives. Eject disk, software protection are some. This utility isn't written yet, but I have the documentation needed from Iomega. Will do this soon!

These products should be available at the Chicago CoCoFest! Look for me there!!

A 512K SIMM upgrade is ready to ship. The unit will ship with the following items:

1 - 512K SIMM Memory Board with 8 or 9 chip 120ns or faster SIMMS
1 - Installation Manual
1 - Schematic package
1 - RSDOS Memory Test Program supplied on 5 1/4" disk.

$40 each including shipping, UPS ground, within the US. If you are outside of the US please indicate method of shipment desired and I will check into the added cost, if any.

BlackHawk Enterprises

New Products!

- Data Windows - $69.95 - A complete flat database program for OS-9/68K. Facilities include database creation, searching, maintenance and report generation. By Alpha Software Technologies.
- GNU TWO - $49.95 - This package include a new port of GNU M4, and the AUTOCONF automatic configuration macros. Together with the included port of BASH these tools make automatic configuration of software a much easier chore. Widely used on UNIX and other operating systems, use it now on your OS-9 platform! Includes two new manuals totaling about 110 pages.
- Model Rocketry Tools - $15 - Includes ports of tools for modeling and tracking the performance of various configurations of model rockets. Essential tools for those interested in designing rockets or achieving specified altitudes. Should run on any OS-9/68K machine.

MM/1, MM/1a and MM/1b hardware and other software still available, inquire!
P.O. Box 10552 • Enid, OK 73706-0552 • (405) 234-3911

Seahawks

Stephen Macri

This is a text style Football game in BASIC for the CoCo 3. A knowledgeable CoCo 2 owner should be able to convert it to run on the CoCo 2 (which it was converted from!). One person plays against the computer.

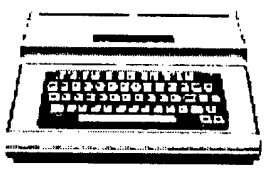
```
1 PALETTE RGB
2 PCLEAR 1
3 CLEAR 8000
4 PALETTE0,0
5 CLS0:PALETTE8,52
6 WIDTH 80
7 HBUFF1,4392
8 KN=180.95:YZ=9999
9 ZD=3:ZF=0
10 POKE 44014,75:POKE44015,78
11 PALETTE 1,52
12 HSCREEN 4
13 HCOLOR1,0
14 ON BRK GOTO 558
15 HGET (500,65)-(640,186),1
16 REM ** ALL RIGHTS RESERVED **
17 REM CONVERTED AND MODIFIED TO COCO 2
18 REM USING WETWARE ON OR ABOUT:(NOV-
11-1985)
19 REM CONVERTED TO COCO 3 (DEC-1986)
20 REM BY: STEPHEN MACRI
21 REM DELPHI USER <DRACMAN>
22 I$=▒9999"
23 IN$=▒KELLY▒:I=3434
24 HPRINT(18,5),▒Welcome to C-Hawk Ball
!▒
25 HPRINT(19,7),▒** COCO 3 PROGRAM **▒
26 HPRINT(15,10),▒* A FOOTBALL TEXT
SIMULATION *▒
27 HPRINT(14,15),▒By: Stephen ( DRACMAN
)▒
28 HPRINT(7,20),▒Instructions for
playing FootBall...type Y or N <ENTER>▒
29 INPUT IN$
30 IF IN$=▒Y▒ OR IN$=▒YES▒ THEN I=1
31 IF IN$=▒N▒ OR IN$=▒NO▒ THEN I=55
32 IF I=1 THEN GOTO 35
33 IF I=55 THEN GOTO 65
34 GOTO 28
35 HSCREEN 0
36 CLS
37 PRINT▒OFFENSE :
38 PRINT▒ plays :▒
39 PRINT▒ ▒
40 PRINT▒ RUN▒
41 PRINT▒ PASS▒
42 PRINT▒ SWEEP▒
43 PRINT▒ SCREEN PASS▒
44 PRINT▒ LONG PASS▒
45 PRINT▒ DRAW PLAY▒
46 PRINT▒ PUNT▒
47 PRINT▒ FIELDGOAL▒
48 LOCATE40,0:PRINT▒ DEFENSE :▒
49 LOCATE41,1:PRINT▒ plays :▒
50 LOCATE41,3:PRINT▒ NORMAL▒
51 LOCATE41,4:PRINT▒ HOLD▒
52 LOCATE41,5:PRINT▒ BLITZ▒
53 LOCATE41,6:PRINT▒ INTERCEPT▒
54 LOCATE41,7:PRINT▒ BLOCK (KICKS)▒
55 LOCATE0,13:PRINT▒At any time during
a play you may call a timeout.▒
56 PRINT▒Permitted only three timeouts
per half.▒
57 PRINT▒The PLAY menu is displayed at
the lower-right of the screen.
58 PRINT▒Move UP and DOWN arrow keys to
select PLAY from menu.▒
59 PRINT▒When you have selected your PL
AY press <ENTER>.
60 PRINT▒Of course I assume you are fam
iliar with the Game of Football alread
y .
61 PRINT▒For Educational Use only.. Enj
oy !!! Stephen (DRACMAN)▒
62 LOCATE5,22
63 PRINT▒Press any key to continue...▒;
64 EXEC &HA171
65 REM ENJOY
66 HSCREEN 4
67 DEF FNT(X1)=SIN(X1)/COS(X1)
68 DIM R(17)
69 FOR I=0 TO 17
70 READ R(I)
71 NEXT I
72 DATA 9,13,100,0,9,10,12,11,12,0,1,5,
3,4,6,2,8,7
73 F=0
74 Z3=0
75 O=0
76 L=0
77 DIM Z(5,3),P(2),D(2,5)
78 FOR A=0 TO  5
```


Ron Bull invites you to attend...

PennFest '98

The Second Pennsylvania CoCoFest!

Buy Software! Buy Hardware! Meet New & Old Friends!
Learn New Tricks! Hear Guest Speakers! Have Fun!



August 15th and 16th
(9am - 5pm each day.)
HOLIDAY INN EAST
4751 Lindle Road
Harrisburg, PA 17111

Last year, 150 CoCoists made the trek to historic PENNSYLVANIA for the first PennFest. Special guests such as John "Sock Master" Kowalski and Steve Bjork held free seminars. This year Ron hopes to enhance the show by providing a better location (closer to the airport, with an airport shuttle available), contacting MORE vendors, and using the experience from the previous show to hold an even better one this time! Summetime is a great time to visit Pennsylvania with your family (and a special family pass is available) to see the sights, as well as the CoCoFest!

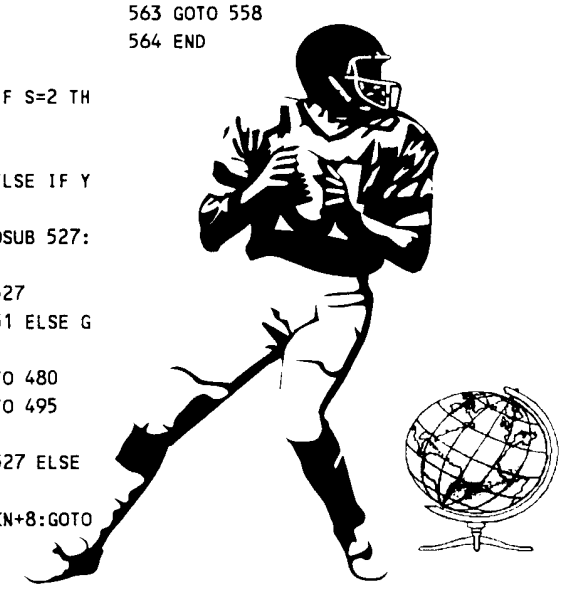
Early Registered Vendors Include: Monk-O-Ware, Sub-Etha SoftWEAR, Ron Bull, Carl Boll, FARNA Systems, Mike Guzzi, CoCoPS, Unl. Electonics Repair, and the Glenside CoCo Club.

www.geocities.com/SiliconValley/Vista/1412 - ronbull@aol.com

From the PA Turnpike: Take exit 19. From I 283: Take exit 1 and turn right. (Shuttle busses from the airport provided.) Call (717) 939-7841 for reservations!

Admission is $5 per person, per day, or $15 for a "family pass" good for both days. Call Ron at (717) 834-4314 for more information.

```
79 FOR B=0 TO 2
80 READ Z(A,B),D(B,A)
81 P(B)=0
82 NEXT B
83 READ Z(A,3)
84 NEXT A
85 HCLS:HPRINT(10,10),"DO YOU WANT TO
RECEIVE ? "
86 X=1
87 HPRINT(5,14),"YES OR NO"
88 HPRINT(2,16),STRING$(71,"-")
89 HPRINT(65,20),"YES"
90 HPRINT(65,21),"NO"
91 GOSUB 460
92 S=2-SGN(13-Q)
93 IF S=1 THEN YZ=4444 ELSE IF S=2 THE
N YZ=2222
94 K=S
95 T1=120
96 U2=3
97 U=3
98 C=900
99 GOSUB 291
100 F1=50
101 B=INT(F1+O*20*RND(-TIMER)+(1-
O)*29*(2-RND(-TIMER)^7-RND(-TIMER)^(3-
Z2)))
102 O=0
103 Z9=8
104 GOSUB 202
105 L=0
106 IF B<99 THEN 300
107 HPRINT(ZD,ZF),"A TOUCHBACK"
108 ZD=ZD+12:IF ZD>60 THEN
ZD=3:ZF=ZF+1
109 B=20
110 L=0
111 IF S=2 THEN 115
112 HPRINT(ZD,ZF),"MY"
113 ZD=ZD+3:IF ZD>60 THEN ZD=3:ZF=ZF+1
114 GOTO 117
115 HPRINT(ZD,ZF),"YOUR"
116 ZD=ZD+5:IF ZD>60 THEN ZD=0:ZF=ZF+1
117 GOSUB 416
118 GOSUB 382
119 D=1
120 HPRINT(5,19),"DOWN :
1ST":HPRINT(17,19),"
AND":HPRINT(22,19),"10"
121 HPRINT(26,19),"TO GO"
122 F2=.03
123 O=0
124 IF C<=0 THEN 353
125 GOSUB 324
126 Z2=1
127 Z3=.3
128 IF C<=F*T1 THEN 375
129 IF C<=0 THEN 353
130 IF L=0 THEN 131
131 HPRINT(5,17),"TIME TO GO"
132 HPRINT(18,17),INT(C/
60):HPRINT(22,17),"min,":HPRINT(27,17),C-
60*INT(C/60)
133 HPRINT(31,17),"sec"
134 HPRINT(5,14),"YOUR PLAY"
135 GOSUB 460
136 IF S=1 THEN 140
137 IF Q>8 THEN 134
138 M=Q
139 GOTO 142
140 Y=Q-8
141 IF ABS(Q-10.5)^2>3 THEN 134
142 C=INT(C-L*(5+23*RND(-TIMER)))
143 L=1
144 IF M=7 THEN 234
145 IF M=8 THEN 252
146 Y1=Y-1
147 REM
148 IF RND(-TIMER)>Z(M-1,Y1)+.025 THEN
174
149 A=2
150 GOSUB 386
151 IF M<4 THEN 158
152 HPRINT(ZD,ZF),"PASS COMPLETE..."
153 ZD=ZD+17:IF ZD>60 THEN ZD=3:ZF=ZF+
1
154 KG=RND(-TIMER)
155 IF KG < .33 THEN L=-1
156 IF KG > .33 AND KG < .65 THEN L=0
157 IF KG > .65 THEN L=+1
158 IF RND(-TIMER)+.035<F2 THEN 321
159 IF G>0 THEN 170
160 IF G=0 THEN i65
161 IF G+B+.395< RND(-TIMER) THEN 281
162 HPRINT(ZD,ZF),"LOSS OF":HPRINT(ZD+
8,ZF),-G
163 ZD=ZD+12:IF ZD>60 THEN ZD=3:ZF=ZF
+ 1
164 GOTO 217
165 HPRINT(ZD,ZF),"NO GAIN"
166 ZD=ZD+8:IF ZD>60 THEN ZD=3:ZF=ZF+1
167 Z9=2
168 GOSUB 202
169 GOTO 224
170 IF B+G>99 THEN 222
171 HPRINT(ZD,ZF),"GAIN
OF":HPRINT(65,13),"SWEEP"
502 HPRINT(65,14),"SCREEN PASS"
503 HPRINT(65,15),"LONG PASS"
504 HPRINT(65,16),"DRAW PLAY"
505 HPRINT(65,17),"PUNT"
506 HPRINT(65,18),"FIELDGOAL"
507 HPRINT(65,19),"TIME-OUT"
508 HPRINT(65,20),"YES"
509 HPRINT(65,21),"NO"
510 GOTO 515
511 REM PLAY MENU
512 IF S=1 THEN YZ=4444 ELSE IF S=2 TH
EN YZ=2222
513 I$="9999"
514 IF YZ=4444 THEN GOTO 481 ELSE IF Y
Z=2222 THEN GOTO 495
515 A$=INKEY$:IF A$="" THEN GOSUB 527:
GOSUB 529:GOTO 515
516 IF ASC(A$)=13 THEN GOSUB 527
517 IF ASC(A$)=13 THEN GOTO 551 ELSE G
OTO 518
518 IF A$="D" THEN YZ=4444:GOTO 480
519 IF A$="O" THEN YZ=2222:GOTO 495
520 REM UP-ARROW
521 IF ASC(A$)=94 THEN GOSUB 527 ELSE
GOTO 524
522 KN=KN-8:IF KN<90 THEN KN=KN+8:GOTO
510
523 REM DOWN-ARROW
524 IF ASC(A$)=10 THEN GOSUB 527 ELSE
GOTO 526
525 KN=KN+8:IF KN>178 THEN KN=KN-8:GOT
O 510
526 GOTO 510
527 DOT=HPOINT(510,KN-4)
528 HPAINT (510,KN),DOT,DOT:RETURN
529 HSET(510,KN-1):HSET(510,KN)::HSET(
510,KN+1)
530 HSET(509,KN+2):HSET(511,KN+2):HSET
(510,KN+2)
531 HPAINT(510,KN),(HPOINT(510,KN-4)),
DOT:RETURN
532 GOTO 510
533 REM POLL PLAY MEMU
534 REM OFFENSE
535 IF KN=92.95 THEN I$="30":REM RUN
536 IF KN=100.95 THEN I$="31"
537 IF KN=108.95 THEN I$="32"
538 IF KN=116.95 THEN I$="33"
539 IF KN=124.95 THEN I$="34"
540 IF KN=132.95 THEN I$="35"
541 IF KN=140.95 THEN I$="36"
542 IF KN=148.95 THEN I$="37"
543 IF KN=156.95 THEN I$="2"
544 IF KN=164.95 THEN I$="1"
545 IF KN=172.95 THEN I$="0"
546 IF I$="9999" THEN GOTO 511
547 HCLS:ZD=3:ZF=0
548 HPRINT(2,16),STRING$(71,"-")
549 GOTO 463
550 REM DEFENSE
551 IF YZ=2222 THEN GOTO 534
552 IF KN=92.95 THEN I$="4"
553 IF KN=100.95 THEN I$="5"
554 IF KN=108.95 THEN I$="6"
555 IF KN=116.95 THEN I$="7"
556 IF KN=124.95 THEN I$="8"
557 GOTO 543
558 PALETTE 0,18:PALETTE 8,0:STOP
559 HCLS:HPRINT(5,10),"SCORE: ME: ":HP
RINT(16,10),P(1)
560 HPRINT(21,10),"YOU: ":HPRINT(26,10
),P(2)
561 HPRINT(5,20),"Press any Key...."
562 EXEC &HA171
563 GOTO 558
564 END
```



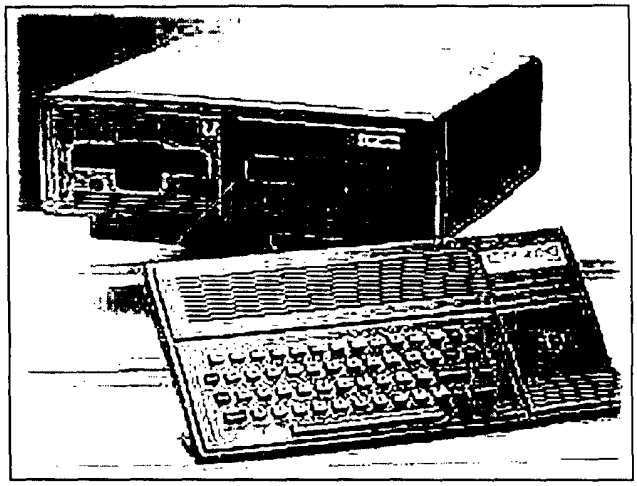
The Prologica CP-400 and CP-450

Andre Ballista

One of CoCo's South American cousins.

Editor: Here is something I received about one of the CoCo's South American cousins, the Prologica CP-400. This was a CoCo clone made in Brazil. The computer was programmed in English, but software was available in English or Portugese (the native tongue of most Brazillians). Proligica built the machines under license from Tandy. The ROM is 100% CoCo compatible. From the information provided below, it was 100% hardware compatible also.

There was some agreement between Tandy and a special United Nations educational program to provide computers (or rather the computer design) to several South American countries, mainly for educational purposes. A company in Mexico also made a 100% compatible CoCo 2 (and later a CoCo 3) clone. More on the Mexican CoCos later.



As far as I know, the CP-400 was totaly compatible with CoCo 2 -- at least at the software level. I don't know if the hardware would be. To try to answer this question, here is a small compilation of the technical manual that came with the computer.

Micropocessr: 6809E, 0.895 Mhz

Rear Conections (back view, left to right)

RF: TV signal Output. It should be connected to the anttena input of the TV set.

Channel: left-right key. Selects channel 3 or 4 at the TV

JD and JE: Joystick connection. This is a five pin conector. It looks like a MIDI connector. Pins position are:

```
   3
 2   4
1     5
```

1: left-right input for comparator
2: up-down input for comparator
3: Ground
4: fire button
5: VCC +5V

RS-232: Serial connector. This is a four pin connector. Also it looks like a MIDI connector. Pins position are:

```
 2 3
1   4
```

1: CD 3: GND
2: RX 4: TX

K7: Cassete Connector. 5 pin Connector. Also it looks like a MIDI connector. Pins are:

```
   3
 2 4
1   5
```

1: Remote 4: AUX
2: Ear 5: Remote
3: GND

Expansion Slot Connector:

40 pin, the connector is positioned in front of the computer. In the picture it's located at the right side of the computer, near the keyboard, under a small black dot (that actually is a small P - for Prologica, the manufacturer).

front view
39 37 35 33 ... 1
40 38 36 ... 2

1. - 12V (100mA)
2. + 12V (+300mA)
3. Halt
4. NMI
5. Reset
6. E
7. Q
8. Cart
9. +5V
10 11 ... 17 -> D0 D1 ... D7
18. R/W
19 20 ... 31 -> A0 A1 ... A12
32. CTS
33. GND
34. GND
35. SND
36. SCS
37. A13
38. A14
39. A15
40. SLENB

The CP-450 was the disk drive unit. This held two full height 5.25" single sided floppy drives, the disk controller, and a very large power supply. It was somewhat

CoNect

1629 South 61st Street
West Allis, WI 53214
(pulland@omnifest.uwm.edu)
414-328-4043

Fast232- 16550 does serial! Port speed to 115200bps, transfers up to 5000 cps. Addressable to four locations. With OS9 and Nitros9 drivers. **$79.95**

2nd Port Daughter Board - $45.00

OS9 lvl2 *lvl1 available!*

Level2 Bundle	$49.95
os9, b09, mvue, more! plug-n-go for 6809	
Dynacalc+Pgraph	$19.95
Profile	$19.95
TSEdit/Word+vi patch	$12.95
Epyx TriPak	$14.95
Koronis Rift, Rescue Fractulus, Rogue	
King's Quest 3	$9.95
Microscopic Mission	$4.95
Sub Battle Simulator	$4.95

Hardware

64K upgrd 2 or 4 chip	$5.95
512K upgrd *(used)* 0K	$24.95
512K	$44.95
decb1.1rom + manual	$12.95
mpi upgrd sat. board	$9.95
cable, cassette	$5.95
cable, printer	$5.95
cable, rs232 (100ft!)	$19.25
colr mouse (1 button)	$9.95
mono composite monitor *(used)*	$24.95
Orchestra90cc Pak	$12.95

DECB

Disk EDTASM *(used)*	$19.95
Disk ProFile *(used)*	$12.95
One on One	$7.95
Sands of Egypt	$7.95

ROMPaks too! (Inquire for titles)

Parts *(many more in stock!)*

1488/89	.75	68b09e	6.95
1723	1.95	6821a	3.95
1773	6.95	SALT	2.25
2764	2.95	74*6	.35
6802	3.50	74ls133	.42

I've also been working on some ***NEW*** hardware that may be available later. One of these items is a revision of my Expander idea that actually works on most CoCo 3's, not just the occasional "right" one. I'll keep everyone posted on any progress!

Check with me for complete disk drive systems, misc. hardware items, hardware repairs, and hard to find new and used CoCo software not listed!

Shipping & Handling $4 US, $6 Can/Mex, $10 World
offworld destinations please consult local Postmaster!

StrongWare

Box 361 Matthews, IN 46957 Phone 317-998-7558

CoCo 3 Software:

Soviet Bloc	$15
GEMS	$20
CopyCat	$5
HFE- HPrint Font Editor	$15

MM/1 Software:

Graphics Tools	$25
Starter Pak	$15
BShow	$5
CopyCat	$10
Painter	$35

ADVERTISER'S INDEX

What are you waiting for?

Get your friends to subscribe to the only magazine that still supports the Tandy Color Computer... *"the world of 68' micros"*!

The more people who want the support, the longer it will be here!