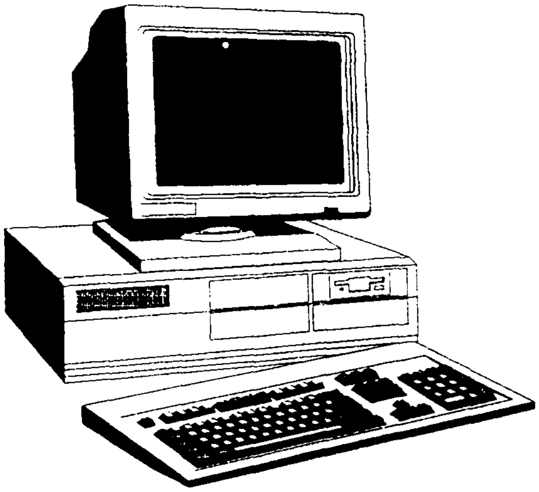
OS-9 Level Two

and the

Tandy Color Computer-3



written by

Rodger Alexander and Scott Honaker

***Hardware Upgrades* compiled and edited by Rodger Alexander**

OS-9

Level Two

and the

Color Computer III

by Rodger Alexander and Scott Honaker

OS-9 Level Two Tutorial

Five Chapters of explanations, instructions, diagrams and practical *Homework Sessions*. Each chapter guides your from a novice to an advanced "hacker".

Hardware Upgrades for the Color Computer-3

A collection of 10 upgrades to improve the performance of the Color Computer-3. Complete instructions, diagrams, and parts list.

OS-9 Level Two Tutorial and Upgrade Disk

Side one contains all of the files described in Chapter 5 of the Tutorial. Side two contains software upgrades to OS-9 Level Two (See Appendix D) and Driver/Descriptors and IPatch files required by some of the Hardware Upgrades presented in this Book.

OS-9 Users Forum, 3404 Illinois Lane, Bellingham, WA 98226. (206) 734-5806.

Table of Contents

OS-9 Level Two Tutorial

Chapter 1

Chapter 2

Chapter 3

Chapter 4

Chapter 5

Chapter 1
Getting Around and Using OS-9

OS-9 is a Disk Operating System for computers using the 6809 or 680xx CPUs (Central Processing Unit). There is a recent relative called OS-9000 which runs on Intel 80386 and 80486 microprocessors as well as the Motorola 680xx family. Microware and Motorola developed this OS-9 to emulate the UNIX Operating System on less expensive hardware. Unix was then and normally is now used on large computer systems such as the Digital Equipment Corporation PDP-11 and VAX computers. The major difference between such systems as the VAX family and most OS-9 systems (especially 6809 systems) is that they are generally smaller and slower. OS-9 systems store most of their commands on disk rather than permanently in memory. As a result most OS-9 systems are very disk intensive. A good understanding of how OS-9 looks at a disk and stores data is the basis for effective OS-9 use.

OS-9 stores most of its instructions in the "execution" directory, normally /DD/CMDS. The executable binary files found in that directory permit the system to provide various services. So, in order for OS-9 to report the contents of a directory to the screen it must first find an executable file (called .../DIR), which supplies the necessary instructions.

Example: **DIR /D0**

Seems simple enough; you type in the command DIR /D0, and OS-9 first looks into its memory for the command "DIR" in order to carry out your command. If OS-9 cannot find "DIR" in memory, it then looks in the execution directory (usually /DD/CMDS). When "DIR" is found, it is loaded into memory and run. "DIR" tells OS-9 to list out the file names stored in the "root" directory on drive zero (/D0). Notice that "D0" had a slash "/" in front of it, which tells OS-9 that "D0" is a "device". Since no directory name was specified after the device name, DIR and OS-9 look at the root directory on floppy drive zero.

MS-DOS and OS-9 (and most other command line oriented operating systems) are somewhat similar in their command syntax and directory structures. In fact, many OS-9 commands are parallel to those in MS-DOS. Unfortunately, however, computers are unforgiving when it comes to proper syntax and it is "path names" which give most OS-9 users difficulties.

Path names:

If you typed in the above example "DIR /D0" on your CoCo, you should see something like:

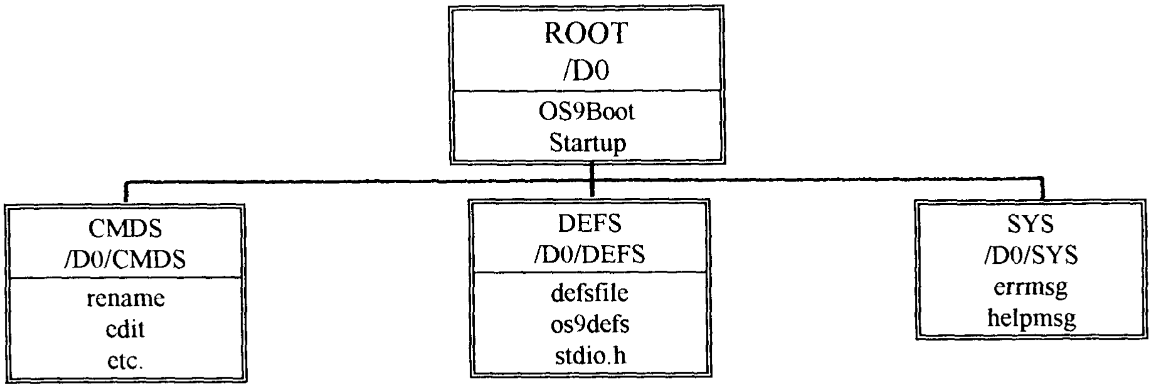
```
os9boot     CMDS     SYS     startup
```

What you are looking at is properly called the "root" directory. It is the initial working directory most OS-9 systems use when booted. You may see other directories and files on your disk too. "OS9boot" and "startup" are files on the root directory, while "CMDS" and "SYS" are subdirectories within the root directory on this drive. Notice that the directory names are displayed in upper case, while the two file names are displayed in lower case. This is an OS-9 convention used to distinguish quickly between filenames and directory names. OS-9 itself doesn't care. If you are ever confused, a DIR e /D0 will always display a complete directory. The attributes displayed with each entry will indicate whether it is a file or directory.

To see what is in the CMDS subdirectory or the SYS subdirectory, simply add their name to the original command line: DIR /D0/SYS will give you a listing to the screen of the contents of the SYS directory, while DIR /D0/CMDS will list the contents of the CMDS directory.

OS-9 Directory Structure

(with typical files)



Homework Session

Getting Started:

To make any progress, we will all have to have a basic understanding of the OS-9 environment, also known as the "OS-9 shell". And the easiest way to accomplish this is to actually practice some of the major command functions in OS-9. We'll start by formatting a new disk and making a backup of your original OS-9 master disk, then we will create a directory, copy files to the directory and even delete a few files. So turn on the power, put your OS-9 boot disk in your CoCo drive (after putting a write protect tab on it) and Enter DOS. When OS9 "boots-up" enter the date at the prompt.

At the OS-9 prompt (**OS9:**):

1. Enter: **FORMAT /D0**
   The format command will ask you whether you are ready to format a disk in drive 0. Place a **blank disk** in your drive and Enter: **Y**.

2. Then the format command will prompt you for the name of the disk:
   Enter: **OS9_Level2**

3. Formatting a new disk takes a few minutes. When it is finished, the OS9: prompt will reappear on the screen, and a new OS-9 disk will be in the drive. BUT, it's blank, there's nothing on it. Use the DIR command to check the contents of the new disk.

4. Place your original OS-9 System disk back in the drive

   NOTE: Be absolutely certain you have a "write protect" tab on your master disk

   Enter: **BACKUP /D0**
   The backup command will prompt you twice before continuing. Make sure that you observe the "source disk" and "destination disk" instructions using your freshly formatted disk as the "destination disk".

   When the OS-9 prompt returns, your "new" OS9_Level2 disk will be in the drive. You can use the directory command to look at your new disk.

5. Enter: **CHD /D0**
   Enter: **CHX /D0/CMDS**
   First we **CH**anged the current **D**ata directory setting to the "root" directory on the new disk, then we **CH**anged the current e**X**ecution directory setting to the CMDS directory on our new disk.

OS-9 notices disk changes and will not be able to find either execution or data directories on a newly inserted disk even if the same directory names are on the new disk! You **MUST** reset the Data and eXecution directory names every time you switch disks.

Quick review........
We have already used the "format" command to initialize (i.e. DSKINI in RSDOS terms) a new disk, then we used the "backup" command to copy all the files and directories from the original master OS-9 disk to our new disk, and then we told OS-9 to change it's directory references to the new disk using the CHD (CHange Data) and CHX (CHange eXecution) directory commands.

Now, Let's Get Creative:

6. Enter: **MAKDIR /D0/MYDIR**
   After the OS-9 prompt returns, do a DIR of /D0 to see where OS-9 placed your new directory (Remember: DIR /D0).

7. Enter: **COPY /D0/CMDS/build /D0/MYDIR/build**
   After the OS-9 prompt returns, do a DIR of your new directory to see if the BUILD file actually did copy over from the CMDS directory (DIR /D0/MYDIR).

8. Enter: **COPY /D0/CMDS/edit /D0/MYDIR/edit**
   After the OS-9 prompt returns, do a DIR of your new directory to see if the EDIT file copied over.

9. Enter: **COPY /D0/CMDS/del /D0/MYDIR/del**
   After the OS-9 prompt returns, do a DIR of your new directory to see if the DEL file copied over.

Review break........
So far you created a new directory, did a DIR of the "root" directory to see if your directory was really created, and then proceeded to populate your new directory by copying files "build", "edit" and "del" from the CMDS directory. **........BACK TO WORK!**

10. Enter: **PWD**
    OS-9 will display the Present Working Directory. Also known as the Data directory. Should be "/D0", the "root" directory.

11. Enter: **PXD**
    OS-9 will display the Present eXecution Directory. Should be "/D0/CMDS".

12. Enter: **CHD /D0/MYDIR**
    Enter: **PWD**
    First we CHanged the Data directory to MYDIR and then we checked to see if MYDIR is now the Present Working Directory.

13. Enter: **CHX /D0/MYDIR**
    Enter: **PXD**
    OS-9 should now show that /D0/MYDIR is now also your execution directory.

Take another break.......
Now we have made the new directory the default working (or data) directory. We've also told OS-9 to make the new directory the execution directory, which is pretty risky considering that there are only three command/execution files available on our new directory. **........BREAK OVER!**

14. Enter: **BUILD lesson1**
    Note the lowercase spelling of "lesson1". Remember we keep directory names in uppercase and file names in lowercase so that we can recognize the differences when we view complex directory listings that contain both directory and file names.
    After entering "BUILD lesson1", the OS-9 prompt does not return, but instead a question mark "?" is displayed. BUILD is now looking for input.

15. At the "?" prompt, type in instruction #6 from above.
    ? **MAKDIR /D0/MYDIR**

16. At the next "?" prompt, type in instruction #7 and then #8, continuing until you have entered instruction lines 6-13 from above.

17. Enter: <ENTER> (Press the Enter Key only)
    This will exit the "build" utility and return you to the OS-9 prompt.

18. Enter: **DIR** -or- **DIR /D0/MYDIR** -or- **DIR e /D0/MYDIR**

19. Enter: **DEL** edit -or- **DEL /D0/MYDIR/edit**

20. Enter: **DEL build** -or- **DEL /D0/MYDIR/build**

21. Enter: **DEL del** -or- **DEL /D0/MYDIR/del**

22. Enter: **CHX /D0/CMDS**
    We are now telling OS-9 to make CMDS the execution directory instead of MYDIR. Might as well, there's nothing left in MYDIR to execute except the "Lesson1" file, which is a text file, not an executable command file.

23. Enter: **LIST Lesson1** -or- **LIST /D0/MYDIR/Lesson1**
    This should print our text file to the screen. If you have a printer connected to your CoCo operating at 600 baud, you could change the above line to:
    LIST Lesson1 >/p. This will send the listing to the printer.

That's it.....you're done!!!!

Wow! You've created a new directory using the MAKDIR command, copied files from one directory to another with the COPY command. You checked your directory status using the PWD and PXD commands. You changed your execution and working/data directories to MYDIR and created a text file called "Lesson1" using the BUILD command, and finally deleted all the command files in your directory using the DEL command and changed your execution directory reference back to the CMDS directory.

For Extra Credit.....(Are you kidding *@#$%!!) Build another text file called "Lesson1B". Include the remaining instruction lines 14 through 23.

Chapter 2

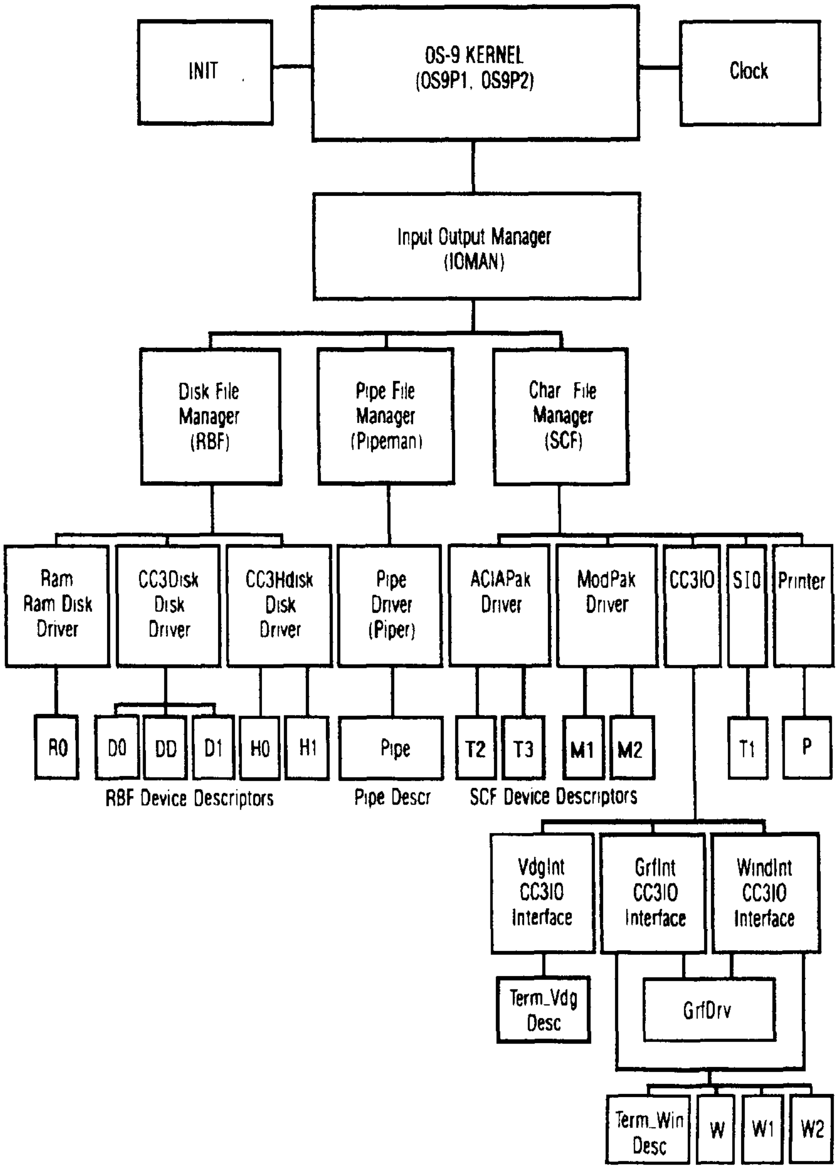
Configuring OS-9

OS-9 the Modular System

OS-9 is an amazingly simple modular system. Every module loaded into memory at boot time has its own function to make the operating system function properly and provide all the necessary functions. The major advantages of this type of system are easy configuration and expandibility. In most cases, the fact that the modules are "reentrant" (meaning the same piece of code an be run by many processes simultaneously) create a very compact system.

The complete OS-9 system is shown below with all the modules that make up a complete system. The modules at the top are required for any OS-9 system but the modules toward the bottom can be removed an more added for custom configurations.

I/O System Modules (see Technical Reference 1-1)



The I/O Manager (IOMAN) controls all communication with OS-9 at a very low level. The level above this breaks communications management into three different types; RBF - Random Block File Manager (random access devices - floppy/hard drives, RAM drives, CD-ROM drives, etc), SCF - Sequential Character File Manager (sequential access devices - serial ports, parallel ports, etc) and Pipe Communications Manager (used for passing information between processes, like shell commands).

Above the manager level come the drivers. The device drivers contain the program code necessary to operate any particular device. Since there needs to be a certain amount of information for these driver to access the devices (like their memory address, size, speed, etc), there is another layer called device descriptors. The descriptors hold all the device specific information. As new devices are added all that is necessary to use them is to get the new device driver and descriptor and install it into OS-9.

ROM BASIC is as compact as can be made for a single tasking system but multitasking adds a level of complexity. Since ROM BASIC is in ROM it cannot be changed or expanded easily. To add a hard drive, larger floppies, a RAM drive or other devices (like a speech cartridge) some type of patch has to be made to make it accessible. Typically some type of driver program has to be loaded to replace the ROMs each time. More sophisticated systems either add ROMs (like the Radio Shack Speech/Sound cartridge) or replace the ones in the computer.

Software has to be written to expressly use the majority of this new hardware under Radio Shack's ROM BASIC. This isn't true under OS-9. OS-9, through its modularity, offers what's known as "device independence". This means that once a driver is installed for the new hardware any piece of software can use it. As an example, once the Speech Pack driver is loaded anything can use it as an output device just by asking for it (directories can even be sent to it!).

Making OS-9 Boot Disks

The simplest method to make a bootable OS-9 disk is to use the BACKUP command (you'll want to make backups of your masters before you make any modifications). One thing to remember is that all the utilities supplied with OS-9 require that you have formatted disks before you start.

It's possible to create a clean, bootable disk using the COBBLER command on a formatted disk. COBBLER copies your current boot to a new disk (it won't create a CMDS directory however, so you'll need to do that and copy the *shell* and *grfdrv* modules over). You can use this for a disk you want to completely determine the commands that are available (special boot disks for kids that don't have DEL, ATTR, etc).

The simplest way to create a custom boot disk is with the CONFIG command. CONFIG can be used with the second OS-9 master (the Config disk) to create a whole new system. Chapter 7 in the *Getting Started* section of the OS-9 manual contains information on running the CONFIG utility.

The CONFIG utility allows you to choose the modules you want from a list. You can add modules to this list by placing them in the MODULES directory on the Config disk and CONFIG will scan for them when it starts. This allows you to keep a custom config disk with the modules used in your system. You can even place help description in the directory below so that you remember what each module does.

The final method to create a system disk is with the OS9GEN command. OS9GEN needs to be run in the directory where the desired modules are located. Also, an input file is needed listing those modules. The input file may be created using a simple text editor. As an example, you could create a file called *bootlist* with the names of all the modules you want on separate lines. You can then run the command **OS9GEN /D1 <bootlist** from the directory with the modules to create a custom boot file on /D1.

Copying Commands and Data

Since every boot disk requires a CMDS directory with *shell* and *grfdrv* in it, you will need to get these and your other command files copied over from you original system disk. The CONFIG utility will copy over a set of command files for you, but that still leaves data files behind that you might want that are not in the CMDS directory.

There is a command called DSAVE supplied with OS-9 that will copy these files all at once. There are other utilities available in the public domain that are easier to use but this one is standard OS-9 and it's important that it at least be explained (this means I don't like it but it does work).

The DSAVE command does not directly do the copy, it actually creates a procedure file which can be edited and run later. To directly copy /D0 to /D1 you can pipe the output to a shell with the following commands:

```
        CHD /D1
        DSAVE /D0 ! SHELL -x
```

This will tell DSAVE to create commands to copy everything from /D0 (including all subdirectories except the OS9Boot file) and send it to the shell (the **-x** tells OS-9 not to stop if an error occurs). The shell will run the commands that will place the files in the current directory (/D1). There are several options for DSAVE (to copy subdirectories or not, copy the OS9Boot file, verify writes, etc) and more examples in the commands portion of the manual.

OS-9 Installable Device Descriptors (on the Config disk)

D0_35S	Floppy Disk Drive /D0, single sided, 35 cylinders
D1_35S	Floppy Disk Drive /D1, single sided, 35 cylinders
D2_35S	Floppy Disk Drive /D2, single sided, 35 cylinders
D3_35S	Floppy Disk Drive /D3, single sided, 35 cylinders
DDD0_35S	Default Disk Drive /D0, single sided, 35 cylinders
D0_40D	Floppy Disk Drive /D0, double sided, 40 cylinders
D1_40D	Floppy Disk Drive /D1, double sided, 40 cylinders
D2_40D	Floppy Disk Drive /D2, double sided, 40 cylinders
DDD0_40D	Default Disk Drive /D0, double sided, 40 cylinders
D2_80D	Floppy Disk Drive /D2, double sided, 80 cylinders
D3_80D	Floppy Disk Drive /D3, double sided, 80 cylinders

P	Printer using the RS-232 serial port
TERM	The CoCo keyboard and display
T1	Terminal number one from the internal RS-232 port
T2	Terminal two using the Radio Shack RS-232 program pack
T3	Terminal three using another Radio Shack RS-232 program pack
M1	Support for the Radio Shack modem pack
M2	Support for a second Radio Shack modem pack
PIPE	Support for pipes within OS-9
W	Generic window descriptor
W1	Window device number 1
W2	Window device number 2
W3	Window device number 3
W4	Window device number 4
W5	Window device number 5
W6	Window device number 6
W7	Window device number 7

Homework Session

In chapter one we explored the OS-9 environment and learned about path names, default or data directories and execution directories and how to change directories. We also used some of the basic OS-9 commands: **FORMAT, BACKUP, MAKDIR, COPY, DIR, DEL and LIST**.

This month we are going to take your original OS-9 (35-track single sided) system disk and create your own "CUSTOMIZED" system disk, designed specifically for your CoCo setup. But first, Rule #1 must be observed: "**ALWAYS MAKE BACKUPS**". So, refer back to last month's lesson and boot up OS-9, FORMAT a new blank disk, and BACKUP a copy of your original OS-9 SYSTEM master disk. From now on we will only use the "back-up" copy of the original. However, DO NOT put a write protect tab on your back-up copy of the BOOT/CONFIG/ BASIC09 Disk.

> **NOTE:** I have been assuming that everyone is now using OS-9 Level Two (or Level One version 2.00). If you are running OS-9 Level One version 1.xx then this lesson is not going to be of much help since the CONFIG utility is not available. It would be advisable to locate a copy of the version 2.0 upgrade to Level I. Also, if your CoCo screen turns on with "Disk Extended Basic 1.0" (or 2.0) then enter RUN "*.bas" in order to boot up OS-9. OS-9 Level Two will only run on a Color Computer -3.

CONFIG:

Boot up OS-9 using the RS Extended Basic DOS command. When OS-9 comes up on the screen, you will be asked to enter the current date and time. You can skip this entry by hitting enter, BUT DON'T! OS-9 records the date on all files or directories you create, so always complete the date entry when booting OS-9.

Follow the instructions in last month's lesson to format a new disk. The CONFIG utility will create a new system disk onto this new disk when we are finished.

Open your OS-9 owner's manual to page 7-1 and follow the step-by-step instructions for using the CONFIG utility.

1. Remove your system disk from your drive (/D0) and replace it with the *OS-9 LEVEL TWO OPERATING SYSTEM -- BOOT/CONFIG/BASIC09* disk.
   Enter: **CHX /D0/CMDS**
   Enter: **CHD /D0**
   Enter: **CONFIG**

2. You will be prompted for the number of disk drives your CoCo is running: ONE DRIVE ONLY or TWO OR MORE DRIVES Well? What's your answer?

   If you answer "**2**", you will then be prompted for the name of the source disk. This is where most people make their big mistake.

3. CONFIG prompts: "ENTER NAME OF SOURCE DISK:"
   Enter: **/D0**

   The CONFIG utility is not actually asking for the "name" of a floppy disk, but the "location" of the source (Boot/Config/Basic09) disk. Which disk drive you are running Config from? Probably "/D0". And then, where is your freshly formated "destination" disk? Probably "/D1".

   CONFIG prompts: "ENTER NAME OF DEST. DISK:"
   Enter: **/D1**

4. CONFIG will then print to the screen a list of device options for you to select from. You select which device you want by using the arrow key to point to the desired device and then press "S" to select or unselect your choices:

   P is the printer driver which is necessary for OS-9 to communicate to your printer. **Choose this one**

   T1 is the "Bit Banger" RS-232 port on the back of the CoCo and is not reliable for serial communications with Level Two. **Don't choose this one**

   T2 is for the Deluxe RS-232 RomPak that Radio Shack used to sell. If you have the RS-232 RomPak then you need T2. If you purchase another vendor's RS-232 card, they will provide you with a T2 driver module that you can add to you boot file using OS9Gen. (We'll get to this later.)

M1 is the Plug-in Modem Pak that Radio Shack has discontinued and selling for $10 (if you can find one).

PIPE is a utility that permits OS-9 to take the output from one file to the input of another file. **Choose this one**

D0 CONFIG provides disk drive options for single sided 35 track drives: D0_35S, D1_35S, D2_35S, D3_35S DD_35S; 40 track double sided drives: D0_40D, D1_40D, D2_40D, DDD0_40D; and 80 track double sided drives: D0_80D, D1_80D, D2_80D

Check to see what type of disk drives you have and decide which one you want to be Drive 0, Drive 1, etc. The FD-502 floppy disk drive sold by Radio Shack is a 40 track double sided drive, while the FD-501 is a single sided 40 track drive.

DDD0 The "DD" drive is the Default Drive and is actually the same as /D0, but some Level Two software calls for the /DD drive, so make sure to include the "DD" driver that matches your "D0" driver.

When you are finished selecting your devices:
Enter **D**

5. CONFIGuring your terminal descriptor depends on your personal preference and the type of monitor you are using. If you are viewing your CoCo display on a Television than you will want OS-9 to come up with the 32 column TERM_VDG screen. If you have a monochrome composite monitor (amber or green) OR if you happen to be lucky enough to have an analog RGB colored monitor (Tandy CM-8 or Magnavox 8CM515 or 1CM135), you will want the TERM_WIN screen

   If you selected #2 (TERM_WIN) then you will also be prompted for which window drivers you want included on your system disk. SELECT ALL 8 WINDOWS.

   Enter: **D** (DONE)

6. The next thing CONFIG prompts you for is the clock frequency. No help needed here.
   Enter: **1** (60 Hz (AMERICAN POWER)

7. CONFIG is ready to generate your customized system disk:
   Enter: <spacebar>

8. CONFIG now prompts you for the commands you want on your disk.
   Enter: **N** (We'll make our own, thank you.)

THAT'S IT!..............well, not quite.

Quick Review....We have created/generated a customized OS- 9 Levely Two disk by using the CONFIG utility. After correctly "name"-ing the Source and Destination disks, we selected the device descriptors and drivers that we wanted on our new system disk: RS-232, Printer, Pipe, and the proper disk drivers to match the physical description of the drives that we are going to use. We also chose the terminal descriptor dependent upon the type of terminal we are going to use, and if we chose the TERM_WIN, we then selected to include all of the window drivers. Finally, we had CONFIG install the 60Hz clock module and requested NO command files to be included.

DSAVE:

9. OS-9 will not boot unless two files, SHELL and GRFDRV, are in the CMDS directory. So we will make our own CMDS directory and copy those two files over from the system master to our new CMDS directory.

   Replace the CONFIG disk with the original System Master disk in drive /D0
   Enter: **CHX /D0/CMDS**
   Enter: **CHD /D0**
   Enter: **DSAVE /D0 /D1 ! SHELL**

   Now sit back and watch the power of OS-9 at work. By using DSAVE instead of having CONFIG generate a CMDS directory you will be saving yourself a lot of work. DSAVE not only makes a CMDS directory and copies all of the files from /D0 to /D1, it also makes the SYS directory and copies all of the files on the system master to your new disk, including the startup file and the window configuration files (window.t38s, window.t80s, window.glr4).

   Place your new customized system disk in /D0 and reboot your CoCo. Enter the date and do a DIR of your new disk to see if everything is there.

That's it!......we're done.....really!!!!!

OPTIONS: Take a break and then try these.

OS-9 provides three display type options using the MONTYPE command.

1. If you are using an analog RGB colored monitor...
   Enter: **MONTYPE r**

2. If you are using a TV or a composite color monitor...
   Enter: **MONTYPE c**

3. If you are using a monochrome (green or amber) monitor...
   Enter: **MONTYPE m**

4. If you configured your system disk with TERM_WIN and included W7, we can initialize an 80 column, 24 row text window operating under it's own shell using the INIZ command.
   Enter: **INIZ w7**
   Enter: **SHELL i=/w7&**

Now lets put the above four commands together and MERGE them to the startup file so that the correct MONTYPE and W7 are automatically setup as part of the boot-up.

1. Enter: **RENAME startup startup.old**
2. Enter: **BUILD startup.mods**

From the BUILD's question mark (?) prompt, type in the following commands...

3. Enter: **MONTYPE r** *or* **MONTYPE c** *or* **MONTYPE m**
4. Enter: **INIZ w7**
5. Enter: **SHELL i=/w7&**
6. Enter: <enter> * to end BUILD
7. Enter: **MERGE startup.old startup.mods > startup**
8. Enter: **LIST startup** * check listing
9. Enter: **DEL startup.old startup.mods**

Chapter 3

Customizing OS-9

Getting the most from your hardware

Customizing Your Shell

When you type a command at the OS-9 prompt, several steps are performed to run this module. OS-9 first searches it's module directory in memory then the current execution directory. If the module isn't currently in memory, OS-9 will load it. Once in memory OS-9 will update the link count stored in the module header (in memory) and run the program.

The *shell* file you see in your commands (CMDS) directory is actually a number of command files merged together. Since the *shell* is loaded into memory at boot time, there are advantages and disadvantages to this. The major advantage is speed. Any module you run that is part of the *shell* will already be in memory. Because of this, it will also take up (possibly) valuable memory space.

Radio Shack has chosen a fair selection of commands to include in the shell (see Table 1 below). Because these commands are included in the shell, it is not necessary to keep a separate copy of these commands in your commands directory. It is possible to add commands to this by merging them in using the MERGE command. There are no standard utilities supplied with OS-9 to break these modules into their components. D. P. Johnson supplies a utility called MODBUSTER that will break these files into modules as well as several other public domain utilities.

Commands merged into the OS-9 SHELL (as shipped from Tandy)

SHELL	COPY	DATE	DEINIZ
DEL	DIR	DISPLAY	ECHO
INIZ	LINK	LIST	LOAD
MDIR	MERGE	MFREE	PROCS
RENAME	SETIME	TMODE	UNLINK

Table I

The addition of more commands to the shell is actually very simple. If all the commands are located in your commands (CMDS) directory, change the data directory to default to the CMDS directory. **Merge** the *shell* file plus the desired aditional files, sending the output to a temporary file. The attributes then need to be changed on this temporary file so that OS-9 knows it can be executed (that it's code not just data). The the old *shell* needs to be deleted or renamed and the temporary file renamed to *shell.*

As an example to merge the DEBUG command into the shell use the following commands (assuming you're running from a floppy):

```
CHX /D0/CMDS
CHD /D0/CMDS
MERGE shell debug >newshell
ATTR newshell e pe
DEL shell
RENAME newshell shell
```

To verify that the DEBUG command file this has been added, use the IDENT command. IDENT will display information about the type and integrity of each module in the file. Any number of modules can be merged into a single file (as long as they're executable). The merging of modules gives programmers a way to keep their software modular yet still distribute it as one file. The only way to determine the exact contents of an executable file is to use the IDENT command.

Since OS-9 Level II has to allocate memory in 8K sections, this merging is also a way to efficiently use memory. The standard shell is just under 8K so it only requires one 8K block to load. Since there are 20 modules total, if each were loaded separately they would require 160K of memory (more than a standard CoCo 3)! This is not true of Level I which can allocate 256 byte pages. The block size is a function of the maximum amount of RAM in the system (the more RAM, the larger the pages).

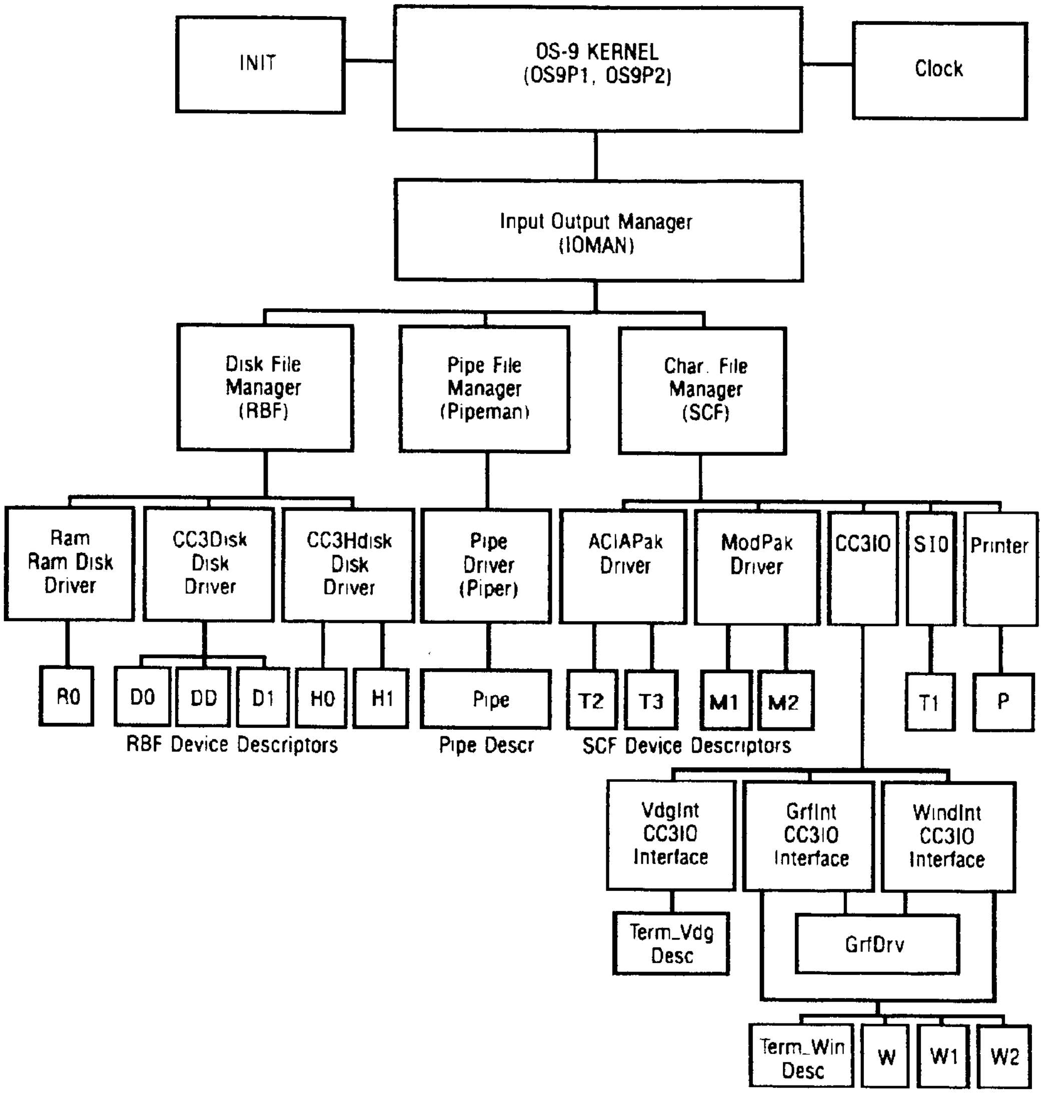
When you are merging files, always attempt to keep the size as close to a multiple of 8K as possible for maximum efficiency. Also be sure to merge any files that will need to be loaded together, so they won't take up separate 8K blocks. OS-9 is smart enough not to load the same module twice and if it sees another one coming in, it will just ignore it if the version number of the new one is the same or less than the one currently loaded in memory.

The IDENT command will return a significant amount of information about each module. most of this information comes from the module header. Each executable module under OS-9 has a very specific format (shown below). This consistency in all executable files allows the operating system to maintain order and control over the system.

Relative Address	Use	
	Sync Bytes ($87CD)	
$00	Module Size (bytes)	
$02	Module Name Offset	
$04		
$06	Type	Language
$07	Attributes	Revision Number
$08	Header Parity Check	
$09	Execution Offset	
$0B	Permanent Storage Size	
$0D	(Additional optional header extensions located here) Module Body object code, constants, and so on	
	CRC Check Value	

Working with Drivers and Descriptors

We saw in chapter 2 that the OS-9 system is comprised of several layers of modules and that each provide a specific level of functions for OS-9. Although there are patches available for some of the system modules, by far the vast majority of modifications will be made to the device descriptors or the driver modules. These modules are shown as the bottom two levels of the chart on the next page.



OS-9 is very protective of its system and tries whenever possible to preserve it's integrity. To insure this, every module loaded into the OS-9 system has a CRC value at the beginning that can be verified against the module's contents. If the CRC value doesn't match, the system won't load it. This security adds additional complexity whenever a modification is made because without proper updates, OS-9 will assume the module is corrupted and not load it.

One of the most common modifications is to increase the step rate of the drive. Because Radio Shack had to keep their system as generic as possible, they choose the slowest step rate for the floppy drive to insure compatibility with all systems. All the descriptors shipped with OS-9 use this 30 millisecond step rate even though most current drives will handle the 6 millisecond maximum. This modification can obviously improve floppy drive performance significantly.

After a modification is made, it is necessary to "validate" the module. That means it has to recalculate the new CRC value and replace the old one. There are two utilities supplied with OS-9 to accomplish this task; DEBUG and MODPATCH. The DEBUG utility is robust but because of that it tends to be more difficult to use. I will show how to make this modification using the MODPATCH utility. Keep in mind the process is the same for DEBUG. There is an entire section of the manual devoted to the DEBUG features and it is a good idea to take a look at these to get familiar with them.

MODPATCH accepts commands from a file and executes them. To effectively use MODPATCH, create a script file for MODPATCH to execute. Use the BUILD command (or EDIT) to create a file with the following contents:

```
L d0
C 14 00 03
V
L d1
C 14 00 03
V
```

After this, assuming the script file is called "FastPatch", execute the following command from the OS-9 prompt:

```
MODPATCH FastPatch
```

At this point the drives should step much faster. You can either use COBBLER to create a new boot disk using these new descriptors or use the SAVE utility to write the new ones to disk. If you write them to disk, you can update the descriptors supplied with OS-9 on the CONFIG disk. Once this disk is updated, you can use the new descriptors to build disks in the future.

Here are a couple of more patches that can be made to the system.

The following patch will upgrade most Level I modules to Level II. To do this you need to change the byte at offset 14 (decimal) from FF to 07. To make this patch to the H0 driver for example:

```
L h0
C 0E FF 07
V
```

If your keyboard echos an extra lowercase letter when you start a new sentence, the following patch will delay the start repeat time and the repeat rate:

```
L cc3io
C 7e 1e 3e
C 86 03 06
V
```

When testing modules or after installing a minimal system it is actually possible to load them after the boot. If for example, you have a new Speech/Sound Cartridge it's possible to LOAD the driver and descriptor. Assuming the name of the descriptor is SSC you can use INIZ SSC to initialize the new device support. That will allow you to use it as if it was loaded at boot time. One thing to keep in mind is that each module loaded will still require OS-9 to allocate 8K each.

Homework Session

In chapter 2 we used the Config utility to customize OS-9 to our own individual computer set-up....that is, we customized OS-9 to the extent that Tandy allowed us. There are alot of 3rd party drivers and descriptors that we will want to incorporate into our personal OS-9 system, and there are other ways to implement them besides using Config.

Modifying Modules:

It's understandable that the disk drive descriptors provided with OS-9 Level Two are "downward campatible" so that someone with an original Radio Shack full heighth, 35 track, single sided, 20 millesecond stepping rate disk drive (whew!) will be able to successfully use OS-9 on his computer. That's wonderful! But what about the 95% of us who have newer disk drives capable of 40 tracks, double sided, 6 millesecond stepping rate, etc.?

Remember in our chapter 2 Homework Session example we found that Radio Shack did provide disk drive descriptors for 35 track single sided, and 40 and 80 track double sided drives. BUT those descriptors still have 30 millesecond stepping rates.....VERY SLOW.

TWO SOLUTIONS: Tandy to the rescue: About one year after OS-9 Level Two was released, Tandy released OS-9 Level Two Development System. This included two utilities, MODPATCH and DEBUG, that we can use to solve our disk drive updating problem.

MODPATCH:

To change the stepping rate of /D0 , we need to modify byte 14 of the D0 drive descriptor. In order to use MODPATCH to accomplish this modification, we first need to create a Procedure file for MODPATCH to execute. Use BUILD or any text editor to create the following file:

1. Enter: **BUILD D0_Patch**
   Wait for the BUILD "?" prompt....

```
Enter: l d0            * <---Link D0 drive descriptor
Enter: c 14 00 03      * <---00=20ms; 03=6ms stepping rates
Enter: v               * <---Verify new file length and unlink
```

2. Enter: **MODPATCH D0_Patch**

That's it! Disk Drive 0 will now purrrr along at a respectable 6 milliseconds. If you have a second drive (/D1), repeat step 1 and 2 above changing the first line in the Procedure file to read "**l D1**". Also VERY IMPORTANT! Make sure you MODPATCH "DD" as well, again substituting "**l DD**" in the first line of the Procedure file.

DEBUG:

The Level-II DEBUG utility is only available as part of the OS-9 Level Two Development System. Fortunately the Level-I DEBUG utility will also work on Level-II. Install DEBUG in your CMDS directory, then carefully type in the following DEBUG command lines:

```
Enter: DEBUG
Enter: l d0            * <--- Link to the D0 descriptor
Enter: . .+14          * <--- space between the periods
Enter: =03             * <--- change byte 14 to 03
Enter: q               * <--- Quit debug
```

Wasn't that easy! Now you have two ways to modify memory modules. Only problem is, when you turn off the power, you loose those wonderful patches. The modifications were made to the descriptors in memory, not to the descriptors in the OS9Boot file on the disk.

COBBLER:

COBBLER is a very handy utility in the CMDS directory that will copy your OS9Boot file from memory (not disk) to a freshly formatted disk. OS9GEN is a similar utility in the CMDS directory, but it copies your OS9Boot file from your system disk (not memory) to a fresly formatted disk.

Read the above paragraph 2 or 3 times to make sure you understand the difference between COBBLER and OS9GEN.

In order to keep our modified drive descriptors from being destroyed when we turn the computer off, we need to create a new System Disk copying our modified OS9Boot file from memory to a freshly formatted disk. It's almost too simple! If you have two disk drives, FORMAT a blank disk in /D1. Now type in the following OS9 command:

Enter: **COBBLER /d1**

When the OS9 prompt returns, your disk in drive 1 will have an OS9Boot file that includes your modified descriptors. Use DSAVE to copy all of your directories and files from drive 0 to drive 1: (Make sure your data directory is set for /d0)

Enter: **DSAVE /d0 /d1 ! shell**

When DSAVE is done, remove your old system disk from drive 0 and replace it with your newly created system disk. Reboot the computer and listen for your disk drive stepping rate to suddenly take off half way through the boot-up.

> Take a break.......
> In this Homework Session we have modified our disk drive descriptors using MODPATCH to change specific "bytes". We used Procedure files to instruct MODPATCH. As an alternate we used DEBUG to accomplish the same thing. Then we used COBBLER to generate a new OS9Boot file (from memory) on a freshly formatted disk. FAST and EASY!

Let's take another approach:

Wouldn't you really like to have your OS9 boot up directly to an 80 column screen rather than the default 32 column screen that Tandy provides? Of course you would!

Let's use BUILD again, or any text editor to write another MODPATCH procedure file.

```
TERM_Patch
l term
c 2c 28 50
c 30 01 02
c 33 02 00
c 34 03 02
c 35 03 02
v
```

> **NOTE:** This patch applies to the TERM_win module, not the TERM_vdg module.

To run the Procedure file *TERM_Patch*:

Enter: **MODPATCH TERM_Patch**

If you prefer to use DEBUG, type in the following commands:

```
Enter: DEBUG
Enter: l term
Enter: . .+2C
Enter: =50
Enter: . .+4
Enter: =02
Enter: . .+2
Enter: =02
Enter: =02
Enter: q
```

OK! So now we have a modified TERM in memory and we want to make it a permanent part of our system disk. Last time we used COBBLER, but this time we're going to use OS9GEN. This will be more complicated than our previous method, so let's do it by the numbers.

1. Enter: **SAVE /d0/term term**
   This will copy the term module from memory to disk.

2. Enter: **MAKDIR tempboot**
   Create a directory to store our new OS9Boot modules.

3. Enter: **CHD /d0/tempboot**

Before we proceed to the next step we need to have in our CMDS directory a utility that will separate the modules that are merged together in the OS9Boot file. There are several such utilities available:

MODUSTER is available from D.P. Johnson, 7655 S.W. Cedarcrest St., Portland, OR. 97223 (503) 244-8152

SEPARATE is a Public Domain file available on Compuserve, Delphi and other CoCo/OS9 BBS's

4. Enter: **MODBUSTER /d0/os9boot or SEPARATE /d0/os9boot**

5. Enter: **DEL term**

6. Enter: **COPY /d0/term term**
   Copy the term module you saved from memory into the TEMPBOOT directory to replace the term module that you deleted in step 5.

7. Enter: **MERGE ioman rbf cc3disk cc3go d0 d1 scf clock t2 sio printer p cc3io etc. >newboot**

   MERGE all of the files in you TEMPBOOT directory to newboot.

8. Place a freshly formatted disk in drive 1

9. Enter: **OS9GEN /d1 </d0/tempboot/newboot**

10. Enter: **CTRL-ESC**
    Hold the CTRL key down while pressing the ESCape key. This will send an EOF (End of File) marker to close the file.

11. Enter: **DIR /d1**
    Just checking to see if a new OS9Boot file was really created on /D1.

12. Enter: **CHD /d0**

13. Enter: **DSAVE /d0 /d1 ! shell**
    This will copy all of your directories and files over from /d0 to /d1.

That's it!!!!!!

Review: We used modpatch or debug to modify the TERM module then we saved the modified module to disk. Then we split the the OS9Boot file, storing the individual modules to a blank directory named TEMPBOOT. We deleted the original TERM module and replaced it with our modified TERM module. Finally we MERGED all of the modules into a file called newboot then we OS9GENed a new OS9Boot file to a freshly formatted disk. To finish up, we used the DSAVE utility to copy directories and files from the original system disk to our new disk.

There must be an easier way! As involved as the OS9GEN process is, you will find it a necessary process that you will do more often then you would suspect.

There IS an easier way. Purchase EZGEN for $19.95 from Burke & Burke, P.O. Box 733, Maple Valley, WA. 98038 (206) 237-2409. With EZGEN you simply delete the old module and insert the new.

Chapter 4

Advanced OS-9 Functions

Finding the hidden features of OS-9

Using advanced features - EDIT

OS-9 is supplied with a line editor that has some very nice features. Line editors tend to be more difficult to use then screen editors, and because of that are usually left with limited features. This is the case with the MS-DOS editor, EDLIN. The OS-9 editor however, is completely usable as the only editor on a system..

The OS-9 Editor has some incredibly powerful features not even found on many word processors.

- **Compact Size (about 5K with at least 2K buffer)**
- **Multiple read and write files open at one time (up to 50!)**
- **All OS-9 commands available inside the editor**
- **Adjustable workspace size (up to 60K)**
- **Repeatable command sequences**
- **Edit macros**
- **Multiple text buffers**
- **Powerful commands -**
  - Move edit position
  - Change edit buffers
  - Search
  - Search and replace
  - Delete line(s)
  - Add to the end of line(s)
  - Get and put lines between buffers
  - Insert copies of a string into a single line
  - Kill characters starting at the cursor
  - List the text to edit
  - Change the workspace size
  - Set tab width
  - Truncates line(s)
  - Set verify mode on/off
  - Shell to OS-9 for any command
  - Display the amount of memory used/available
  - Edit macros

Using EDIT

The most common method of using EDIT is to modify an existing file or create a new one. When EDIT is started it looks for the specified filename. If the file isn't found, EDIT will create it. As an example, edit the STARTUP file to set the /T2 port baud rate.

Type **EDIT STARTUP** from the rooot directory of the boot drive. Then type **L*** to list the current contents from the edit position. Press **<Enter>** to move the edit cursor to where you want to insert the XMODE command (text will be inserted in the line previous to the last displayed line).

Text can be entered by preceding the first character with a space. If the space is not the first character, EDIT will assume it's a command. Type in the following two lines:

```
E:<Space> XMODE /T2 baud-6 <Enter>  (this will set the baud rate to 9600 buad).
E:Q      (this will save the file and exit).
```

This is a very simple editing session but very typical. It is also common to delete lines, change the edit cursor position and make text changes (with the change command). Most of those can be seen easily in the Homework Session below and won't be covered here. There are also much more complex sample session in the OS-9 manual if you want to find the real power in EDIT.

One comment about the change function is that any character can be used as delimiters. Normally to replace the string "Scott" with "Rodger" in a line you would use the following command in EDIT:

```
E: C/Scott/Rodger/
```

Here the slashes are being used as delimiters. If you wanted to change a path name which may contain slashes, this obviously would not work. To replace "**/D0**" with "**/DD**" you would have to use a different delimiter, such as :

```
E: C!/D0!DD!
```

Here the exclamation mark functions exactly the same. Keep in mind any character can be used for this purpose.

Common Pitfalls

There are to common problems when using EDIT. The firs is when trying to start it and an error 218 (File already exists) is immediately returned. This is caused because EDIT creates a file called SCRATCH to maintain all the edits. If the system is rebooted during an edit session, it will not be able to delete this scratch file. During a normal exit it will remove it. If it exists when you start EDIT, this message will appear. To correct the problem, just delete the SCRATCH file (unless you need it's contents, then just rename it).

The other common problem is loading a file and getting just part of it. It is possible to edit a C source file called HELLO.C that was perhaps 10K long by issuing the command:

```
EDIT HELLO.C
```

Because no buffer size was explicitly listed, it will default to a minimum (Probably about 4K). This means that the end of the file will not load. To get the whole file to load in, add about 6K to the file size and append it to the command. The correct command for this file would read:

```
EDIT HELLO.C #16K
```

Homework Session

In chapter 3 we used the BUILD utility to write patch files for MODPATCH. In fact we have been using the BUILD utility a great deal in all 3 previous lessons. But we have been overlooking Microware's Macro Line Editor. You'll find this very powerful application in your CMDS directory listed as **EDIT**.

All of the things we have previously done with BUILD you could have done exactly the same way with EDIT. But EDIT is much more powerful and therefore more complicated to use IF you choose to take advantage of it's greater capabilities. The nice thing about EDIT is you can operate it at whatever level of sophistication you wish.

MACRO EDITOR

To get started load in the "TERM_Patch" file that you wrote as part of Lesson 3:

Enter: **EDIT TERM_Patch**

- OR -

Enter: **EDIT TERM_Patch NEW_TERM_Patch**

You couldn't do that with BUILD. BUILD will only create and save a file. It will not load a file, nor will it permit you to edit your entries, and it certainly won't let you save your modified file to a new file-name. In other words, eventually you will have to use EDIT.

NOTE: If you did not save the TERM_Patch file from the previous lesson, then the command above will still work. EDIT will simply create the file instead of loading a previously created file.

Remember that BUILD prompted each line with a question mark "**?**". EDIT however, prompts each line with the letter "**E:**" followed by a colon. Run through the following exercises to get acquainted with some of the EDIT commands:

E:**<ENTER>**

EDIT will display: c 2c 28 50 (2nd line of TERM_Patch)

E:**<ENTER>**

EDIT will display "c 30 01 02", the 3rd line of TERM_Patch

E:**L<ENTER>**

EDIT will display "c 30 01 02", the 3rd line of TERM_Patch

Each time you press <ENTER> alone, you advance the Editor one line in your file and the new line is displayed. As a contrast, "**L**" (short for List) will display the current line without advancing the Editor.

E:- (Press <ENTER> after the command)

EDIT will display "c 2c 28 50" the 2nd line of TERM_Patch

E:-

EDIT will display "l term", the 1st line of TERM_Patch

Obviously, the "-" minus sign backs-up the editor one line. Hmmmm, what does the "+" plus sign do?

E:+

EDIT will display "c 2c 28 50" the 2nd line of TERM_Patch

Pressing the ENTER key by itself advances the Editor one line and displays the new line, while pressing the "+" sign will do exactly the same thing. How about......

```
E:+5
```

EDIT will display "c 35 03 02", the 6th line of TERM_Patch

If "+5" advances the editor 5 lines then "**-5**" should have the opposite effect.

```
E:-5
```

EDIT will display "l term", the first line of TERM_Patch

The "+" and "-" signs are logical command functions. The plus sign to go forward and the minus sign to go backwards. In fact, all of the EDITor commands are logical single key entires with only one modifier, that being the wild card symbol "*" (asterisk). The wild card = "ALL"

```
E:L*
```

EDIT will display the entire listing of TERM_Patch

```
E:+*
```

EDIT will advance to the last line, one line beyond the last line of text.

```
E:-*
```

EDIT will display "l term", the first line of TERM_Patch

There are three other basic commands that we need to get acquainted with: The first one is the "**D**"elete command; the second is the "**S**"earch command, and the third is the "**C**"hange command.

```
E:S;c 34 03 02;
```

EDIT will display "c 34 03 02", the fifth line of TERM_Patch

Notice the ";" semi-colon used as deliminators (boundary marks). You can use any symbol you want as long as both deliminators are the same symbol.

```
E:S*;02;
```

EDIT will display line 3, line 4, line 5 and line 6

```
E:-*            (Back-up to the beginning of the file)
E:+4
E:C/02/00/      *NOTE: "/" used for deliminators
```

EDIT will display "c 34 03 00", the 5th line modified

```
E:C,00,02,      *NOTE: "," used for deliminators
```

EDIT will display "c 34 03 02", the 5th line restored

```
E:-*
E:C*.03 02.03 00.
```

EDIT will display lines 5 and 6, both lines modified

```
E:-*
E:C*"03 00"03 02"   *NOTE: quotation mark deliminators
```

EDIT will restore "all" lines meeting the first argument and Change them to the second argument, restoring the original values.

The Delete command is almost too obvious to practice, but...

```
E:-*          * Return to top of listing
E:L*          * EDIT displays entire listing
E:S/v/        * EDIT displays the last line "v"
E:D           * EDIT displays text it deleted
E:-*          * Return to top of listing
E:L*          * EDIT displays entire listing less the "v"
E:+*          * Advance to end of file
E:  v         * NOTE the space between the prompt and "v"
```

The last line is of GREAT IMPORTANCE! The space immediately after the "E:" is reserved for EDIT Commands. If you are simply entering text, you must enter a space between the prompt and the text.

> **TAKE A BREAK**: A listing of all of the EDIT Commands is provided below. Cut it out or photocopy them and post them near your computer. Better yet, play around with EDIT until you have memorized all of the commands!

MACRO EDITOR COMMANDS

+	= Move to next line
+*n*	= Move *n* number of lines ahead
+*	= Move to end of listing
-	= Move back one line
-*n*	= Move back *n* number of lines
-*	= Move back to beginning of file
>*n*	= Moves the editor to the right *n* characters
<*n*	= Moves the editor to the left *n* characters
S *string*	= Search for string
S*n string*	= Search for *n* number of string occurences
S*	= Search for all number of string occurences
D	= Delete line
D*n*	= Delete *n* number of lines
D*	= Delete all lines
C *string1*	= Changes string 1 with string 2 *string2*
C*n string1*	= Changes *n* number of occurences of *string1 string2* with *string2*
Q	= Quit
X*n*	= Displays the *n* lines that preceed the current line
[CTRL][7]	= Moves the editor to the first character
/	= Moves the editor past the end of the file
I*n string*	= Inserts *n* copies of *string* immediately before current position of editor
K*n*	= Kills *n* number of characters starting at current position of the editor
.SHELL	= Permits use of any OS-9 shell command

NOTE: Check out Chapter 7 in the OS-9 Commands section of your OS9 Level Two Manual for more EDIT Commands.

We still have our TERM_Patch loaded into our EDITor and we need to get rid of it in order to proceed with our next example.

```
E:Q
```

The OS9> prompt should return to the screen indicating that we are no longer in the Macro Editor. Now we are going to use the Editor to write a short MACRO.

macro: a short computer instruction that represents a sequence of operations----also called ***macroinstructions***
----Websters 1974 Collegiate Dictionary----

```
OS9>BUILD Macro
? c*/03 02/03 01 *Background and Border Colors/
? q
? <ENTER>

OS9>EDIT TERM_Patch <Macro
```

(Bunch of activity going on here)

```
OS9>LIST TERM_Patch
```

TAH DAH !!

WOW!

The power of a Macro Line Editor

Chapter 5

Public Domain Utilities

More than Microware or Radio Shack ever expected

The OS-9 package is supplied with a very wide variety of commands. However, there are certina limitations when using these commands. They could be improved upon. Fortunately, because of the modular nature of OS-9 it is a very simple matter to add or replace commands with more powerful ones.

A 5 -1/4 inch floppy disk is supplied with this tutorial that conatains some "must have" commands and utilities that are described below. This disk is an OS-9 data disk, meaning that it does not contain and OS9Bootfile and therefore cannot boot up OS-9. You will need to boot you computer with your system disk and then place the data disk into your disk drive. Enter **CHD /d0** in order to view a directory of the available files.

Disk Contents:

```
Shell21.ar
Datamod.ar
EasyEdit.ar

CMDS
        tree
        ed
        dircopy
        ar
        shell
        ls

DOCS
        EdDocs.doc
```

TREE allows the entire disk to be viewed as one large tree with each level indented. This can be very helpful when trying to locate a single file. The directory listing above is a "tree" listing.

ED is a full screen editor, similar to EDIT, but much **much** easier to use.

Dircopy allows entire directories or directory trees to be copied much more simply than by using DSAVE.

AR is a file archival utility. It allows multiple files to be merged and compressed into a single smaller file. This can be used for backups or to transmit files via a modem. Almost all files downloaded off Bulletin Boards use some type of archival program.

Shell+ is an enhanced shell with many new features and is easier to use. This can actually be used as a replacement for the standard OS-9 shell and is strongly recommended.

LS replaces the DIRectory command. This is actually a UNIX system directory utility. The major advantage is that is supports wildcards and has output that can be piped into other functions.

EasyEdit is a bootfile modification utility. It allows you to modify your device descriptor modules in the OS9Bootfile without having to use MODPATCH or DEBUG or COBBLER or OS9GEN.

Homework Session

During the past four lessons, we have learned the basics of OS-9, including the modification of boot modules and the writing of procedure files. By this time it may have become apparent that many of the command utilities are bare essentials, without options and lacking sophistication. Boy aren't we spoiled! But it's true. Most of the OS-9 commands are not much better than those provided in Extended Disk Basic, yet OS-9 is suppose to be so much more powerful?

We are going to examine some major public domain programs that you may choose to include in your CMDS directory to supplement the original utilities you already have. A single sided OS-9 data disk is included as part of the ***OS-9 Level Two Tutorial***. You will find that all of the files referred to in this section are on this disk. In some cases you may even choose to replace some of your original files with these more powerful utility files that serve the same functions, only better.

AR:

I did not plan to include the ***ar*** utility in this lesson but out of necessity, here it is. *AR* is a file compression utility, or better known as an "archiving" program. It can combine several files into one large file and then compress the files by removing all unnecessary spaces and packing the data code. The whole point is to reduce the file length to save disk space or reduce transmission time when downloading files over telephone modems. It was necessary to archive all of the *shell+*, *datamodand* *EasyEdit* files in order to include everything on one single sided disk. Simply couldn't have done it without ***ar***.

The best thing about this part of the lesson is you will have to use *ar* whether you want to or not! The only way you can get at the 17 *shell+* files will be to de-**ar**chive the *Shell21.AR* file.

First we'll need to format a disk to accept all of the *shell+* files. *Shell+* along with it's companion program *datamod* are of such major importance that they deserve their very own disk anyway.

ENTER: **format /d1 r "SHELL.PLUS"**

ENTER: **copy /d0/shell21.ar /d1/shell21.ar**
(If you have only one drive, ENTER: **copy /d0/shell21.ar /d0/shell21.ar -s)**

ENTER: **copy /d0/datamod.ar /d1/datamod.ar**
(If you have only one drive, ENTER: **copy /d0/datamod.ar /d0/datamod.ar)**

(With a single drive system, make sure your system disk is back in drive 0, then ENTER: **load ar)**

ENTER: **ar**
This will display the help screen for *ar*. To extract files from an archived file, use the **-x** command.

ENTER: **chd /d1** **(cd /d1)** = *shell+* equivalent

(With a single drive system, remove the system disk and place your Shell+ disk into drive 0)

ENTER: **ar-x shell21**

Stand back and watch the magic....

ENTER: **ar-x datamod**

More magic....

ar is utility that is a requirement for every OS-9 user, simply because without it you won't have access to files that have been archived. But then again, if you don't have any archived files, then you'll never use it.....Don't hold your breath, someday someone is going to give you some real neat "stuff" that you just gotta have. "SUPERFILE.AR" That "AR" extension tells you that the file is archived with the *ar* utility.

You should also consider that floppy disk, although not very expensive, can store more data when files are archived. *ar* is capable of compressing text files by more than 50%. And listing a text file from an ARchived file to your screen or to your printer is just as fast as a normal listing. With *AR* you Enter: **ar -p shell21 shellscript** or **ar -p shell21 shellplus.doc** >/p.

SHELL+:

Now we can get to work on your SHELL, the core of the OS-9 operating system. Sometimes we tend to compare OS-9 with MS-DOS and question why OS-9 doesn't do some of the "neat" things that MS-DOS does. Well, SHELL+ does! In MS-DOS you have the ability to change the screen prompt, so does Shell+. In MS-DOS you have the ability to use wild cards in your command lines, ie., "DELETE *.*", so does Shell+. In MS-DOS you have the ability to set multiple paths or multiple directories as default, so does Shell+, PLUS much, much more.

My greatest frustration with OS-9 is forgetting which directory I'm using, and as a result I get lost and keep getting error 216 (bad path name). With Shell+ I have the prompt set to always display the current time and the current working directory. No more #216 errors for me!

In order to replace the standard SHELL with SHELL+ we will have to *ident* our shell file in the CMDS directory to see if or what files have been merged with SHELL. We will then have to compare the new shell with the old to see what the size difference is in order that we can determine what files we can merge with the new SHELL+ and what files may have to be left out due to size restrictions. Keep in mind that we do not want to exceed 8K since that is the size of one memory block. No matter how small a file is, OS-9 Level Two will always designate 8K of memory for every file loaded into memory individually. If we can merge several files with SHELL+ and keep the total merged file length under 8K, OS-9 will be tricked into loading all of those files into one 8K memory block.

ENTER: **ident /d0/cmds/shell (ident /d0/cmds/shell >/p)**

Make a list of all of the merged files and their lengths.

ENTER: **del /d0/cmds/shell**
Delete the original shell (including merged files)

ENTER: **copy /d1/cmds/shell2.1 /d0/cmds/shell**
(If you have a single drive system, replace your system disk with your shell+ disk that you de*ar*chived previously. Enter: **copy /d0/shell2.1 /d0/cmds/shell -s)**
This will copy over SHELL+ to your CMDS directory

ENTER: **copy /d1/cmds/ls /d0/cmds/ls**
(If you have a single drive system, replace your system disk with the Tutorial Disk. Enter: **copy /d0/cmds/ls /d0/cmds/ls -s)**
This will copy over LS directory utility to your CMDS directory

ENTER: **ls -e /d0/cmds (ls -e /d0/cmds >/p)**

Make a note of the size of the new *shell* file. (Note that LS prints size in decimal instead of hexidecimal......COOL!)

Now make a selection of the files that you want to include with *shell+* to make up the 8K merged SHELL file. Actually you will need to leave appproximately 130 bytes of free space so calculate 7,870 bytes as the maximum merged file size.

According to *LS* directory utility, the new *shell* file (*shell+*) is 6323 bytes leaving 1547 bytes of space to add some merged files. Personally I merged *shell+*, *setime*, *link*, *list*, *echo*, *iniz*, *unlink*, *del* and *procs* into SHELL (as shown below).

```
MERGE shell21 setime link echo list iniz unlink procs >new
ATTR new e
DEL shell
RENAME new shell
```

After you have merged all of your selected utilities into a new *SHELL* file in your CMDS directory, reboot your computer. The first thing you'll notice after the initial OS-9 Welcome screen clears is the "**Shell+ v2.1**" followed by the date. The second thing you should notice is the new OS9 prompt. Instead of the usual **OS9:** prompt, you should see the current time, followed by the current device "**[TERM]**", followed by your current/default path name/directory "**/D0**".

SHELL+ is capable of many things that are not possible under the original *shell*. It includes sufficient system commands and arguments to qualify as a programming language. Listed here are a few of the new commands available in *shell+*:

.PWD	Same as original PWD but now built into *shell*
.PXD	Same as original PXD but now built into *shell*
Path=	Sets more than one directory as default (Path=/d1/cmds /d2/cmds) will cause OS-9 to search the CMDS directories on /d0, /d1, /d2 instead of the standard default /d0/CMDS.
p=prompt	Sets customized OS9 prompts (p="Hello!") (p=") [@] $>") will result in: Date[term device]current path
var.1-10	Supports up to 10 variables in memory

pause	Waits for a key press or mouse click (pause 'Hit any key when ready')
if/then/else/endif	Logic arguments
goto	Logic arguments
onerr goto	Logic arguments
*** / ?**	Wildcard arguments
cd & cx	Same as CHD and CHX
prompt	Similar to echo but without linefeed

The *SHELL* documentation file explains how to use all of these neat new features. It's not necessary to read the documentation to use SHELL+ at present, but when you're ready to advance and take advantage of your new shell, the documentation is very complete with lot's of examples.

ED:

This next utility is actually a complete application soft-ware that is being given to you as a reward for putting up with all of those previous exercises using *BUILD* and *EDIT*.

Hopefully, after Chapter 4's Homework Session <u>you</u> <u>are</u> <u>aware</u> that *EDIT* is a very powerful "macro" editor. But, its still strictly a line editor and certainly does not resemble anything close to a word processor. Fortunately, there are several very good full screen text editors available for free in the public domain:

ED1.6 ED2.0 SLED WP UEMACS

And of course their are several very good commercial text editor/word processors available from $10 - $200:

SCRED	TSEDIT	SCREEN STAR	STYLOGRAPH
VED	DYNASTAR	WINDOW-WRITER	DESK-MATE

ED1.6 is included on the data disk along with documentation in the DOCS directory. This program is very easy to use, the documentation is almost superfluous, but read it anyway. We are going to use ED to help write some *help* files.

HELPMSG is a file located in your SYS directory. If you do not have a SYS directory on you boot disk or a *helpmsg* file in you SYS directory, find the copy of your original OS-9 boot that contains a copy of the original SYS directory.

ENTER: **ed /d0/sys/helpmsg**

Ed will initialize and load in *helpmsg*. Please note the format being used. the "@" placed before each file name, the "Syntax", "Usage" and "Opts" headings used to describe each file, etc.

Use the down and up arrow keys to scroll through the text until you find @**BACKUP** then insert the following:

ENTER: @**AR**
ENTER: **Syntax : Ar <-cmd>[<modifier>] [file ..]**

ENTER: **Usage** : **Archive file manager**
ENTER: **Cmds** : **t = show table of contents for archive**
u = update/add file(s) to the archive
p = print file(s) from the archive
x = extract file(s) from the archive
ENTER: **Mods** : **a = all versions (for extract)**
s = suppress file compression
z = read names for <cmd> from stdin

Use the down and up arrow keys to scroll through the text until you find **@DISPLAY** then insert the following:

ENTER: **@DIRCOPY**
ENTER: **Syntax** : **DirCopy <from path> <to path> [opts]**
ENTER: **Usage** : **Copy files from source directory to destination directory**
ENTER: **Opts** : **C = confirm copying of all files**
D = enable copying of sub-directories
I = interactive mode
R = auto overwrite of existing files
S = sorted directory
T = replace outdated to_path files
U = update to_path file's owner number and date

ENTER: **[CTRL-E]** (Hold the CTRL key down and press "E"

Initiates the FIND option on the EDitor

ENTER: **@ECHO**
(If you get an error message, press [ENTER] and try again)

ENTER: **@EASYEDIT**
ENTER: **Syntax** : **easyedit**
ENTER: **Usage** : **Modifies descriptors in the OS9Bootfile**

ENTER: **[CTRL-F]**
ENTER: **@DATE**

ENTER: **@DATAMOD**
ENTER: **Syntax** : **datamod {modname} <{pathin} >{pathout}**
ENTER: **Usage** : **Takes shellscript from stdin and converts it to a data module in stdout**
ENTER: **Syntax** : **datamod -<{pathin} >{pathout}**
ENTER: **Usage** : **Takes a data module from stdin and writes it to stdout as a shellscript**
ENTER: **Syntax** : **datamod -m {modname} >{pathout}**
ENTER: **Usage** : **Takes a data module in memory and writes it to stdout as a shellscript**

ENTER: **[CTRL-F]**
ENTER: **@MAKDIR**

ENTER: **@LS**

ENTER: **Syntax** : **ls [-opts] [<pathname><filename>]**
ENTER: **Usage** : **Unix type directory utility**
ENTER: **Opts** : **-? = show HELP**
-S = flag directory files with "*"
-D = display subdirectories
-E = display all information
-N = narrow format
-P = PIPE mode on output

ENTER: **[CTRL-F]**
ENTER: **@TUNEPORT**

ENTER: **@TREE**
ENTER: **Syntax** : **tree [-opts] {directory path}**
ENTER: **Opts** : **-f = don't report files**
-ln= only n levels of the tree (1..9)
-u = report space utilization

ENTER: **[ALT-F]**

This should display the FILE MENU. Use the arrow keys to highlight the **Save** option and press <ENTER>. After the file is saved to disk....

ENTER: **[ALT-Q]**

> At last we're done. We've used *ED* to modify/add new text to the *HELPMSG* file in our SYS directory. Any time you have questions about how to operate a certain file just enter **help**. OS-9 will prompt you for the name of the file that you seek help, and then display the appropriate information about the file you requested. (MS-DOS won't do that!)

DIRCOPY:

Before we go any further, let's copy the files over from the CMDS directory on the data disk to the CMDS directory on your boot disk. If you are running double sided disk drives you should have plenty of room. If you are operating single sided disk (630 to 720 sectors) you may not have enough room on your disk to store all of the files. The CMDS directory on the data disk contains 68,980 bytes or 231 sectors.

(NOTE: If you only have a single disk drive system, you will not be able use *DIRCOPY*.)

ENTER: **load /dl/cmds/dircopy**

ENTER: **dircopy /dl/cmds /d0/cmds**

If you do not have enough space on your boot disk for all of the files, you can select the files you want *dircopy* to copy by using the **interactive** mode command.

ENTER: **dircopy /dl/cmds /d0/cmds i**
Dircopy will display all of the files on "/d0/cmds" with a prompt: **A C P M R S X - (H-help) ?**

ENTER: **H** (to get a display of options)

ENTER: **A** (to arrange file for copy)

Dircopy will display another set of options: **C P X**

ENTER: **C**

Dircopy will prompt you with: **copy shell (C/P*/X) ?**

If you want to copy *shell* then ENTER: **C** else ENTER: **P**.

Dircopy will prompt you with: **copy dircopy (C/P*/X)**

If you want to copy *dircopy* then ENTER: **C** else **P**.

Continue making your selections until you reach the end of the directory.

Dircopy will then prompt you with: **C P X - (H-help)**

ENTER: **X**

Dircopy will then prompt: **A C P M R S X - (H-help)**

ENTER: **X**

Dircopy will copy the files you selected to /D0. If a file being copied already existed on your boot disk, *dircopy* will ask if you want to overwrite the file.

If you want to copy all of the files on the Tutorial Disk in drive 1, including the CMDS directory to another disk in drive 0:

ENTER: **dircopy /d1 /d0 d**

After going through our exercise above, it should be apparent that *dircopy* is similar, yet much more flexible then *dsave*. Since they both serve the same function, you will probably want to delete *dsave*. Having both programs on your disk is a luxury that only hard drive users can afford.

EASYEDIT:

This utility is an answer to an OS-9 users prayer. It will modify most of your device descriptors and then write over your old OS9Bootfile on your system disk with you new modifications. No MODPATCHES, no FORMATting, no COBBLERing or OS9GENing.

ENTER: **easyedit**

Easyedit greets you with a colorful title screen and a prompt requesting for the pathname of the OS9Boot file that you want to modify. If you want to modify the OS9Boot file on your system disk in drive 0, just press <**ENTER**>.

Next Questions:
"What type of monitor: ?" (RGB = **R** Composite = **C** Monochrome = **M**)

Next Question:
Default screen width (32/40/80)

EasyEdit then display all of the possible device descriptors available in your OS9Boot file that can be modified:

```
Select Descriptors:
D0
D1
DD
T1
T2
P
TERM
W
W1
W2
W3
W4
W5
W6
W7


<Q>uit        <ENTER> = select          arrows to move
```

After you select the descriptor module that you want to modify, EasyEdit will then prompt you for each possible parameter that can be modified in that specific descriptor module. The example below shows what to expect if you wanted to modify the disk drive #1 descriptor:

```
Editing descriptor: D1
Regarding the drive:
New Stepping rate (30) : ?
How many tracks (40):  ?
Number of sides  1/2*) :
Verify (write check) on  (Y) :

Data correct?
```

When you have finished, EasyEdit returns to the Descriptor listing. If you don't wish to modify any other descriptors, Enter <**Q**>. EasyEdit will then re-write the OS9Boot file on your system disk with your updated descriptors replacing the original files. How simple and efficient!

TREE:

Now that you have your public domain files copied over to your boot disk CMDS directory, lets use the *tree* utility just for a little diversion from all of this hard work:

ENTER: **tree /d0**

Now that is one strange directory display. It shows all of the directories and all of the files within each directory, similar to the concept of twigs on the branches of a tree. Get it!

ENTER: **tree**

Tree displays a help screen showing the command syntax and the options available in the command line:

```
Usage: tree [-flu] from_dir
       f  = don't report files
       ln = only n levels of the tree (1..9)
       u  = report space utilization
```

ENTER: **tree -f /d0**

Try some of the other options......

That was real quick and easy. Notice that when you entered *tree* on the command line by itself without any pathname or argument(s), the file produced a help message. Most of the files included on our Public Domain data disk have help options. Don't you wish all files included help screens? Actually most OS-9 utility files do have help options that appear when the file name is entered without pathnames or legal options. When you have some spare time you might try entering utility command names without options just to see how many of the files in your CMDS directory do have the help option.

THAT'S IT! Sorry for making this Lesson so short. But we did cover the usage and installation of ***shell***+, and the ***ls*** directory utility, ***DirCopy*** (*dsave*'s replacement), ***tree*** (also known as a hiearchical directory), *EasyEdit*, and finally the archive utility ***ar***.

Almost forgot ***Ed***, the full screen text editor. Does every-thing a word processor does except word wrap. Sure beats the heck out of *edit*.

Hardware Upgrades for the Color Computer-3

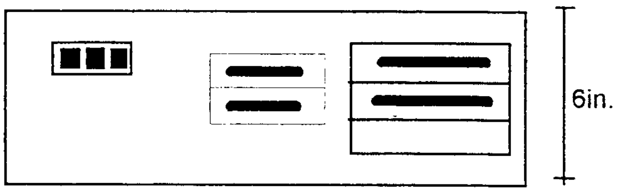
Table of Contents

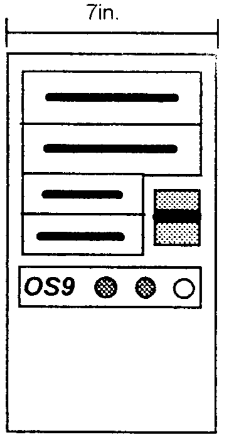
CoCo to PC-Case Installation

DISCLAIMER:
The installation of your color computer into a PC type case may require minor or major modification to your color computer hardware. It is understood that any modification voids all warranties and may result to inadvertent damage to components and/or circuit traces on the circuit boards. Every effort should be made to limit board modifications. All modifications to the circuit boards in this text are restricted to only those necessary to permit installation into appropriate PC cases.

TOOLS:

1. Electric Drill (Drilling mounting holes in PC Case
2. Hack Saw (For cutting older style Multipak Circuit Board)
3. Soldering Iron (15-25 watt) and fine electronic solder
4. Screwdrivers (small-medium straight edge and phillips)
5. Wire cutter (stripper)
6. Needle nose pliers or surgical forceps
7. File to reduce thickness of rom-pak card edge
8. Vice to press on solderless connectors





NOTE: ***Minimum height/width required is 6 inches to accommodate the width of the Color Computer and multipak circuit boards.***

PARTS:

1. Mini or Midi Tower or standard horizontal AT IBM clone PC Case
2. Multipak Interface (Preferably the newer/smaller model) OR the Disto 3-in-1 or 4-in-1 board OR one of the more recent multipak replacements now on the market.
3. Color Computer-3.

NOTE: ***It is assumed this will be for OS-9 Operation. Rom-Pak's will not be easily accessible in this installation. Also, the slot select switch will be permanently fixed at slot 4.*** 4.

4. "S-Cable" made up of 3 to 6 inches of 40 conductor ribbon cable and a male and female "*KEL-AM*" connectors. (Special connectors are available from Terry Laraway. Standard female card edge connectors will not work!.)
5. Disk Controller and associated Disk Drives

*6. Hard Drive Interface and Controller and associated Hard Drive IF you plan to use a Hard drive....this can be installed at a later date.

*7. Hi-Res Joystick Interface (If desired) ALSO Double Pole, Double Throw mini 12vdc relay. (Radio Shack Cat.# 275-249)

*8. Printer Interface (Serial to Parallel) if you want a parallel printer port output from the case, otherwise you will have a serial printer port.

*9. "Y-extension power cable". (This may be needed if you have more than 3 drives.)
10. 6 - 4/40 x 2inch bolts and nuts for mounting circuit boards
11. 4ft. of 4 conductor (22 gauge) wire (Radio Shack Cat# 278-858)
12. 4 pin amphenol jack and plug (Radio Shack Cat# 274-224 & 274-234)

I/O Port Options:

13. D-sub type chassis mounted jacks (may be supplied with case) 2 - 9 pin male solder type (Radio Shack Cat.# 276-1537) and 2 - 25 pin male solder type (RadioShack Cat# 276-1547)

NOTE: ***You may wish to use a 5-pin chassis mounted DIN connector instead of one of the 25-p Dsubconnector for the printer port and a 6-pin DIN connector instead of one of the 9-pin D-sub connector for the joystick port. See step 15 and 16 in the PROCEDURE section for more information***.

14. D-sub type solderless female connectors: 2 - 25 pin (Radio Shack Cat# 276-1565) and 2 - 9 pin (not available from Radio Shack)
15. 2 - RCA type 1/4 inch panel mount jacks (Radio Shack Cat# 274-346)
*16. 1 - 36 position Male Printer (Centronics) solderless connector (Radio Shack Cat# 276-1533) [Note: See #8 above]
17. 4 - 6 feet of 34 conductor ribbon cable (Keyboard extension Kit)
18. 4 - 6 feet of 25 conductor ribbon cable (Misc. I/O extension)
19. 1 - 2 feet of 10 conductor ribbon cable (Joystick I/O extension)
*20. 4 - 6 feet of 9 conductor ribbon cable (RGB Monitor extension)
*21. 1 - 10-pin solderless header for the RGB terminal on the computer circuit board.
*22. 1 - 10-pin male chassi mount header connector for RGB monitor.

Keyboard Extension Kit:

23. 1 - discarded rom-pack circuit board **OR** 1 - 1 inch length of 17 conductor single sided (34 conductor double sided) circuit board. (Conductors width and spacing must match the keyboard jack on the CoCo-3 motherboard.)
24. 1 - 34 pin card edge connectors (Radio Shack Cat.# 276-1533).

* = Optional parts. See procedures below.

PROCEDURE:

1. Determine desired orientation of computer and multipak circuit boards (See pictorial diagrams)
2. Remove the Voltage regulator IC and the associated heat sink from the multipak circuit board and the Color Computer-3.
3. It may be necessary to cut off the end of the reset switch on the Color Computer-3 that extends beyond the edge of the circuit board, also the power switch on both the Color Computer-3 and the Multipak **IF** the switches extend beyond the dimensions of the PC Case. The power switches no longer function anyway, and the reset lines will be wired to the reset switch on the front panel of the PC Case.
4. **IF USING THE OLDER LARGER MULTIPAK:** Remove the slot select switch and resistors R-1 & R-2. Cut off 3/4 inch from the bottom side, 2-1/4 inch on the right side (power supply components) See diagram on last page.

DANGER: Excessive flexing of the circuit board and vibration may cause damage to solder joints, circuit traces and integrated circuit components.

5. **IF USING THE OLDER LARGER MULTIPAK:** Solder a jumper to the appropriate slot select trace lines to achieve the same logic function accomplished by the slot select switch when positioned for slot-4

6. Use circuit board templates to determine the location of mounting holes to be drilled through the chassis of the PC case. Drill mounting holes with appropriate size drill bit to match the size of your mounting bolts.

8. Assemble an "S-Cable" using the *KEL-AM* connectors and approximately 2 to 4 inches of 40 conductor cable.

9. Mount Disk Drives (and Hard Drive) with hardware screws from their original mountings.

10. **<u>CONNECTING PC POWER SUPPLY</u>** ***(See Pages 5 & 6):*** Solder approximately 1 foot of hook up wire to the +12 vdc, -12 vdc, +5 vdc and Ground traces on the multipak. and the Color Computer circuit board.

12. Solder a short length of wire from the +12 vdc connection on the Color Computer to the +8 vdc regulator on the Color Computer circuit board.

13. Determine which of the 6-pin connectors from the PC Power Supply contains the following potentials: +12, -12, +5 and Ground. Cut off the original connector and replace with either the male or female 4-pin amphenol connector.

14. Solder the wires in step 10 above to the appropriate connections on the remaining 4-pin amphenol connector.

15. **IF YOU ARE NOT GOING TO MOUNT YOUR PRINTER INTERFACE IN THE COMPUTER:** Solder approximately 1 foot of wire to each of the corresponding trace lines on the bottom of the computer circuit board directly below the printr I/O jack. Solder the other end of each wire to one of the 25-pin D-sub chassis mounted jacks. Press fit (with a vice) 4 to 6 feet of 25 conductor ribbon cable to the 25-pin D-sub solderless connector. Solder the 5 corresponding conductors to your serial/parallel interface. (You may choose to use a 5-pin chassis mounted DIN connector instead of the 25-pin D-sub connector.)

    IF YOU ARE GOING TO MOUNT YOUR PRINTER INTERFACE IN THE COMPUTER: Solder approximately 1 foot of wire to each of the corresponding trace lines on the bottom of the computer circuit board directly below the printer I/O jack. Solder the other end of each wire to the correct input line on the serial/parallel interface. Solder the parallel output lines to one of the 25-pin D-sub chassis mounted jacks observing the correct pin numbers to correspond to the centronics printer connector. Press fit (with a vice) the 36 pin centronics connector to 4 - 6 feet of 36 conductor ribbon cable. At the other end, trim conductors 26-36 and press on (with a vice) the solderless D-sub 25-pin connector. Mount the interface in an appropriate location. If you want to have access to the baud switch, you might consider mounting the interface to the chassis with a switch extension shaft extending through the front of the case.

16. **IF YOU ARE GOING TO USE THE HI-RES JOYSTICK INTERFACE:** Solder approximately 1 foot of wire to each of the corresponding trace lines on the bottom of the computer circuit board directly below the cassette port and one of the joystick ports. Solder the other end to the appropriate connections on the Hi-Res Joystick Interface circuit board. (See special instruction for connecting a relay via the "TUBO" switch in order to access "HIGH" and "LOW" resolution from the front panel. Solder approximately 2 feet of wire to the output a 6-pin DIN jack plugged into the interface circuit board. Solder the other end of each wire to one of the 9-pin D-sub chassis mounted jacks. (You may choose to use a 6-pin DIN chassis mounted jack instead of the D-sub jack.)

    IF YOU ARE NOT GOING TO USE THE JOYSTICK INTERFACE: Solder approximately 1 - 2 feet of wire to each of the corresponding trace lines on the bottom of the computer circuit board directly below the joystick ports. Solder the other end of each wire to one of the 9-pin D-sub chassis mounted jacks. (You may choose to use a 6-pin DIN chassis mounted jack instead of the D-sub jack.)

17. **COMPOSITE MONITOR:** Solder approximately 1 foot of shielded.wire to the appropriate circuit trace lines on the bottom of the computer circuit board directly below the composite video I/O jack. Solder the other end of each wire to one of the 1/4 inch panel mounted RCA type jacks. Be sure the grounded conductor is connected to the grounded terminal on the jack.
18. **AUDIO I/O:** Solder approximately 1 foot of shielded wire to the appropriate circuit trace lines on the bottom of the computer circuit board directly below the audio I/O jack. Solder the other end of each wire to one of the 1/4 inch panel mounted RCA type jacks. Be sure the grounded conductor is connected to the grounded terminal on the jack.
19. **RGB MONITOR:** Press on the 10-pin solderless header connector to approximately 1 foot of 10 conductor ribbon cable. Press on (with a vice) a 10 pin chassi mount male solderless header to the other end of the cable and mount in one of the 9-pin D-sub chassis mounted jack holes on the rear of the case.
20. **IF YOU ARE USING A MAGNAVOX MONITOR:** Press on (with a vice) a solderless female 10 pin header to 4 feet of 10 conductor ribbon cable. At the other end solder a 6-pin male DIN connector. Check continuity to make sure input and output lines match.

    IF YOU ARE USING A TANDY CM-8 MONITOR: Press on (with a vice) a solderless female 10 pin header to 4 feet of 10 conductor ribbon cable. At the other end, press on (with a vice) a solderless 10 pin male header connector. Check continuity to make sure input and output lines match.
21. Connect the multipak and the color computer circuit boards together via the short "S-Cable" and mount the two circuit boards into the PC chassis. Use the shielded backings on the circuit boards as templates to provide accurate locations for drill holes in the PC Chassi. Re-attach the shielded backings before mounting the circuit boards with the 2 inch 4/40 bolts. Use extra nuts to provide desired mounting height.
22. Do a continuity test between the different voltage trace lines and the amphenol connectors....Then turn power on and check for proper voltages on both circuit boards. If no voltage is present then there is a short and the PC power supply will automatically shut off.
23. Plug in your Disk controller and connect to the disk drive(s). Connect disk drive power connectors to disk drive(s). Connect monitor to appropriate jack(s).
24. **KEYBOARD EXTENSION KIT** ***(See Page 7)*****:** Cut 1 inch off the end of a discarded rom-pak (preferably Spectaculator). Cut the card edge so that only 16 trace lines will fit into the keyboard jack on the computer circuit board. Then file, sand or grind down one side only until the card edge is the correct thickness to insert firmly into the keyboard jack without forcing or damaging the jack.
25. Press on (with a vice) a 34 pin card edge connector (female) on one end of a 4 to 6 ft. length of 34 conductor ribbon cable. Identify one side of the connector as the "Top Side" and at the other end cut the insulation off the 16 conductors that match the "Top Side" pins connectors.
26. Solder the 16 wires to the circuit board matching the keyboard conductors with the circuit board conductors.
27. Glue, or otherwise attach a thin piece of cardboard to the back side of the mylar keyboard cable and plug the keyboard into the other 34 pin card edge onnector.
28. Do a continuity test so that each trace line on the keyboard mylar cable corresponds to the proper pin on the keyboard jack on the computer.

SMOKING TEST:

After all voltage checks and continuity test have been made (with proper results), plug in the keyboard and monitor, then apply power. **Watch for smoke!**. If power levels remain constant at the right levels and you get the Color Computer "Disk Extended Basic" screen.......you're lucky! And you have succeeded with your installation! **CONGRATULATIONS! ! ! ! !**

PC Power Supply Connections

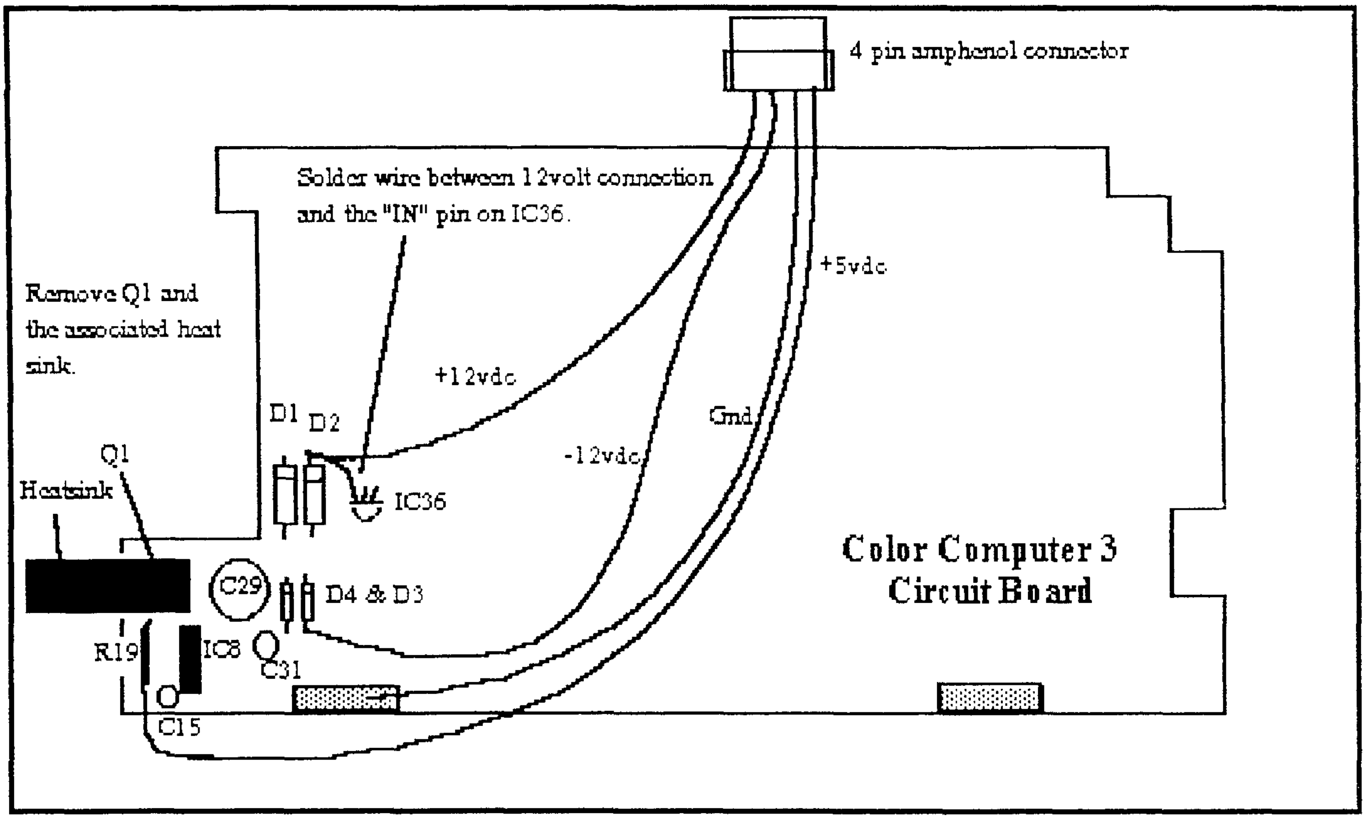

4 pin amphenol connector
Solder wire between 12volt connection and the "IN" pin on IC36.
+5vdc
Remove Q1 and the associated heat sink.
+12vdc
D1
D2
Gnd
-12vdc
Q1
Heatsink
IC36
Color Computer 3 Circuit Board
C29
D4 & D3
R19
IC8
C31
C15

Fig. 1

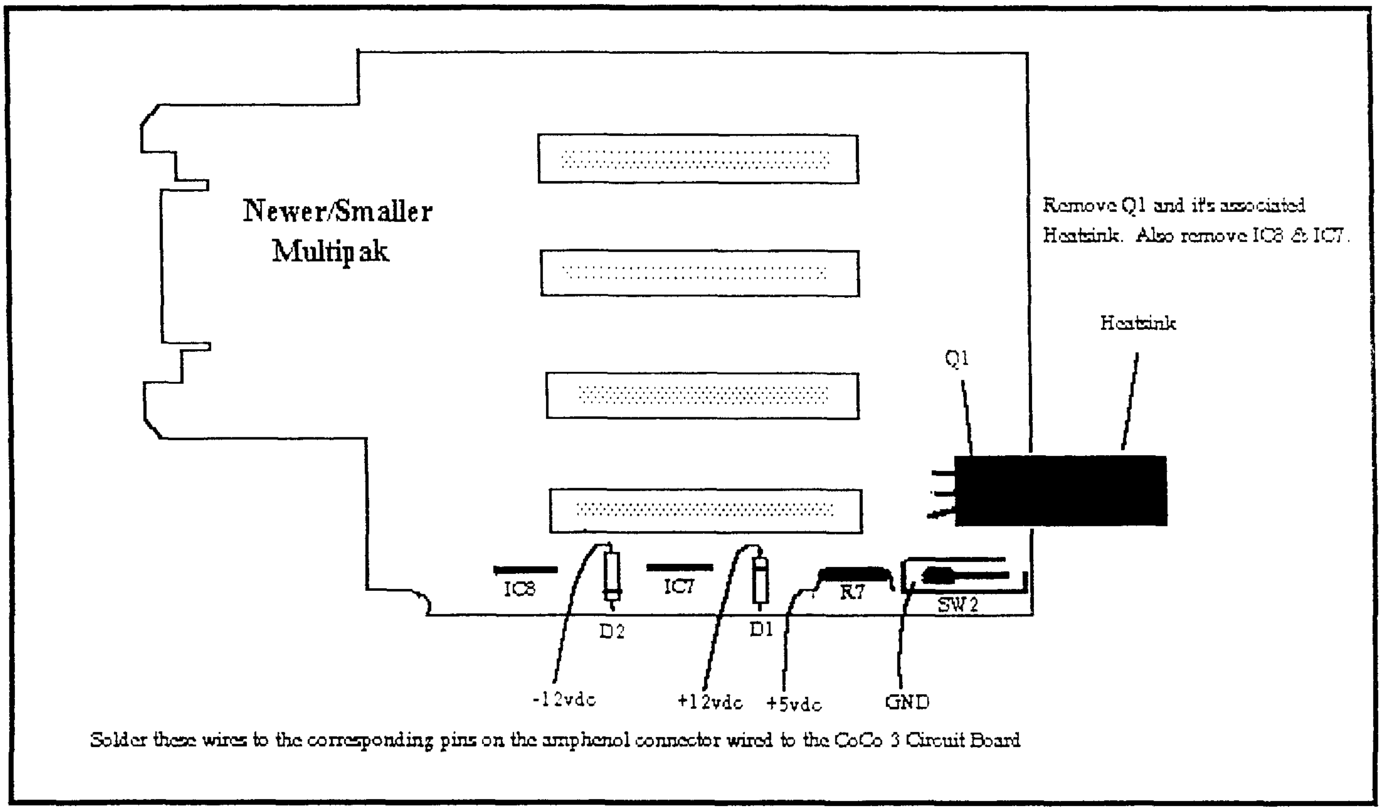
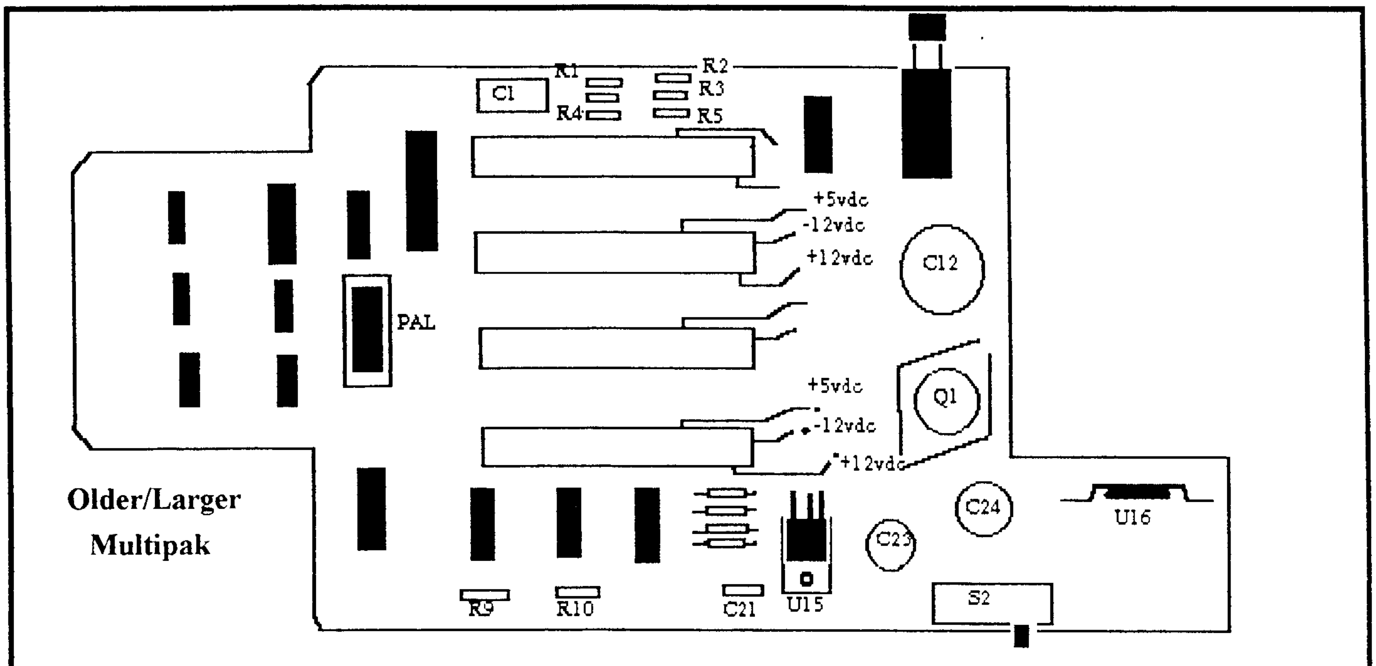

Newer/Smaller Multipak
Remove Q1 and it's associated Heatsink. Also remove IC8 & IC7.
Heatsink
Q1
IC8
IC7
R7
SW2
D2
D1
-12vdc
+12vdc
+5vdc
GND
Solder these wires to the corresponding pins on the amphenol connector wired to the CoCo 3 Circuit Board

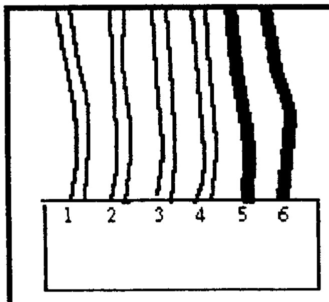
Fig. 2



Each ROM Pak jack has easily accessible traces for each voltage. Solder your wires to any of these traces and connect to the corresponding pins on the same amphenol connector that you wired to the CoCo-3 Circuit Board.

Fig. 3

Regardless of which Multipak you use, connect the wires to the same amphenol connector that you attach to the CoCo-3 Circuit Board (See Figure 1). Check your work using an ohm meter/continuity checker to be sure that the corresponding voltage lines on both the Multipak and the CoCo-3 are connected to the same pins on the amphenol connector.



PC Power Supply Color Code

Pin 1 = Voltage Good (System Test)
Pin 2 = +5 vdc (Red)
Pin 3 = +12 vdc (Yellow)
Pin 4 = -12 vdc (Blue)
Pin 5 = GND (Black)
Pin 6 = GND (Black)

Fig. 4

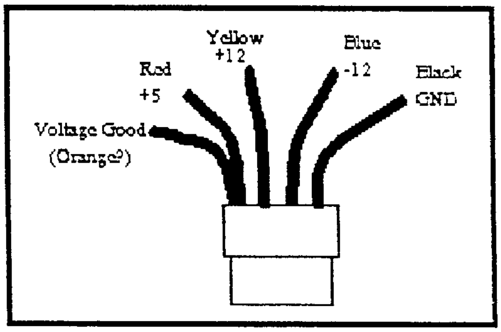
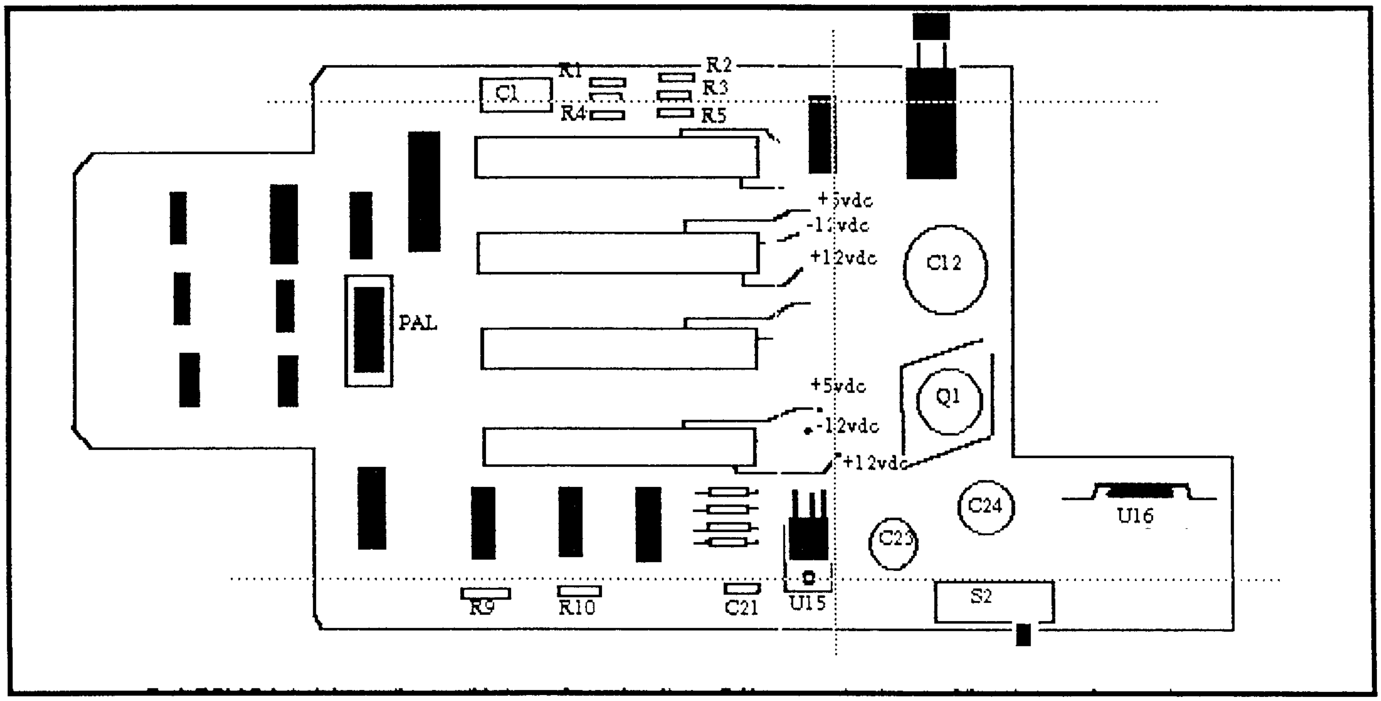


Fig. 5

Your PC Power Supply will have two 6-pin connectors designed to plug into a PC Circuit board. If your power supply is for an AT machine (80286, 80386, 80486) then you will see the above color coded wires on one of the two 6-pin connectors. *The Black ground wires will always be pins 5 & 6.* Cut this connector off and replace with a male 4-pin amphenol connector. Solder the "**Voltage Good**" wire to the Red wire (5volts) before soldering the Red wire to one of the amphenol pins. If your Power Supply is for an XT machine (8088, 8086) then simply ignore the "**Voltage Good**" wire. Solder the Black, Blue and Yellow wires to the remaining amphenol pins and insert the pins into the amphenol casing so that they will match the corresponding wires on the female amphenol connector attached to the CoCo/Multipak Circuit Boards.

'Hacking' the OLD Multipak

The problem with using the Older/Larger Multipak in the PC Case Installation is it's width is greater than that of the Color Computer-3. Since most PC Cases rarely are wider or higher than 6 inches, it makes it very difficult to fit the OLD Multipak anywhere in the case without using an excessive length of 'S Cable' (Y-Cable). Fortunately, a great deal of space is used on the Multipak for power supply support, and is not needed electrically or physically since we will be getting our power from the PC Power Supply (See Pages 6 & 7). The following diagrams will show you where you can cut away the excessive circuit board:



Notice the dotted "CUT" lines and how close they are to R9, R10, C21, R4 and R5. The Integrated Circuit located in the upper right corner next to the ON/OFF Switch is a SALT chip, similar to the one near the heat sink on the CoCo. Resistors R1 - R5 are simply part of a voltage divider network for the SALT chip. None of these components are necessary for our new installation. However, there are some logic lines running just below the + side of C1. These lines must remain intact, even if you have to solder some jumper wires to replace what you may have sawed off.

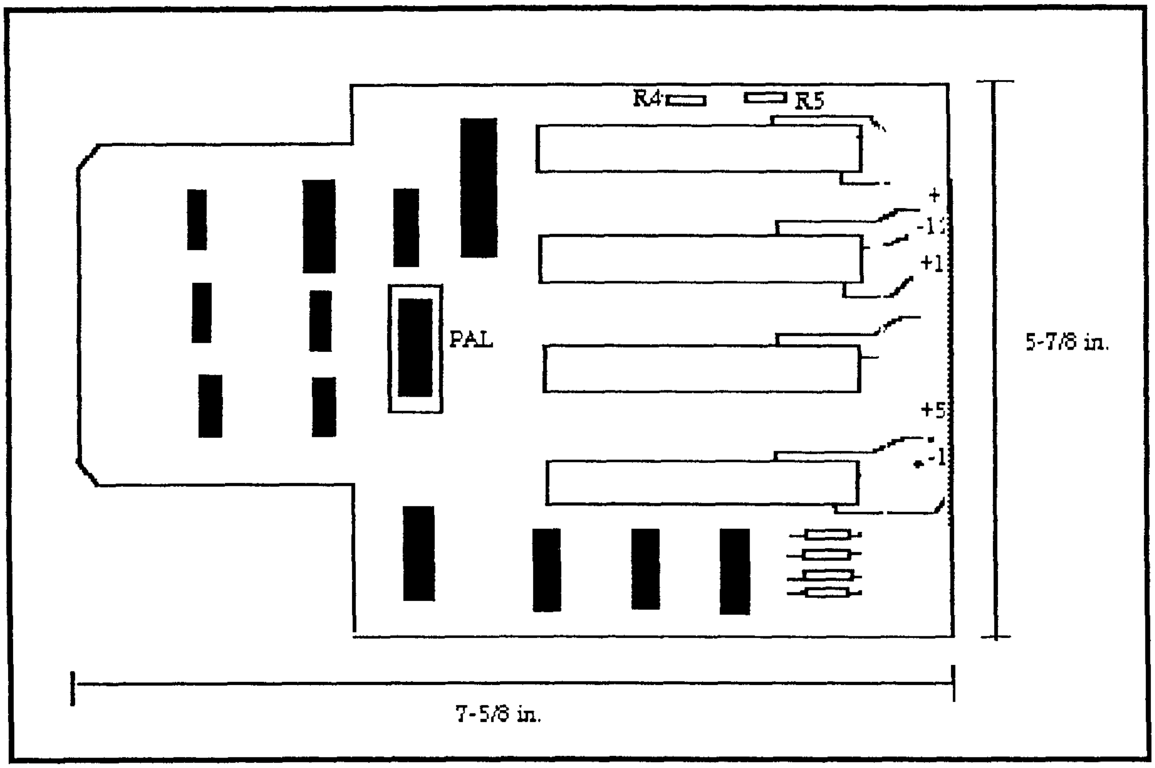
Remove the shielded backing from the circuit board as well as the SALT chip (U5) and the voltage regulator (U15) before carefully sawing the circuit board as marked above. After cutting the top portion of the circuit board, check both sides to see that the logic traces are still intact. If a trace was damaged, simply replace the trace with a length of small gauge wire wrap wire. Solder from the beginning of the trace to the end where the small solder pads are provided on the board. Don't try to scratch off the protective insulation and solder across the damaged area, you will only end up doing more damage.

Make the vertical cut along the ends of the 4 rom pak slots. Keep the cut just to the right of the voltage pads coming from the rom pak slots.

BEWARE: Before making the bottom horizontal cut, you need to be aware that the ROM SLOT SELECT SWITCH (S2) in position 4 (for disk controller access in slot 4) places a ground potential to both R9 and

R10. Since the resistors are in effect connected in parallel, the two 4700 ohm resistors result in a 2350 ohm resistor between ground and pin 16 of U12. After you make this final horizontal cut, you will have to remove the two resistors from the discarded cut off section and solder them electrically in parallel from any ground potential to pin 16 of U12. **ALSO** the SLOT SELECT SWITCH places a ground potential on pins 2, 4, 6 & 10 of U12. Use small gauge wire wrap wire to provide the ground connection to these pins. That's it!

NOW LOOK AT WHAT YOU'VE DONE!



Very Important: The bottom horizontal cut will also destroy two or more trace lines which you must replace with small guage (wire wrap type) insulated wire. The traces go from just below U11 to pins 8 on 2 or more of the I/O slots.

CoCo Keyboard Extension Kit

How many times have you wished that your computer desk could be neater instead of a bunch of cables, multipak, more cables, disk drives, more cables, etc. If you could just hide the CoCo out of site somewhere with only the keyboard, disk drives and monitor on the desk. It would be so neat.

The easies approach would be simply by a keyboard extension cable from CoCo Pro for $15 (1-800-937-7746, Catalog# AY-278), but I'm just too cheap to spend the money. Besides I like to "hack" around a little bit, and it couldn't be that hard to make some sort of keyboard extension, right?!

There are several approaches to making up a keyboard extension cable, here's mine.........

CONSTRUCTION:

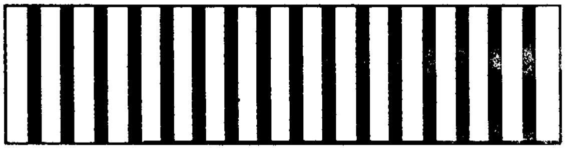
The idea is simple. Take a desired length of cable and devise a way to connect the keyboard at one end and attach the cable to the computer at the other end.

Parts:

1- 34 Conductor Card Edge Connectors (Radio Shack Cat.# 272-1533)
4 to 8 ft. of 34 Conductor Ribbon Cable
1 Discarded ROM Pak
11/16th x 1 inch piece of tag-board or thin card-board

1. Press on the 34 card edge connector (use a vice) to one end of the desired length of 34 conductor ribbon cable.

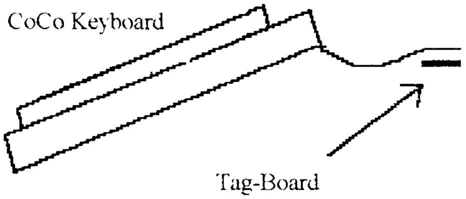
2. Remove the circuit board from a discarded ROM Pak and cut off the part of the circuit board that plugs into the computer, using a hack saw. You only need 16 conductors on one side of the circuit board section that you cut off from the ROM Pak, so trim the length of the section to 11/16ths. The overall dimensions should be 11/16ths by 1/2 inch.



3. Use a file to reduce the thickness of the circuit board section without damaging the 16 conductors on one side. Be careful to file only a 1/4 inch wide strip along one edge of the circuit board to a thickness that will permit the circuit board to insert easily into the keyboard jack on the Color Computer.

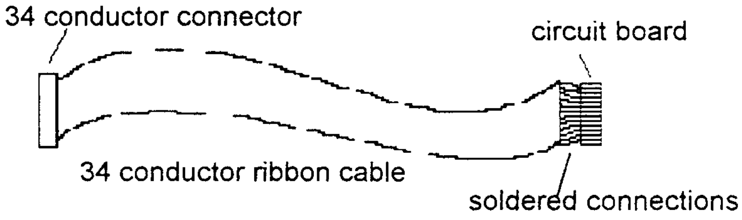
4. Strip 1/4 inch of insulation off the individual conductors. Decide which side of the 34 card edge connector will be the "Top Side" and then use an ohm meter or continuity checker to identify the 16 conductors that match up with the keyboard mylar cable when it is plugged into the card edge connector. Tin the exposed ends of the selected conductors and trim down the unused wires so that they are out of the way. Finally solder the tinned wire ends to the circuit board section prepared in step 3 above. Again use an ohm meter or continuity checker to see that the keyboard mylar cable matches the same position on the circuit board section: 1 = 1, 2 = 2, 3 = 3, etc.

5. Use *Elmer's Glue* to attach a piece of tag-board or thin card-board (such as used to package electronic components) under the mylar keyboard cable so that the combined thickness of the mylar cable and the tag-board will provide a solid contact when inserted into the other 34 card edge connector.



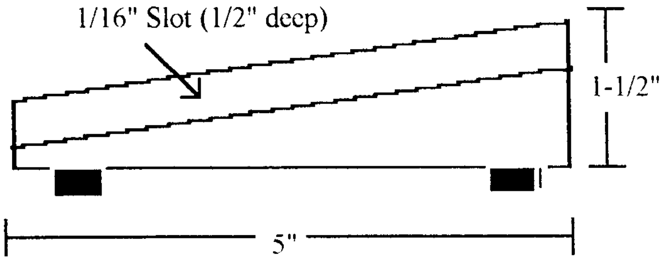
TESTING:

With the 16 conductor circuit board soldered onto one end of the cable and then plugged into the Color Computer Keyboard Jack, and the Keyboard plugged into the other end of the cable, turn on the computer and see if you can "type" to the computer. If there is no keyboard response, use a continuity checker and trace each of the 16 conductors from the keyboard jack on the mother board to the matching pin at the keyboard end of the cable.



FINAL TOUCH:

You may choose to use the CoCo Case to mount your extended keyboard, OR you may choose to build an enclosure for your keyboard. Even more simple is a pair of wood blocks mounted to each end of the keyboard. They provide functional support at the proper angle and a sharp appearance.



End View

The dimensions are 5 inches long, 3/4 inch wide, 1-1/2 inch high at one end and 3/4 inch high at the other end. Cut a 1/2 inch slot, 1/16 inch wide and 1/2 inch from the top angle cut. The blocks will press fit on the end of the keyboard and can be finished with varnish, varathane or paint. With the extra wires , you can add an LED Power-On light and even a Reset switch. Self adhesive rubber chassis feet may be attached to the bottom of the wooden blocks to provide a secure non slip grip. A second slot can be cut parallel to the first slot about a 1/4 inch lower lower so that a thin sheet of metal or plastic can be inserted as a bottom plate to cover up the wiring. The overall appearance is neat and a very small foot print on your desk top.

Cool your RAM with a FAN

Installing 512K of Random Access Memory opens up a whole new world of OS9. The memory barriers are gone, but two consequences must be considered. The added memory chips takes it's toll on the power supply and those same 16 chips generate a tremendous amount of heat which only reenforces the power supply problem, since the added power requirements will result in increased heat from the power supply transformer. The temperature inside your CoCo case can get so hot that one or more of your RAM chips may fail. At the very least, the life of your RAM chips will certainly be reduced. Obviously a FAN is required to increase the aif flow inside the computer's case, specifically in the area around the RAM cihps and the Power Supply.

Radio Shack sells a 12 volt D.C. Brushless Micro Fan (Catalog # 273-244) for $17.95. It draws only 150 milliamperes and is only 1-9/16 x 1-9/16 x 13/32. The fan fits perfectly between the ribs of the ventalation slots in the top of the case directly above the 512K RAM Board. Use double sided adhesive tape to provide shock absorbtion as well as increased mounting strength. Connect the red wire from the fan to the cathod side of either diode D1 or D2. Connect the black wire from the fan to any convenient ground potential on the CoCo's circuit board. That's IT!

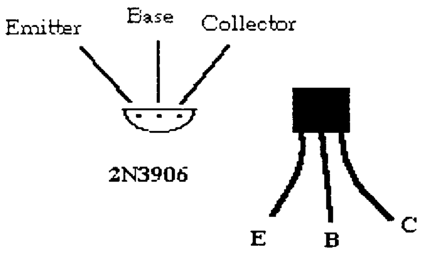
Even with the extra burden of the 512K RAM chips, the Power Supply seems to be able to support the additional 150 milliampers needed by the Micro Fan. I assume the cooling effect more than makes up for the difference.

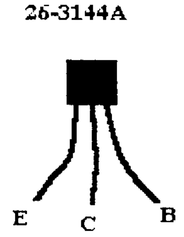
Speech Sound Pak "fix"

One of the major advantages of OS-9 Level Two is that it operates at twice the normal clock speed that the CoCo-3 uses under Disk Extended Basic. Unfortunately, the Speech Sound Pak made by Radio Shack was not designed to work at this higher clock frequency. The modifications to fix this problem are in two parts.

The hardware modifications to the Speech Sound Pak consist of replacing the one and only transistor with a 2N3906 PNP transistor and then cutting the trace line to card-edge pin 7 and finally connecting pin 9 of the 74LS86 to ground.

There is some variation to the above paragraph depending on which Speech Sound Pak you are modifying. The two different models are labeld as Catalog # 26-3144 and Catalog # 26-3144A.





The modification to the 26-3144 is the simplest since the 2N2907 has the same pin out as the 2N3906. The modification to the 26-3144A is slightly different because the 2N2907 pin outs are different than the 2N3906. (See diagram). When installing the 2N3906 transistor into the 26-3144A Multipak, you must swap the Base and Collector pins so that they match the original transistor's pin out.

EDITOR'S NOTE: The new descriptor (SSPak) and driver (SSP) for the Speech Sound Pak for OS-9 Level Two are found on the reverse side of the Tutorial Disk.

Make an RS-232 Pak from a ModemPak

by Wes Gale

I found this a great way to get a piece of hardware that I could not find anywhere (RS-232) from some piece of junk I though was completely useless (the ModemPak)

Parts List:

1 Direct Connect Modem Pak
1 ICL232 CMOS chip
4 22 microfarad electrolytic capacitors
1 male d-sub 25 connector
1 small bread board

Depending on what kind of setup you have now, you may also need a new cable for this. If you were using the bit banger port before, then you'll also need a female d-sub 25 connector to change your present cable or for around $15 you could buy one.

If you want all the lines for a TRUE RS232 port, you'll need two of the ICL232 CMOS chips. However, one will do the job just fine giving you transmit, receive data, carrier detect and the data terminal ready lines. I chose this I.C. because it does not need +12vdc, and -12vdc to operate, and has send and receive buffers in one chip. The RS232 Pak sold by Tandy required a Multi-Pak or a Y-Cable with a power supply. So, as far as I know, this conversion will not need a Multi-Pak.

The hardest part is to first strip everything from the old Modem-Pak board EXCEPT the following parts:

IC2:	6551A Integrated Circuit Chip
IC3:	74LS133 Integrated Circuit Chip
IC4:	74Ls04 Integrated Circuit Chip
X1:	1.8432 MHz Crystal
C3, C11, C12, C13, C18:	.1 ufd Capacitors
C1:	100 ufd Capacitors
R1:	4.7K Resistor

And the diode on the back side of the board that's connected to pin 27 of the 6551 (IC2)

NOTE: If you 're having trouble getting things out, just cut the traces that go anywhere from that part.

To change the addressing so the Modem-Pak will be enabled when the RS232-Pak is addressed:

1. Cut the trace going to pin 2 of IC2 (6551A).
2. Run a wire from that trace (the first through board connection that was from pin 2 of IC2) and connect it to pin 1 of IC4 (74LS04)
3. Take a wire from pin 2 of IC4 to pin 2 of IC2.

Now, any hardware looking for an RS232 Pak will see the Modem-Pak as such.

Now we have to fix up a couple things:

1. Cut the trace from pin 16 of IC2
2. Run a wire from pin 1 of IC2 to pin 9 of IC2

Now comes the last step. This can be done on a small bread board, or as I did using the existing holes in the board where IC5 (74HC943) used to be. I made sure no traces were left connected to any of the holes, and soldered a socket in there, from there I ran my wires. This may be difficult if you do not have a de-soldering pump to clean out the holes, but it leaves lots of room. The first one I converted had a small bread board crammed in there with a rat's nest of wires all over the place, not very aesthetic, but effective..

The pinouts of the ICL232 are as follows:

```
            __ __
C1+     1|   U  |16    Vcc
V+      2|      |15    Gnd
C1-     3|      |14    T1 Out
C2+     4|      |13    R1 In
C2-     5|      |12    R1 Out
V-      6|      |11    T1 In
T2 Out  7|      |10    T2 In
R2 In   8|______|9     R2 Out
```

1. Run a wire from the + side of C1 to Pin 16 of the ICL232
2. Put a 22 mfd capacitor across -pins 1 and 3 of the ICL232 (- side to pin 3) pins 4 and 5 of the ICL232 (- side to pin 5) pin 6 and gnd pin 2 and 16.
3. Make the following connections:

IC2	to	ICL232
pin 11		pin 10
pin 10		pin 11
pin 12		pin 12
pin 16		pin 9

4. Make the following connections:

ICL232	to	d-sub 25
pin 14		pin 2
pin 7		pin 20
pin 13		pin 3
pin 8		pin 8

5. Connect pins 1 and 7 of the d-sub 25 connector and run a wire from them to ground

That's it, put it back together, and try it out. It should work in both OS9 and RSDOS.

Four Double Sided Drives on a CoCo Under OS-9

By Bob Devries

This article originally appeared in the Australian OS9 Newsletter of March, 1991.

Here are the details of modifications to my Disto/CRC SCII controller to enable it to use four floppies. The modifications should also work on other controllers, BUT NO GUARANTEES are implied. You're on your own!

There are not four drive select lines, only three and the side select line. Well, that is true, and I had to do a bit of hardware hacking (my main source of enjoyment with the CoCo. After C programming of course!). Well, I'll tell you about how I did the mods. Let me first say that if you decide to try it for yourself, you're on your own! I will not be responsible for any damage.

The disk controller I have is a CRC-Disto Super controller II (with 4-in-1 fitted, but that's by-the-way). Now I would imagine that the mods should be able to be done to any disk controller, but again, NO GUARANTEES.

CABLE:

I used a 23 pin DB connector to replace the 34 pin card edge connector at the end of the cable that you plug into the controller. I wired it along the lines of the Amiga A500 external drive connector. Here is the pinout I used.

DB23 pins	Standard pins
1. Ready (not used on COCO)	(34)
2. Read Data	(30)
3 - 7 Ground	(all odd pins)
8. Motor on	(16)
9. Drive select 2	(14)
10. Disk reset	(not used on COCO)
11. Disk change	(not used on COCO)
12. +5 Volt supply	
13. Side select	(32)
14. Write protect	(28)
15. Track 00	(26)
16. Write Gate	(24)
17. Write Data	(22)
18. Step	(20)
19. Direction	(18)
20.Drive select 3 (not currently connected - read on)	(6)
21. Drive select 1(already used for 5 1/4 drives)	(12)
22. Index	(8)
23. +12 Volt supply	

CONTROLLER:

OK so here's how I modified the controller. Firstly, you'll need a 74LS138 chip, as well as three 10k ohm 1/4 watt or 1/8 watt resistors. Here's the circuit:

```
                              74LS138
controller drive sel  0   1 |--U--|16   to +5 volt
   "         "    "   1   2 |     |15   to drive sel 3 (pin 6)
   "         "    "   2   3 |     |14   not connected
   connect to ground      4 |     |13   to drive sel 1 (pin 12)
   connect to ground      5 |     |12   not connected
   connect to +5 volt     6 |     |11   to drive sel 2 (pin 14)
   not connected          7 |     |10   not connected
   connect to 0 volt      8 |_____|9    to drive sel 0 (pin 10)
```

The three resistors must be connected one end to +5 volt, and the other ends to pins 1, 2 and 3 respectively.

This little circuit may be connected either in your disk drive case, or in the controller itself, where-ever there is sufficient space. I mounted mine inside the controller, placing the chip piggy-backed on top of another chip of similar size, with all the pins except pins 16, and 8 bent out sideways. I soldered these two pins to the chip underneath, making sure of two things: (1) that the chip underneath was another 74LSXXX chip or 74XX chip (e.g. 7416), and (2), that the piggy-backed chip was the same way 'round. Well, OK, I also checked that I did not blob the solder all over the other chips, but that's normal for any electronics practices.

To make the connection to the circuit, I cut the traces just before the connection to the 34 way edge connector where the drive cable plugs in. Only three need to be cut, pins 10, 12, and 14. There should be no connection to pin 6, and the wire from pin 15 of the 74LS138 chip may be connected here. By the way, I used 28 guage 'Kynar' wire wrap (any fairly thin gauge wire would do).

SOFTWARE:

OK, so now how do you do the modificatin? OS9 needs to have a module patched. The module is CC3Disk. I strongly suggest that you use 'DED' to do the modification. Find the character combination that goes like this (in HEX) 01020440. This is the drive select table bit pattern. Change the byte 40 to 07. Don't forget to write the new module to disk, and verify to correct the CRC. You can do this from within DED. Please read the DED docs FIRST, and do it on a BACKUP of your normal system disk. Next, you'll need to add two new device descriptors /D2 and /D3 to your OS9Boot file. You will of course do this in the normal way (whatever that is). The /D3 descriptor you'll have to create yourself, or modify the one on the Boot/config disk named d2_35s.dd. Here is a dump of one of my descriptors to show you what they look like:

This is the one for /D2

```
ADDR  0 1  2 3  4 5  6 7  8 9  A B  C D  E F   0 2 4 6 8 A C E
---- ---- ---- ---- ---- ---- ---- ---- ----  ----------------
0000 87CD 0030 0021 F181 D400 2300 26FF 07FF  .M.0.!q.T.#.&_._
0010 400F 0102 0320 0300 5002 0000 1200 1203  @.... ..P.......
0020 0844 B252 42C6 4343 3344 6973 6B83 4376  .D2RBFCC3Dis;.Cv
```

Don't forget... I am using the CRC-Disto Super controller II in NO-Halt mode. The descriptors may be different from yours! CHECK THIS FIRST.

1.2 and 1.4 High Density Floppies on your CoCo

Instruction for modifying the ORIGINAL RADIO SHACK FLOPPY CONTROLLER

The controller MUST be the one with the full sized board, a 1793 controller chip and three adjusting potentiometers. According to the Western Digital manual, the 1773 (used in the newer controllers) CANNOT do high density.

This modification is NOT for the faint of heart or those unexperienced with hardware modifications. If you don't know what "piggyback" means when refering to chips, forget it! This modification requires 32 soldering connections, 18 jumper wires and a lot of patience. Do this on you old spare controller if you can. The old controller needs 12 volts therefore you MUST have a multipak or equivalent. This modification will allow the controller to use either 250 kbs or 500 kbs data transfer rate. This is the difference between the standard 5.25" 360k or 3.5" 720k drives and a 5.25" 1.2 meg or 3.5" 1.4 meg drive.

WHAT YOU NEED:

1 74LS74
1 74LS158
1 3.9k 1/4 watt resistor
1 mini DPDT toggle switch (optional)

Wire for the jumpers. (I recommend standard wire wrap wire as RS carries. This is very important, DO NOT use thick wire. Wire wrap wire is 30 gauge. Just right for these kind of projects.)

The mod will be done so if a mistake is made and you want to abandon it, you can just remove all of the jumpers plug in replacement chips for the ones piggybacked to and you'll be back to where you started. If you want this option, buy an extra 74LS74 and a 74LS221. There are NO trace cuts in this mod. IC pins are left out of the socket to get the equivalent of a trace cut. If you need to reverse the mod, those pins MUST be reinserted into their respective sockets. There is ABSOLUTLY NO GUARANTEE OR WARANTEE EXPRESSED or IMPLIED FOR THIS MODIFICATION. Now, on to the fun part!!

We will be piggybacking a 74LS74 on to the existing 74LS74 at IC1. We will also be piggybacking a 74LS158 onto the 74LS221 at IC7. Some other chips will be soldered to and some pins will be removed from the sockets for some IC's. These instructions will be entirely verbal, no illustrations.

First, remove U1 (74LS74) from it's socket. Position a new 74LS74 on top of it with the pins EXACTLY overlapping (this is called piggybacking). Be sure both pin 1's are lined up or it'll be poof time when you apply the power. On the upper 74LS74, bend up pins 2,3,5,6,8,9,10,11,12 and 13 so they point directly

away from the body of the IC. Pins 1,4,7 and 14 should still be overlapping the lower 74LS74. Carefully solder these pairs of pins together being careful not to blob the solder onto the legs of the lower 74LS74 as you will be plugging the pair (stack) of chips back into the U1 socket when done. On the lower 74LS74, bend pin 11 out away from the body of the chip as you did for some of the pins on the upper IC. Pin 11 will NOT be going back into the socket. Prepare six 3" jumper wires (prepare means strip back the insulation on each end of the wire, no more than 1/16". Then tin the exposed wire on each end of the jumper). Solder the wires to the stacked IC's as follows:
One end of each wire will be unconnected.
1 jumper to pin 11 on the lower IC (the pin sticking out)
1 jumper to the lower IC pin 3 (must still be able to go into the socket)
1 jumper to the lower IC pin 6 (must also be able to go back into the socket)
2 jumpers the the upper IC pin 3
1 jumper to the upper IC pin 6

Also, prepare a 1.5" wire and solder it from the upper IC pin 2 to the upper IC pin 6 taking care not to disconnect the wire already on the upper IC pin 6. You may now carefully plug the IC stack back into the IC1 socket making sure all pins get seated into the socket with the exception of pin 11.

Second, we'll be doing a similar piggyback mod to the 74LS221 in the U7 socket Remove the 74LS221 from the socket. Position the 74LS158 on top of the 74LS221 Make sure that the two IC's are properly aligned and that the two pin 1's are aligned together. Bend up all of the pins on the upper IC EXCEPT pins 8 and 16. solder the two pin 16's together and also solder the two pin 8's together. As before, make sure not to blob solder on the legs as the stack will be plugged back into the U7 socket. Bend pin 13 on the lower IC away from the body of the IC so it cannot be reinserted into the socket. Prepare four 1.5" jumpers, one 2" jumper and one 3" jumper. Solder them in as follows:
1 2" jumper to the lower IC pin 2
1 1.5" jumper from the joined pin 8's to the upper IC pin 15
1 1.5" jumper from the upper IC pin 15 to the upper IC pin 10 (taking care to not disconnect the wire already at pin 15)
1 1.5" jumper from the tied together pin 16's to the upper IC pin 11
1 1.5" jumper to the upper IC pin 7
1 3" jumper to the upper IC pin 1

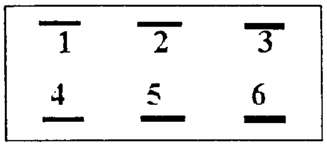
Plug the stack back ino the U7 socket making sure all of the pins are seated firmly EXCEPT pin 13 which should be sticking out. Solder a 3.9k resistor from the upper IC pin 9 to the side of R18 (3.9k) which is the closest to the U7 socket.

Final Assembly

Remove U11 (the 74LS629). Solder one of the 3" jumper wires from U1, the upper 74LS74 pin 3 to the top of the 74LS629 pin 7 making sure not to blob solder. Plug U11 back in making sure ALL of the legs seat firmly into the socket. Unplug U3 (7406 or 7416). Connect the 2" wire from the lower IC pin 2 of the stack at U7 to the top of pin 1 of the IC that was in U3 (making sure not to blob solder on the leg). Plug U3 back into it's socket making sure all of the legs seat firmly into the socket.
Solder the open end of the jumper connected to U1 lower IC pin 11 to U7 upper IC pin 4
Solder the open end of the jumper connected to U1 lower IC pin 3 to U7 upper IC pin 3
Solder the open end of the jumper connected to U1 lower IC pin 6 to U7 upper IC pin 2
Solder the open end of the jumper connected to U1 upper IC pin 6 to U7 upper IC pin 5
Solder the open end of the jumper connected to U1 upper IC pin 3 to U7 upper IC pin 6

Choose 1 of the following select methods:

Select option 1 - using WRITE PRECOMP bit and a SWITCH (For Hard Drive booting systems) Mount the dpdt mini switch somewhere handy. I mounted mine in the hole near C1 and the piggybacked 74LS74's. Make sure that the switch DOESN'T SHORT OUT any traces! I'll refer to the switch pins as follows:



1 2 3 pin 2 toggles between pins 1 & 3
4 5 6 pin 5 toggles between pins 4 & 6

Carefully remove U12 (the 1691)from its socket. Bend up pins 9 and 16 away from the body. Put the 1691 back into the U12 socket making sure that all pins firmly seat with the exceptions of pins 9 & 16.

Prepare and solder a 4" jumper from U12 (1691) pin 9 to the DPDT switch pin 5

Solder the open end of the jumper connected to U7 upper IC pin 7 to U12 pin 16

Solder the open end of the jumper connected to U7 upper IC pin 1 to the DPDT switch pin 2

Prepare and solder a short jumper from the DPDT switch pin 3 to the DPDT switch pin 4

Prepare and solder a short jumper from the DPDT switch pin 4 (taking care to not disconnect the wire already there) to a convenient ground (for example, IC1 pin 7 on the SOLDER side of the board)

Prepare and solder a short jumper from the DPDT switch pin 1 to the DPDT switch pin 6

Remove U8 (the MC14174) and prepare a 3.5" jumper. Solder a wire to the top of pin 12 without blobing solder on the leg. Plug U8 back in making sure all of the pins seat firmly into the socket.

Solder the open end of the jumper connected to U8 pin 12 to the DPDT switch pin 1 taking care not to disconnect the wire already there.

Skip to check procedure below.

Select option 2 - using a DRIVE SELECT BIT

Carefully remove U12 (the 1691) from its socket. Bend up pin 16 away from the body. Put the 1691 back into the U12 socket making sure that all pins firmly seat with the exception of pin 16.

Solder the open end of the jumper connected to U7 upper IC pin 7 to U12 pin 16

Remove U2 (7406) from the socket. Choose a drive select line to use, either DS1 or DS2 (DS0 should not be used or you will not be able to boot, DS3 is usually used to access the back side of double sided drives so that cannot be used either). Solder a 2" jumper to pin 3 (DS1) OR pin 5 (DS2) without blobing solder on the leg. Plug U2 back into it's socket making sure all pins seat firmly.

Solder the open end of the jumper just attached at U2 to U7 Upper IC pin 1.

CHECK PROCEDURE:

Now recheck the entire procedure to make sure no mistakes were made. Check all soldering joints for good connections. Check for shorts, especially by the DPDT switch. There should be NO unconnected jumper wires! If there are, go through the entire sequence to see what you missed. Now, we need to calibrate and test the controller.

Use a multipak which will protect the CPU (you need +12 anyway) in case you made a fatal wiring mistake. Plug the controller into slot 4 as usual. Power on the multipak, then the computer. If the DISK BASIC message doesn't come up quickly then shut the computer off immediatly and power everything off. Unplug the controller and check for shorts and recheck all connections against the modification procedure. If all else fails, you can always remove the piggybacked stacks at U1 and U7, carefully pull off all of the jumpers, insert a new 74LS74 into U1 and a new 74LS221 into U7, pull out U12, carefully bend pins 9 and 16 back down and reinsert it into it's socket, remove the switch and you'll be back to where you started.

Presuming you made it past the smoke test, you will need to figure out your switch position and calibrate the controller.

SWITCH POSITION DETERMINATION (Skip if drive select method was chosen)
When the switch is in the position such that pins 1 & 2 are connected together (also pins 4 & 5) the controller is in the HIGH DENSITY enabled position (use a meter to test the connection between pins 1 & 2). When the switch is the other way, the normal configuration is active, which means write precomp is available. Put the switch into normal position for calibration.

CALIBRATION OF THE VCO:
The controller can be calibrated either with a scope or by trial and error. Either way, mark the original position of R8 so you can reverse the modification if you can not get it to work right.

If using a scope, connect the scope to the VCO output of the 74LS629 (U11) pin 7 and adjust R8 for 4 mhz. If doing the adjustment by trial and error, put a formatted RSDOS disk into drive 0 and do a DIR from RSDOS. Turn R8 until you can get a directory. You may have to do lots of DIR commands. Try to find the extreme settings of R8 that will still produce a directory, then set R8 between the two extreme settings. The range in which the DIR will work will be quite small and your final setting for R8 should be as close as possible to the middle of the range. THAT'S IT FOR THE HARDWARE.

To complete the modification you need to apply my IPATCH file "***cc3diskhigh.ipc***" to the ORIGINAL Radio Shack ***cc3disk*** edition 9 (CRC $759161). You will also need one or both of the following disk descriptors: ***dl_1.2.dd*** (high density 5-1/4 inch drive) and ***dl_1.4.dd***.(high density 3-1/2 inch drive) The patched ***cc3disk*** detects the old 8inch drive bit in IT.TYP in the drive descriptor and uses it to switch the data transfer rate.

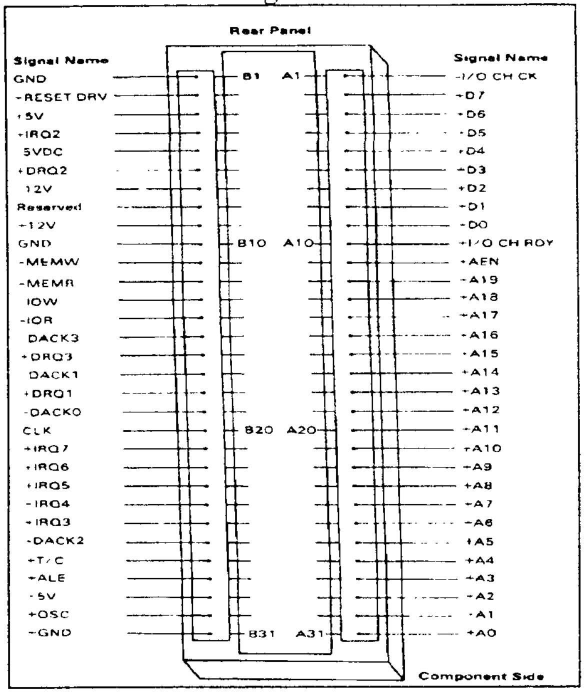
Make a new OS9 boot with the new ***CC3Disk*** and the appropriate drive descriptor(s) to match your high density drive(s). After booting put the switch (option 1) into the HIGH DENSITY ENABLED position and you're ready to go. For use in RSDOS, the switch (option 1) should be in the normal position. Note, the high density drive is not usuable in RSDOS.

EDITOR'S NOTE: The IPATCH file *(cc3diskhigh.ipc)* and the high density drive descriptors *(dl_1.2.dd* and *dl_1.4.dd)* are all in an archived file called **CC3DIS.AR** found on the reverse side of the Tutorial Disk.

CoCo to IBM Interface

by Pat Pleuard

Fig. 1



The following diagrams should provide you with the necessary information needed to construct a "CoCo to IBM Bus Interface" so that you can plug an IBM-XT type Hard Drive Controller (MFM or RLL type) to one end of the interface and then plug the interface into a Multipak Interface.

Figure 1 shows an IBM bus socket with the bus pin outs and Signal names that an XT type Hard Drive controller expects to plug into.

Figure 2 is the circuit diagram for the interface card. It's very simple and can be accomplished using point to point wiring with 30 guage wire such as that used for wire wrapping (R.S. Cat.# 278-501). Only two integrated circuits are used: 74LS32 Quad 2-Input OR Gates, and a 74LS04 Hex Inverter. Both chips are very inexpensive and readily available from any electronic supply store. The most difficult part of the construction is soldering the 62 position PC/XT Bus Card-Edge Connector (R.S. Cat.# 276-1453) to the end of the circuit board.

CONSTRUCTION:

To simplify the construction, use a pre-punched plug-in circuit board available from Radio Shack (Cat.# 266-192). This board has 72 position card edge connectors, we only need 40, so carefully cut off the unnecessary connector traces making sure that the remaining 40 traces will line up properly when plugged into the Multi-Pak. Check out the CoCo Bus Pin labels in Figure 2 so that you will properly identify the card edge traces. Mount the three IC's in a convenient location on the circuit board, then carefully solder the 30 guage wire to accomplish the connections as shown in the circuit diagram (Figure 2).

Solder in the integrated circuits or use IC sockets if you wish. Deposit a drop of solder at each desired contact point on the circuit board before you actually start wiring. The insulation on the wire is very thin and heat sensitive. Heat each drop of solder, then bring the wire in contact to the hot solder. The insulation will automatically strip away from the contact points making a very neat and easy connection. Use this method to quickly wire your circuit board point to point. BUT, always check each connection with a continuity testor or ohm meter to insure a good connection.

Parts List

Plug in PC Board with 0.1" contact centers
(Radio Shack Cat. # 276-192 at $4.95)
62 Position Card-Edge Connector
(Radio Shcck Cat. # 276-1453 at $2.95)
2 - 74LS04 Hex Inverters (approximately $0.49)
74LS32 Quad 2-Input OR Gate (approx. $0.49)

The only remaining hardware is an XT-MFM or RLL Controller Card which retails for about $50. I've also purchased several cards recently at the AM Computer Swap Meet for $20. Specifically you are looking for a Western Digital **WD1002** series or the **WDXT-GEN** Controller Card. You can also use the **DTC 5150CRH** and **5160CRH** or Adaptec's **2072.** Just make sure that if you are going to use an RLL Hard Drive that you use an RLL type controller. Some MFM Hard Drives can be made to work with an RLL Controller, but you're pushing it! (See Burke & Burke's CoCo XT Manual, pages A16-A20 for a more detailed explanation)

SOFTWARE:

This project is almost too good to be true. For less than $10 in parts you can build your own hard drive interface that looks very similar to the Burke and Burke Interface, and in fact it operates under the Burke and Burke Software. How convenient. But then again.....that's the catch!

Although the circuitry is quite different from Burke and Burke's **COCO-XT** Hard Drive Interface, the results are the same. In order to set up your hard drive descriptor and driver(s) you will need to purchase Burke and Burke's **CoCo XT & CoCo XT-RTC** Software and Documentation Package. However, be aware that B&B's software package is copyrighted and Chris Burke spells it out in very clear terms on the first page of his manual. And I quote........

In other words, you will have to purchase the software from Burke and Burke (P.O. Box 733 Maple Valley, WA 98038. 1-800-237-2409 or 206-432-1814). Even if you already have a hard drive system using Burke & Burke's Interface, you must still purchase a 2nd software package to operate Pat Pleuard's interface. Burke & Burke will sell the software separately for about $10.

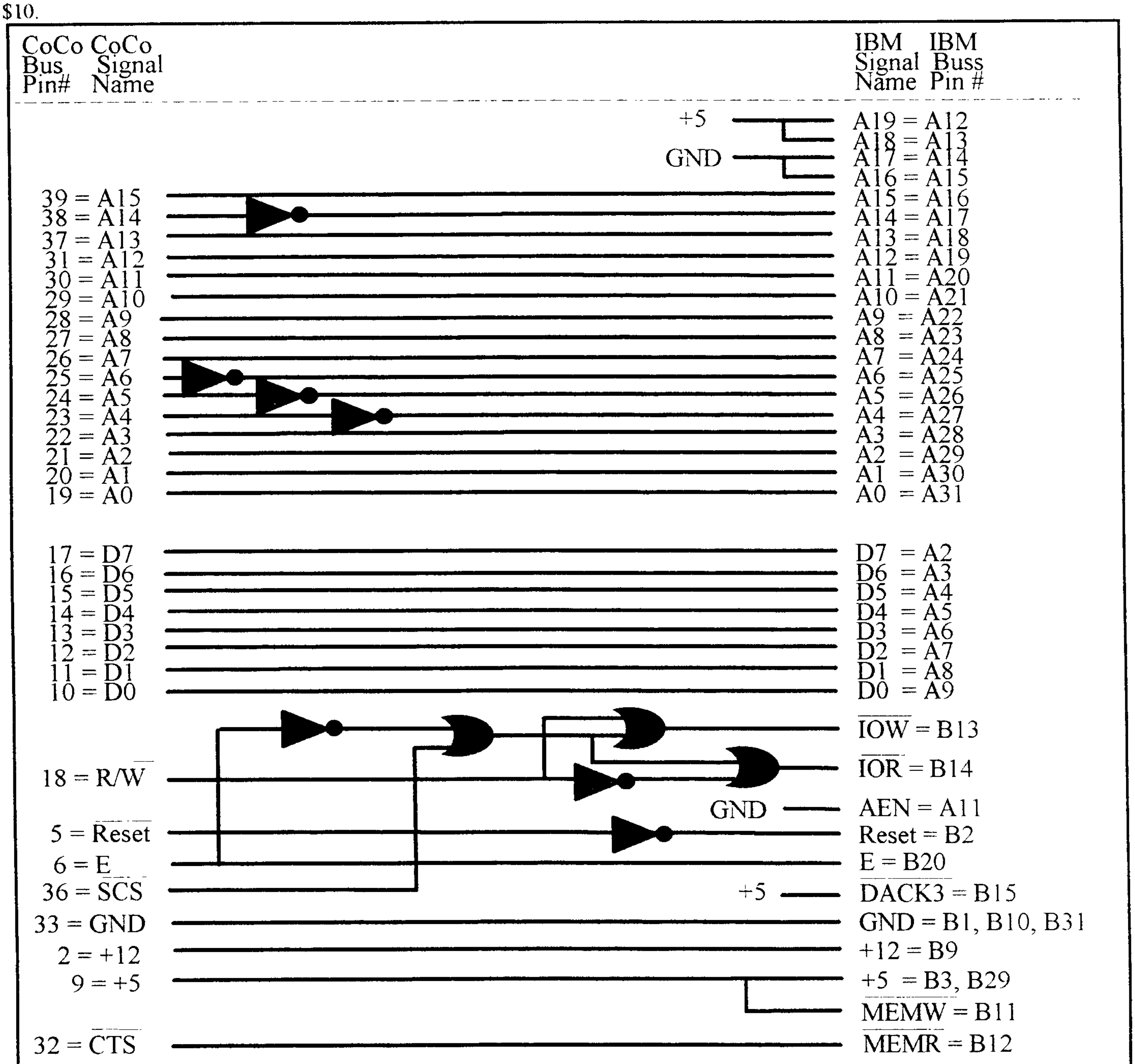


Figure 2

-- Rodger Alexander --

The "*Correct All*" Fix for the CoCo-3

The CoCo III has a slight timing problem with SCS/CTS/ROM selects. This is due to IC9 (74LS138) being selected (i.e. transiting) during an entire E-clock cycle instead of being gated to the READ portion. The tech manual quotes "Due to the nature of the ROMs and in order to prevent data bus contention, the ROMs are enabled only during the E portion of a read cycle." pp 36. This is blatantly false when you look at IC9. It has both *G2A and *G2B and G1 mux selects tied wide open! The fix came from the Motorola 8-bit Microprocessor Handbook Data Sheets on the MC6883 SAM (GIME's father :-)).

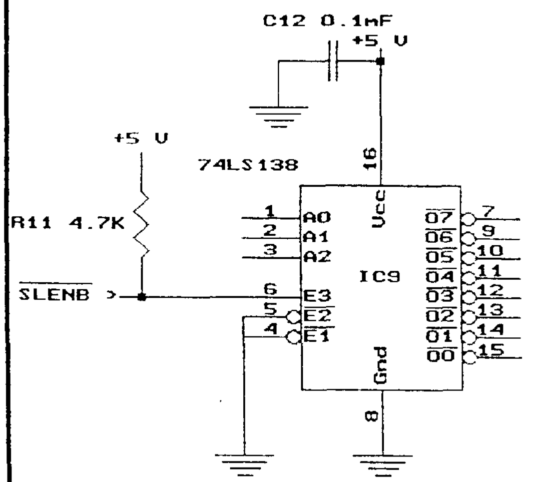
Take a 74LS02 NOR gate and tie one input to pin 3 of IC9. Cut the connection between *G2A of IC9 (pin 4) and ground. Tie the NOR gate output to *G2A of IC9 (pin 4). Tie the other NOR gate input to the E-clock output at the intersection of R9 (47ohm resistor) and C10 (39pF capacitor). R9 is coming from pin 6 of the GIME chip. Make sure you tie pin 14 of the 74LS02 to pin 16 of IC9 (+5v) and pin 7 of 74LS02 to pin 8 of IC9 (Gnd).

This is not something for the weak of heart to try! At the very least you will have to remove the motherboard from the case (unless you clip IC9's little leg!) and cut a trace on the bottom of the mother-board. I cut/desoldered IC9, replaced it with a socket and built a little daughterboard which held a new IC9 plus 74LS02.

This fix resulted in the following improvements on my system:
(1) Sparklies disappeared on graphics screens under OS9. (GIME chip circa 1986)
(2) Before the fix, I had blob problems (expr. yeilded ~1 in 4). I no longer have the BLOB.
(3) My Performance Perp. No Halt controller wouldn't run reliably in No Halt mode with the Burke & Burke RTC Hard Disk interface. They now coexist peacefully with the fix.

Note: *An article in "The Rainbow" said that Disto products "depended"? upon the timing not corresponding to the E clock. My personal opinion is:* ***bull twinkies!*** *As far as I know all Disto products should work with CoCo's 1 and 2 which do have the proper E-clock gating. Remember, that is just my opinion and all Disto users should be prepared for this fix not working. (If you put a socket in and build the daughterboard you can have it either way...)*

Charles C. Bundy IV
Internet: <COCO@PUCC>



Original Sehematic: **INCORRECT**

Figure 1

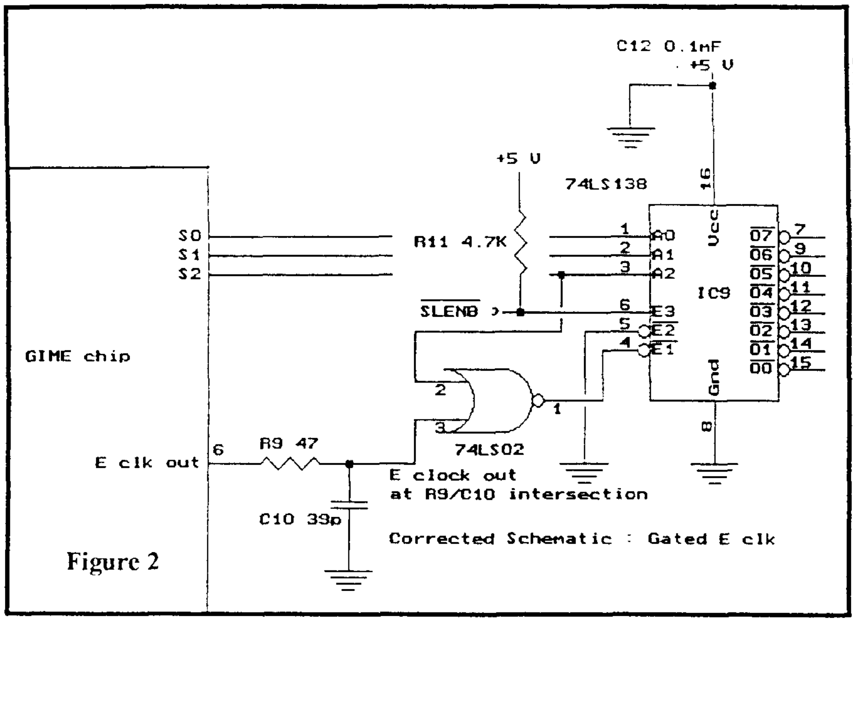


Figure 2

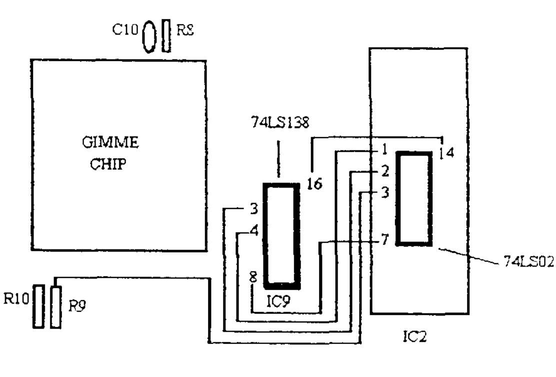


Figure 3

Simple Switch for your Hi-Res Interface

by T. Warren

Isn't it aggravating every time you have to switch between "HI" and "LOW" resolution joystick. It sure would make computer life easier if programs gave you a choice of either LOW or HI resolution joystick operation (Actually some programs/games do?). Better yet.... a simple switch to access either HI or LOW resolution operation without disabling the casette port.

MODIFICATION THEORY:

Take a look at figure 1 to see the standard joystick wiring to the color computer and compare it to figure 2 which is the High Resolution Joystick connection to the color computer. Notice that pin 1 and pin 2 are interrupted by the integrated circuit (IC) in the HI-RES Interface. By simply switching the IC in and out of the circuit it's possible to have both LOW and HI resolution operation by the flip of a switch.

Also notice in figure 2 that the 5-pin jack is the casette port and that pin 5 is monitored by the IC in the hi-res interface. It's not necessary to disconnect the casette tape recorder during hi-res operation.

Simply put, we want to modify the Hi-Resolution Joystick Interface to switch between LO and HI resolution operation. This can be accomplished by using a double-pole double-throw mini switch (Radio Shack Cat.# 275-407) mounted on the top of the interface box. The switch will route pin 1 (C-LR) and pin 2 (C-UD) of the joystick port around the Hi-Res joystick integrated circuit (IC). Also note that it is possible to continue the casette lines through the interface box with a 5 pin DIN inline female jack (Radio Shack Cat.# 274-006).

MODIFICATION INSTRUCTION:

1 Open the Hi-Res Interface box by removing the four retaining screws.
2. Cut the two plastic wire tie straps to give you easier access to the individual wires.
3. Unsolder the red wire (C-LR) from the circuit board and solder it to one of the center connectors on the mini slide switch.
4. Solder a 2 to 3 inch length of insulated wire to the same hole in the circuit board from which you removed the red wire in step 3 above. Solder the other end of the wire to one of the end terminals on the mini slide switch on the same side that the red wire is connected to (See figure 5).
5. Unsolder the blue wire (C-UD) from the circuit board and solder it to the other center connector on the mini slide switch.

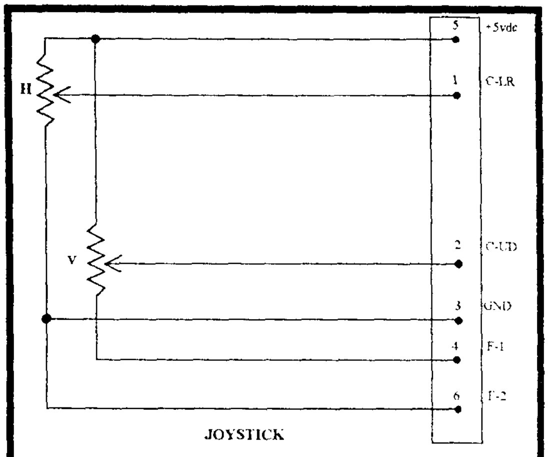


Figure 1

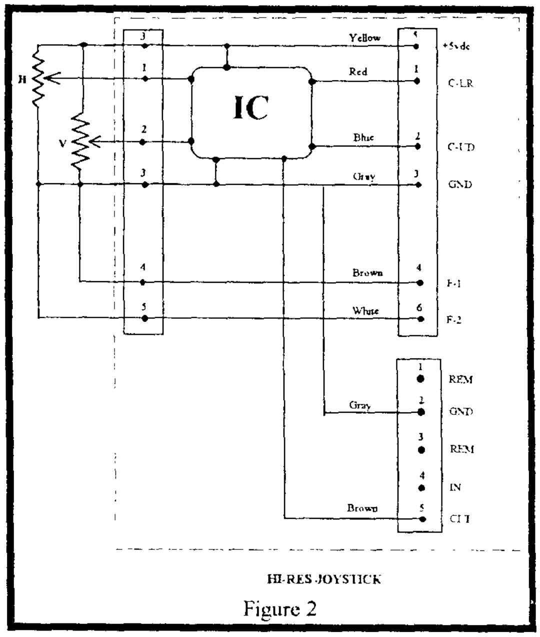


Figure 2

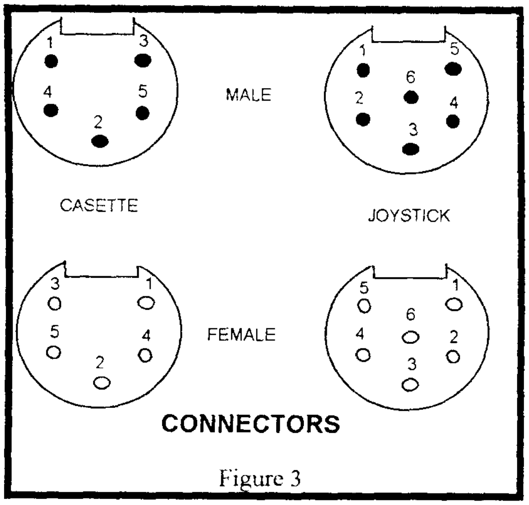


Figure 3

6. Solder another 2 to 3 inch length of insulated wire to the same hole in the circuit board from which you removed the blue wire in step 5 above. Solder the other end of the wire to the vacant end terminal on the mini slide switch next to the other wire in step 4 above (See figure 5).
7. Using an ohm meter or continuity checker, find the pin on the integrated circuit (IC) that is connected to pin 1 on the joystick socket mounted on the circuit board. Solder a 2 to 3 inch length of insulated wire to the identified pin on the IC, then solder the other end of the wire to the identified pin on the IC, then solder the other end of the wire to the remaining end terminal on the mini slide switch on the same side as the red wire referred to in step 2 above (See figure 5).
8. Using an ohm meter or continuity checker, find the pin on the IC that is connected to pin 2 on the joystick socket mounted on the circuit board. Solder a 2 to 3 inch length of insulated wire to the identified pin on the IC, then solder the other end of the wire to the remaining end terminal on the mini slide switch on the same side as the blue wire refrerred to in step 3 above (See figure 5).
9. Cut an appropriate length of 5 conductor cable and solder one end of the cable to a 5 pin DIN inline female jack. File an appropriate size whole in the Hi-Res Interface box 1/4 inch to either the left or right side of the large joystick jack hole (See figure 7).
10. Using an ohm meter or continuity checker identify where pin 1 of the casette jack is soldered to the circuit board then solder the pin 1 wire from your female DIN jack to the same point on the circuit board.
that you have continuity from the male casette jack that plugs into the color computer and the female DIN jack.
11. Repeat the procedure in step 10 above to connect the 4 remaining wires from your female DIN jack to the corresponding connections points on the circuit board so that you have continuity from the male casette jack that plugs into the color computer and the female DIN jack.
12. Secure the original casette and joystick cables with two small wire ties in their original holes. Drill a third tie down whole to line up your new casette cable with the hole you filed in the box in step 9 above. Secure your new casette cable with a small wire tie down strap (See figure 6).
13. Cut and drill appropriate size holes in the top of the Hi-Resolution Interface box to accomodate the mini slide switch. Mount the switch and close up the box securing it with it's 4 retaining screws (See figure 7).

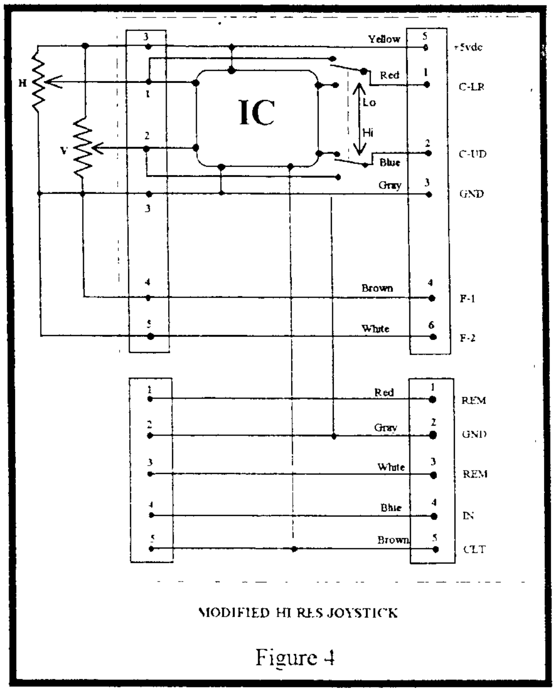


Figure 4

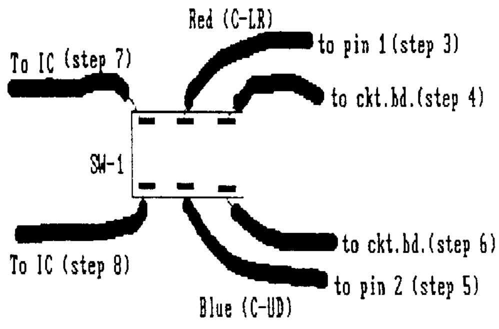
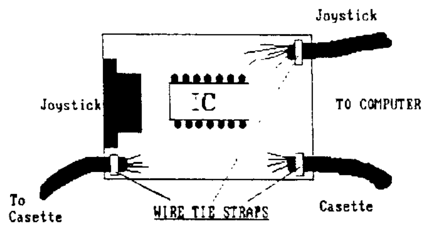


Figure 5



HI-RES INTERFACE CIRCUIT BOARD

Figure 6

SOFTWARE SWITCHING:

It is possible to switch between HI and LOW resolution via software. This can be achieved by sampling the casette out (pin-5) and feed the signal to a two stage operation amplifier (OP AMP), then filter out the alternating current (audio) component. The resultant DC voltage (5VDC) can be used to turn on a relay to function as the switch. In fact it is possible to mount a "mini" 5 volt D.C. relay (low current) inside the interface box. (See figure 8 for a schematic diagram of the software switch.

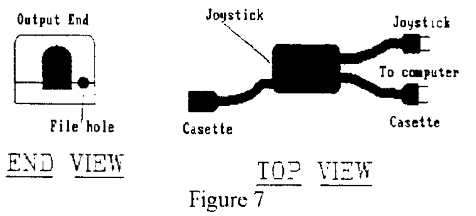
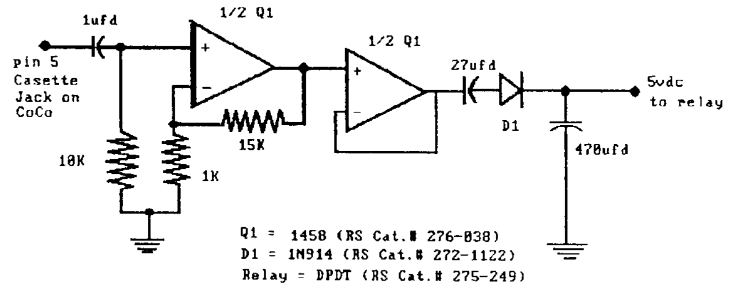


Figure 7

The software switch works because programs/games requiring the hi-resolution interface send out an audio signal through the casette port. When the audio signal is present, the relay closes, switching in the hi-res IC. When the audio signal stops, the relay opens and the joystick is returned to low-resolution.



Q1 = 1458 (RS Cat.# 276-038)
D1 = 1N914 (RS Cat.# 272-1122)
Relay = DPDT (RS Cat.# 275-249)

PARTS LIST

5 pin DIN inline female plug (Radio Shack Cat.# 274-006)
DPDT mini slide switch (Radio Shack Cat.# 275-407)
3 small wire ties
Insulated wire. Solder. Tools. etc.

NOTE: If you plan to build the software switch. specific parts are listed on the schematic diagram.

Installing the 6309

Whether you've bought a PowerBoost kit from Burke & Burke, or just want to upgrade the processor in your Color Computer, you may want to install a socket for the processor. This article describes how to install a processor socket in your Color Computer without unsoldering the old 6809 chip. By following these instructions, you can actually install the new processor socket directly on top of the old processor!

The 6809E TSC Pin

Pin 39 of the 6809E processor is called the TSC (Tri-State Control) pin. TSC was added to the 6809E processor to support DMA controllers, multi-processor configurations, and other shared bus configurations.

When pulled to ground (less than .5 volts), TSC allows the 6809E to operate normally. When pulled to a logic high level (more than 3 volts), TSC places the 6809E's address and data busses, as well as the R/W line, in a high-impedance state - nearly the electrical equivalent of unplugging the 6809E from its socket.

In the Color Computer, there's no DMA controller and the video controller uses a "hidden DMA" technique to access memory without slowing down the 6809E. Under these conditions there's no need for TSC; the Color Computer ties it directly to ground.

Using TSC to Advantage

By disconnecting TSC on your CoCo's 6809E from ground, and reconnecting it to +5V, we effectively remove the 6809E from the computer's electronic circuitry. This allows us to stack a 2nd processor (or processor socket) on top of the old one, taking care to connect the 2nd processor's TSC pin to ground.

The Color Computer then ignores the old processor, and takes commands only from the new one.

Preparation

You'll need a 40 pin IC socket, a pair of needlenose pliers, a pair of small diagonal cutters, a low-power fine-tip soldering iron, and about 3 inches of wire-wrap wire. And, of course, the 6309 chip.

Begin by bending pins 5, 6, 33, 36, 38, and 39 of the socket inward, underneath the body of the socket. You can actually clip off pins 5, 6, 33, 36 and 38 if you like, but be sure you just bend pin 39.

Now solder a 1" length of wire-wrap wire between pin 39 and pin 1 of the socket. Run the wire across the bottom of the socket, and don't bend pin 1 or get a lot of extra solder on it. Set the socket aside in a safe place when you're done.

Unplug the Color Computer, and open it up (voiding the warranty, of course). Locate IC1, which should be marked MC68B09EP. This is the old 6809E processor. Cut pin 39 of the processor, using the diagonal cutters. DO NOT JUST CUT THE CIRCUIT BOARD TRACE GOING TO PIN 39. YOU MUST CUT PIN 39 ITSELF. If you'r not sure where pin 39 is, look at the circuit board. You should see the numbrers 10, 20, 21, 30, and 40 printed in white on the circuit board around the processor. Find the number 40, which marks pin 40 at one corner of the processor. The next pin down is pin 39. Be sure you cut all the way through the metal pin, so that the top half of the pin sticks out of the black plastic body of the processor while the bottom half sticks out of the circuit board. Make sure that the two halves of the pin don't touch each other.

Stack the socket (prepared as described above) on top of processor IC1. The bent-under socket pins must rest on top of the processor, without touching any processor pins. The other socket pins must each make firm contact with the corresponding processor pins. When you're happy with the way the socket sits on top of the old processor, solder together each pair of touching pins. This locks the socket firmly in place, in addition to establishing good electrical connections.

Finally, plug the 63B09E into the socket. Put the Color Computer back together, and it's ready for use.

Technical Notes

The bent-under pins include TSC, BS, BA, LIC, AVMA, and BUSY. You have to bend over the last 5 of these because the TSC line doesn't place them in high-impedance state; if you connected them up to the 6809E, you could cause a battle for control between it and the new 63B09E. It's OK to just bend these pins over because they're all outputs and they aren't connected to anything in the CoCo.

If the system doesn't work right after installing the new processor, make sure you've bent under the correct pins and that all of the pins you didn't bend have good solder joints. If you absolutely can't get the computer to work, you can either reconnect the old processor (pin 39) and try again some other time, or desolder both the socket and the old processor to install a fresh socket directly on the CoCo's circuit baoard.

The tests I've done indicate that this "piggyback" technique works reliably when done correctly, but it's certainly better from an engineering perspective to remove the old processor and install the socket in its place. The technique described in this article is about 1/10 the work, and unless you're an expert at repairing circuit boards the chances of hurting the computer with this technique are much lower than if you actually remove the old processor.

6551 Hardware Hack

By Terry Trapp

How many times have you said to yourself. "I wish my RS232 Pak would work correctly with my BBS." Dont you just hate it when you can't see modem responses? :(It's all because of the 6551 chip in your RS232 pak or Disto 4-in-1 board. It will not allow you to receive any information from the modem when there is no "Carrier detect", or "CD" for short. Bruce Isted saw this problem and decided to write a driver called "SACIA". It is a fully buffered driver that does one special thing it will allow you to swap "DSR" <Data Set Ready> and "CD" on your cable. DSR is always on when you have a modem hooked up. What you would do with this driver is swap pin 6 <DSR> and pin 8 <CD> (An inverted cable) & SACIA will look at the DSR location for carrier. CD will always be on, so you can see modem responses. That's fine and dandy with the 232 pak, but with the 4 in 1 board from Disto -- Now that's a little different.

The 4 in 1

I got my 4 in 1 three days ago. I took my boot disk over to Trix's house (John Farrar) and begged him to install my device drivers for the SC II & the 4 in 1 board. (He's the software dude, I'm the hardware dude--- together we make a good CoCo programmer. :)) He installed my new drivers but left SACIA in place, he just changed the loaction of where to find the 6551 chip. It should work right?? Eaaaa! Wrong answer thanks for playing! Tony did something different to his 6551 than the one in the Deluxe RS-232 Pak. I fired up RiBBS and BOOM! Carrier Detected! DARN! Ok, I'll disconnect my modem, it will lose carrier then! Eaaaa, Sorry wrong answer thanks for playing! It still had carrier. Why is it doing this??? Look on page 5 of your Disto 4-in-1 board manual. In the middle of the page you see-- *The CTS (Clear To Send) and DSR (Data Set Ready) input signals to the ACIA are always enabled. This means the ACIA device will always transmit, reguardless of what is connected to it.*

What Tony doesn't tell you is that the 6551 <The ACIA device> will ALWAYS transmit regardless of DSR. I sat down and thought to myself-- "I bet he just "Hard Wired" DSR on!" <Hard Wired means physicaly wired> As I came to find out after about 2 hours of trial and error voltage readings, Pin 16 is at 0 volts when CD is on, and at 5v when it is off. The same is true with Pin 17. When DSR is on, 0v is on Pin 17. When DSR is off, 5v is present on Pin 17. <I found this out with my Deluxe RS-232 Pak>.

pin	on	off	use
16	0v	5v	CD
17	0v	5v	DSR

DSR, Pin 17 on the 4 in 1 is soldered to ground. That means there is no voltage on that lead. Therefore DSR is ALWAYS on. That is why I ALWAYS detected carrier with the inverted cable and SACIA. The 4-n-1 CAN'T read DSR from the modem, the hardware isn't connected from the modem to the 4-n-1's 6551. The Deluxe RS-232 DOES have the hardware, therefore an inverted cable can be used with it.

BE CAREFUL MODIFING YOUR 4 n 1, SERIOUS DAMAGE CAN OCCUR IF YOU ARE NOT CAREFUL. IF YOU HAVE ANY PROBLEMS UNDERSTANDING THE MOD.----

=*DO-NOT-ATTEMPT-IT*=

Get help from someone if you can. Remember, electronics works on blue smoke, if you accidently let the smoke out----- they wont work!

The Modification

Here is how to mod. the 4-in-1 for SACIA or any other "Inverted Cable" driver.

1) Disconnect all power and connections from the Disto SC II.
2) Unscrew the 4 screws that hold the cover on.
3) Remove the cover carefully.

4) Get a static wrist strap and put it on.
5) Place the wrist strap's ground to the SC II's CoCo edge card ground located at either end of the CoCo connector.
6) Carefully disconnect the 4-in-1 board.
7) Locate the 6551ap chip.
8) Locate Pins 16 & 17 on the 6551ap.
9) Flip the board to the BACK side.
10) Locate the trace going from Pin 16 to a Node on the circuit board. <A Node looks like a little round solder blotch, the trace continues on the top side of the 4 in 1 board>
11) Cut the trace as close to pin 16 as possible with an "Exacto knife". Be SURE the trace is CUT. Take a continuity reading <if you can> to be sure.
12) Flip the board back over to the TOP side and cut Pin 7 off of the circuit board. CUT IT CLOSEST TO THE BOARD **NOT** THE CHIP.
13) On Top of the board, solder a piece of solid wire through the node that used to goto Pin 16.
14) Solder the other end of the wire to pin 17 of the 6551. NOT the pin on the board, the chip itself.
15) Flip the board over to the BACK side.
16) Solder another wire to Pin 16 of the 6551.
17) Solder the other end of the wire to ground. <I used the larger trace located closest to the pin, GOING UNDER the 6551 chip.>
18) Re-Install the modified 4 in 1. <Be careful to get all the connectors back in their correct positions>
19) Disconnect the static strap.
20) Place the cover back on the 4 in 1, be sure none of the mod. you made will short to ground.
21) Connect everything back together.
22) Pray.
23) Turn on the CoCo, If you get an "OK" with a "Disto Cursor" boot OS9 and see if it works!! If your CoCo crashes when you cut it on, remove the cover to the SC II and check all your connections.

The Conclusion

And there you have it! Simple Huh?? CD is always on, and DSR is connected to the CD from the modem. SACIA will change it back to CD for you!! No need to even set RiBBS up for an inverted cable. No need for an inverted cable, it is all done my the Mod. and SACIA!!!
I hope you have no trouble with this mod. If you need any help, you can reach me at (615)-781-8679 DATA 2400 8n1, or through Fidonet at 1:116/41. Also you can send mail to me via TRIX on Delphi.

"CoCo Max" This document is free.
SACIA is Copyrighted By Bruce Isted and included on the Upgrade Disk packaged with this booklet
The DISTO SuperController II and 4-n-1 board are Copyrighted by DISTO.

== Terry Trap ==

"Hack" a 2nd port to your Deluxe RS-232 PAK

by Bob Brose;Delphi

In reference to several users asking about a second serial port for CoCo OS-9, Bob Brose posted this article on CompuServe (CIS) a while ago.

Parts List:
6551 integrated circuit chip
1489 integrated circuit chip
1.8432 MHz Crystal
Optional: 1488 chip (See Instructions)

Hardware Instructions:

1. Remove the existing 6551 from it's socket
2. On the new 6551, bend pins 2, 5, 6, 7, 8, 9, 10, 11, 12, 16 and 17 up so they point directly away from the body of the chip.
3. Place the new 6551 over the old 6551 (line up the pin 1's) and solder the following top pins to the bottom pins 1, 3, 4, 13, 14, 15, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28 without getting any solder on the lower parts of the legs of the lower 6551.
4. Plug the 6551 back into it's socket. (if your 6551 wasn't socketed you can still do the mod but it will be harder to solder the 2 chips together on the board, be careful not to short any connections with solder blobs).
5. Connect a short jumper from the 74LS04 pin 9 to the top 6551 pin 2. This provides the select signal fro the new 6551 (FF6C-FF6F).
6. **3 Wire Port Option:** Connect the following pins on the top 6551 together: 1, 9, 16 and 17. This sets the CTS, DCD and DSR lines to low (true for them) which is correct for a three wire line with no hardware handshaking. If you want a full port, DO NOT connect these pins together

6a. **Full Port Option:** Connect pins 9, 16 and 17 to gates on the soon to be piggybacked 1489.

7. To pins 6 and 7 on the top 6551, solder a 1.8432 MHz Crystal. This is necessary as the two 6551's cannot share the same crystal because of the way they generate a signal from it. Alternately, you can make a crystal generator out of spare gates on the 74LS04 with one crystal and feed the signal into both pin 6's on the 6551's (leaving pin 7's unconnected) but this requires much more work and since crystals are only about $1, it really isn't worth it.
8. Piggyback the new 1489 on top of the current one making sure that the pins 1 line up, bending up all pins on the new 1489 except 7 and 14. Solder the 2 pin 7's together and the 2 pin 14's together making sure not to short out any other pins or traces.
9. Connect a wire from pin 3 of the piggybacked 1489 to pin 12 of the top 6551 (this is receive data).
10. The existing 1488 has 1 free gate which we will use for transmit data. Solder a wire from the top 6551 pin 10 to the 1488 pin 2.
11. Get your desired RS-232 connector (I used a 25 pin female connector like the original). Solder a wire from a convenient ground (I used the pad by the right rear mounting screw) to pin 7 on the 25 pin RS-232 connector.
12. Solder a wire from pin 3 of the 1488 to pin 2 of the RS-232 connector (this is transmit data).
13. Solder a wire from pin 1 of the piggybacked 1489 to pin 3 on the RS-232 connector (this is receive data).

NOTE: *Pins 2 and 3 can be reversed depending on whether you are talking to a modem or terminal.*

Testing:

Test out the RS-232 PAK by plugging it into a multipak (protects you from major soldering errors) and powering up your machine. If your computer doesn't act completely normal, turn it off immediately and recheck all of the connections against the above instructions. If everything is OK, try out a terminal program for the existing RS-232 PAK. If it works, proceed to the software mod section below, otherwise go back and check your work again.

End of Hardware Mods:

If you want to hook up other input status lines, the piggybacked 1489 can be used to hook up the 3 input status lines, CTS, DCD and DSR. If you are going to use this port with a CALL IN modem, you will need to do this.

NOTE: See the 1489 data sheet for pinouts of the unused gates.

Also, if you need to hook up outgoing status lines like DTR and RTS, you will need to piggyback another 1488 on top of the existing one and connect it up. I'm using my second port for a terminal so none of the handshaking lines were necessary.

Software Modifications:

I use the port only with OS-9 so the changes are minor. You can use the port with RSDOS, but you will need to write your own software to do so. Remember in RSDOS, if you use both ports as interrupt driven ports, your interrupt routine will have to check both ports to see which one caused the interrupt as they are connected together (PC owners WISH they could actively share interrupts!).

For OS-9, I use T3 for my new descriptor. I did this by taking an existing T2 descriptor and changing the least significant byte of the port address at offset 10 (hexadecimal) in the descriptor to 6C from 68. Also you need to change the name of the descriptor, I did this by changing the high bit set "2", which is B2 at offset 38 (hex) to B3 (which is a high bit set "3"). Don't forget to verify the CRC and save out the new descriptor. Create a new boot with the T3 and you are ready to go. The name offset at 38 (hex) above will vary from one descriptor to the next because there are so many versions of the ACIA driver around. I use Bruce Isted's SACIA with great success and recommend it highly.

I routinely call in on T2 and then connect to another computer via T3 and it works completely perfectly.

Hardware "*hack*" to fix IRQ Problem

This fix replaces previous "hacks" to use the hardware interupt provided by the Tandy RS232 Pak, instead of getting it after it circles the entire computer.

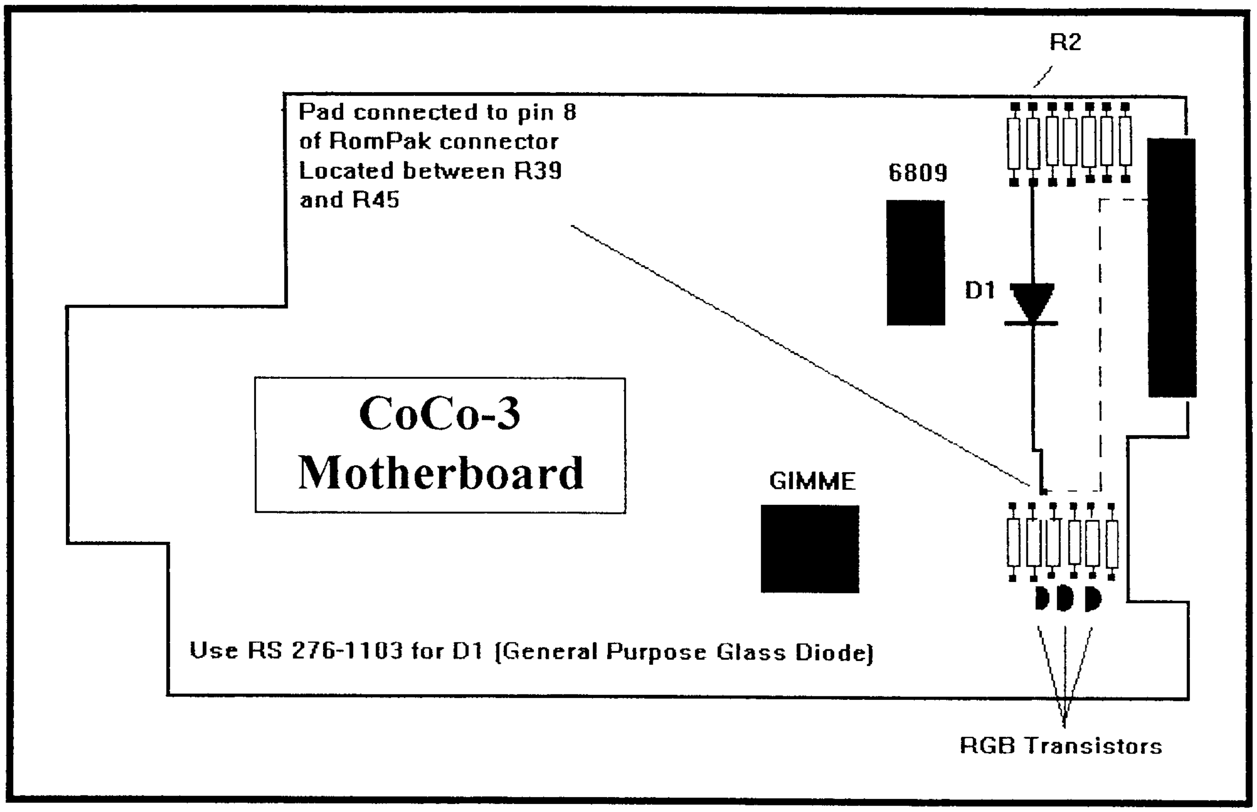
The original fix has you tie all pin 8's on the Multipak together, then, on the RS232 Pak, cut the jumper wire, next to the rompak connector, which connects the 6551 IRQ line to pin 8 of the RS232 Pack Rompak connector. Then run a wire from the 6551 side of the open jumper to pin 3 (IRQ line) of the 6809, by catching the pad where the front of R2 is soldered to the top of the board. That fix seems to work fine, but creates the necessity for a wire, running from the Multipak to the CoCo.

This fix uses a general purpose glass diode 1N914, or Radio Shack RS 26-1103, to couple the interupt signal from pin 8, of the rompak connector, to pin 3 of the 6809. It is all done inside the CoCo, without the need for any external wires. The only catch is that if you have made the earlier fix, you must reconnect the jumper, on the RS232 pack, back to the original position, so that the interupt sent by the 6551 gets to pin 8 of the edge connector.

Pin 8, on the edge connector, on the CoCo III, is hard to get to, as the foil runs on the bottom of the board, but there is a pad on the top of the board, located between resisters R39 and R45.

Refer to the diagram below for the location of the pad, and the direction of the diode. The cathode of the diode must go to pin 8, as the IRQ is a negative going pulse.

Note: It is still necessary to tie all the pin 8's of the multipak together, because of the design of the Multipak, switching between slots (which is done a lot during use of the ACIA, Disk drive, and Hard drive) causes interupts to be lost.



== Eddy Cardone ==

Editor's Note: *There has just been released into the Public Domain a file called* **CLOCK60HZ.AR** *Includes replacement clock drivers for the Tandy's 60Hz Clock driver and Disto and B&B clock drivers that eliminates the need for any hacking!*

OS-9 Parallel Port for the CoCo

Parallel Port for the CoCo?

Why not. It's easy and cheap. Software was the only problem for me and now that's solved. So get out your soldering gun and rip the case cover off you CoCo. This one is going to be easy. <grin>

HARDWARE REQUIRED

40 pin IC socket (RS Cat.# 276-1996)
68B21 PIA chip (found at most electronic supplies)
7404 hex inverter (RS Cat.# 276-1802)
DB25 for IBM type cable (RS Cat.# 276-1565)
or 36 pin Centronics type (RS Cat.# 276-1533)
25 or 36 conductor flat ribbon cable (RS Cat.# 278-772)
thin hookup wire (30 gauge)
double sided tape (RS Cat.# 64-2343).

THEORY: A parallel printer interface functions by placing a byte of data into a parallel latch, then bringing the DATA* handshake line low for a brief period of time so that the printer knows that data is available on its input bus. When the printer has received the data it asserts ACKNG* low to tell the computer that it has received the byte of data. It also unasserts BUSY (= low) at this time. Since BUSY can indicate many other off-line or similar conditions (such as a line-feed in progress), It is necessary to monitor the state of the BUSY handshake line with this interface. The FAULT* line asserted low by the printer indicates a printer malfunction. If PE* is high in addition, the printer is out of paper. PRIME* asserted low by the interface empties the buffer and resets printers that use it.

SOFTWARE THEORY: This driver functions by simply interrogating the parallel interface to see if the printer is ready to receive data. If it is, it sends the byte that was passed to it. If it is BUSY, it loops back and tries again. If a FAULT* signal occurs from the printer, the driver tests for two conditions, no paper (PE) or general error. (The addition of a new error code for Paper Out, E$Paper, (#199) allows the Printerr function to display this. If you use *Printerr* or *OS9p3*, be sure to add an appropriate entry to your Errmsg file in the SYS directory.) If either of the conditions are detected, an error is returned. The only Getstat call supported is "Device Ready". During initialization a PRIME* pulse from the interface to the printer is provided for printers that utilize it.

ADDITIONAL: The device descriptor (P1) included in this article contains no new surprises. The base address of the interface PIA is in the device descriptor -- no hard-coded stuff in this one -- so that you can change it as needed to match your hardware. There are a few unused lines on the port that you could use for other applications as needed.

SUGGESTED REFERENCES: *Investigating the PIA* by Tony DiStefano (July'86 *RAINBOW*), Stearman DOS. *Cooking with CoCo* by Colin Stearman (Dec'84 *RAINBOW*), *Color Computer-3 Technical Reference Manual.*

CONSTRUCTION: Study the schematic diagram (figure 1). Note the asterisk (or stars) located next to pins 1, 20, 21, and 23 - 40 on the 68B21 PIA chip. The stars indicate the pins on the 40 pin IC socket that are soldered to the original 68B21 PIA chip in the CoCo, labeled IC4 (at least that's what it says on my CoCo-3). All of the pins that are not marked with a star should be bent up so that when the socket is placed over the PIA (IC4), the socket pins will not come in contact to the corresponding pins on IC4.

DB25 or Centronics Pin #		Function	IC Pin #
1	1	Data Strobe	19
2	2	Data Bit 1	10
3	3	Data Bit 2	11
4	4	Data Bit 3	12
5	5	Data Bit 4	13
6	6	Data Bit 5	14
7	7	Data Bit 6	15
8	8	Data Bit 7	16
9	9	Data Bit 8	17
11	11	Busy	2
12	12	Page End	3
18	16	Signal Ground	1
16	31	Prime	8
15	32	Error	9

On the original PIA (IC4) cut pin 22. Bend the pin up and solder a 4 inch length of small hookup wire to pin 22. This is one of the Chip Select pins (CS3). When this pin has a high logic level voltage applied to it, the PIA is enabled.

Press the 40 pin IC socket over the top of IC4 and solder socket pins 1, 20, 21 and 23 - 40 to the corresponding pins on IC4. Solder a 4 inch length of thin hookup wire to socket pin 22.

You can connect your parallel I/O lines on the 40 pin socket directly to a Centronics type male plug with 36 conductor flat ribbon cable or connect your parallel I/O lines to a female DB25 type plug using 25 conductor flat ribbon cable. In either case, solder the wires to the 40 pin IC socket as indicated in table 1.

Now plug your new 68B21 PIA chip into the socket being careful that all of the pins insert properly.

Cut a piece of double sided tape to the appropriate size in order to mount the hex inverter chip (7404) upside down on the CoCo mother board between R16 and R7, approximately 1 inch to the left of the PIA (IC4). This is an unused section of the motherboard just the right size for the 7404 chip. With short lengths of thin hookup wire, connect pins 1, 3, 5, 7, 9 & 11 of the 7404 chip to pin 1 of either of the two PIA chips or any other convenient ground potential on the CoCo mother board. Connect pin 14 of the 7404 chip to pin 20 of either of the two PIA chips. Solder the free end of the wire connected to pin 22 of the original PIA chip (IC4) to pin 12 of the 7404 chip. Solder the free end of the wire connected to pin 22 on the 40 pin socket to pin 13 of the 7404 chip. Solder an 8 inch length of wire to pin 13 of the 7404 chip and pin 7 on IC2 (ROM Chip) located directly below the 6809 CPU chip and to the right of the GIMME chip.

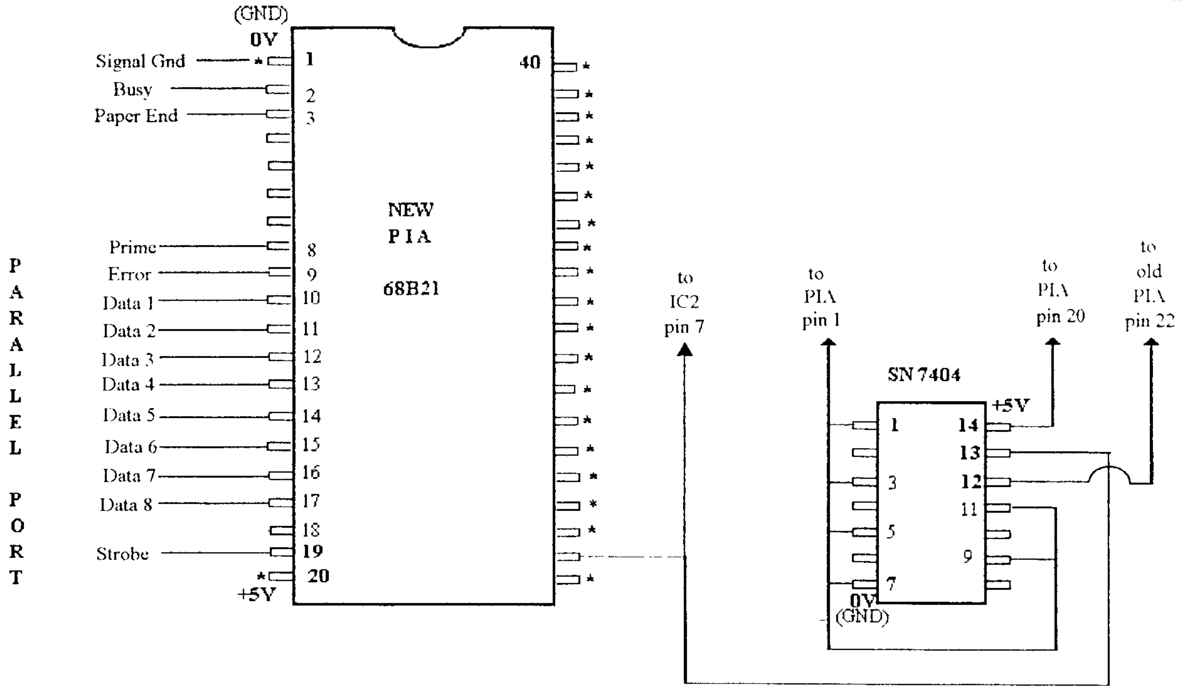


Figure 1

CIRCUIT THEORY: The original PIA chip in the CoCo is addressed at FF20 (hexadecimal). As long as the Chip Select lines are set correctly (CS0, CS1, CS2) the PIA is active. CS0 and CS1 are high active inputs. CS0 (pin 22) is hard wired to a high logic level. CS1 is controlled by the logic voltage from IC9 (74LS138). CS2 is controlled by address line 5, which is placed at a low state within the PIA address range. When CS0 and CS1 are set high and CS2 is set low, the PIA chip is active. CS1 and CS2 are already being controlled by address line 5 and the GIMME chip via the 74LS138 (IC9). Only CS0 is available to "mess with".

The original printer descriptor "P" in the OS9Boot file has a port address set at FF22. This provides the right logic levels on the address lines to enable the 68B21 PIA chip which is used for the bit banger port we use for the serial printer. What we want to do is provide a toggled logic level that we can apply to pins 22 of the 68B21 PIA chip instead of the hard wired high logic level. We accomplish this by picking up Address Line 3 from pin 7 on the Extended Basic ROM Chip (IC2) and applying it to pin 22 of the new PIA and the input of an inverter. The output of the inverter is applied to pin 22 of the original 6821 PIA instead of the hard wired logic level. (*That is why we had to disconnect pin 22 from the circuit board*).

If we use FF28 as our I/O address instead of FF20, Address Line 3 will be set high. That logic high level will be applied to the new 6821 PIA chip which already has the proper logic levels on CS1 and CS2 so it will be enabled. Address Line 3 also being present at the input of the inverter results in a low logic level at the output of the inverter which is tied to pin 22 of the original 6821 PIA causing it to be disabled.

SOFTWARE: The prallel driver *LPRINT*, and the descriptor *P1* are included in the Upgrade Disk included with this booklet. Install both of them in your **OS9Boot** file using *OS9Gen* or *EZGen* or *QWIKGen*.

FINAL COMMENTS: I originally had this type of set up in my old grey CoCo-l F-board running Stearman DOS burned into an EPROM. I knew that it must be possible to do the same type of piggyback hardware modification to the CoCo-3 using OS-9. Writing the driver and descriptor was what prevented me from accomplishing this objective. I finally found a file on Delphi called ***LPRINT.AR*** by Duane Penzien in the OS-9 Sig. Now the software problem is solved.

Duane originally intended to mount the PIA on a circuit board plugged into the Muli-Pak. He had concerns about loading down the un-buffered bus in the CoCo. I have not experienced any problems although I am concerned about the free air space between the two PIA chips. Excessive heat will reduce the life expectancy of the chips.

== Rodger Alexander. Mike Pleas and Duane Penzien ==

Replace CoCo-3 Crystal for more speed

I created this file in order to explain how I (not necessarily you) got my CoCo 3 running with a 38 MHz crystal. effectively boosting the clock speed of my machine from 1.78 MHz to 2.375 MHz (a noticeable improvement!)

Following is the information in "instructions" format. PLEASE don't do this if you're not experienced at least a little bit with a soldering iron and electronic circuitry. for a) it requires opening both your CoCo-3 and your monitor. and b) there's no guarantee it will work so you may have to reverse the process.

BENEFITS OF A FASTER CRYSTAL

The faster crystal in a CoCo-3 will noticeably improve speed. not only under OS9, but under RS-DOS as well. The speed increase will be relative in everything. not just math. text scrolling. or disk access.

While at first the increase from 1.78 to 2.375 (with a 38 MHz crystal) sounds small in number. when you calculate it. it comes to a 33.5% speed increase. twice the increase boasted by PowerBoost or NitrOS9. When coupled WITH NitrOS9 or PowerBoost (as I have done). your CoCo will be approximately 50% faster than a stock Color Computer 3.

DRAWBACKS OF FASTER CRYSTAL

You are using the computer and monitor at speeds for which they were not originally designed. While I've been using the 38 MHz crystal for some time now. things run a little hotter. and the GIME obviously struggles sometimes (i.e. it momentarily stops generating sync every once in a while. and runs very warm).

Software's timing will be off. Software that uses the fixed rate of the original crystal for internal workings will either not work at all. or will function differently. While there isn't much of this software. it does exist. (OS9 is one of these. but there's a fix which will be described later).

You will have to replace your 150 or 120 nanosecond DRAMs with 100 nanosecond or better. Users of the Tandy 512K upgrade are shut out from using crystals over 32 MHz, because the board is mounted face down and generates so much heat at higher speeds that the machine crashes very quickly.

You may not be able to get a display. Monitors are designed to sync at specific rates. When you alter your crystal value. your monitor will be expecting a different rate. and thus. will display data incorrectly. Thus. you will in most cases be required to open up the actual monitor and adjust internal workings.

WHAT YOU WILL NEED

- A crystal of known value
- Soldering Iron, Solder Sucker, and Solder
- OS9 Users: The Eddie Kuns CLOCK module (edition #9). This is needed because it allows you to easily change the value of the clock rate. (also, it's just a much better clock module than the OS9 clock.)
- The guts to open up your CoCo and monitor and modify them!

STEP 1: THE CRYSTAL

Crystals come in almost an infinite number of values, from less than 1 MHz to almost a hundred. The idea is to choose the highest value of crystal that your system will function with. Here are the experiments from my system. It should be noted that I have a 2 MHz rated 6309 (63B09) installed in my system, I have not attempted this with a 2 MHz 6809 (68B09).

Also note that while this is the way my system responded, yours may be entirely different, as CoCo hardware is noted for inconsistency.

Value	CPU	MHz	MultiSync	CM-8	VM-5
28.636*	Normal	1.8	Display	Display	Display
30.000	Normal	1.9	Display	Adjust	Adjust
31.117	Normal	1.9	Display	Adjust	Adjust
33.880	Normal	2.1	Display	Adjust	Adjust
38.000	Normal	2.4	Adjust	Adjust	No Sync
40.000	No function	2.5 --	--	--	

NOTES:

(28.636) is the speed of the standard CoCo-3 crystal, and was included for reference.
VM5 *refers to the Tandy VM5 monochrome monitor.*
MultiSync *refers to a Sony MultiSync monitor*
CM8 *here refers to the Tandy CM8 Monitor.*

STEP 2: INSTALLATION

Installing the crystal is very simple.

1. Open your CoCo-3 case and locate the crystal. It is a silver cylinder under the keyboard at the edge of the motherboard and is marked "**KDS 28.63636**" (or a similar value for PAL machines).
2. With your soldering iron and solder sucker, desolder the crystal and remove it.
3. Insert the new crystal and solder it in.

STEP 3: MOMENT OF TRUTH

Hook your CoCo and monitor up. Turn on the monitor first, and give it a few moments to warm up. This is so that you can see what happens immediately after you turn your machine on and can turn it off again quickly if it's bad! <grin>
After you've given the monitor 10 or 15 seconds, turn the CoCo on. If all appears to be working fine, then your hardware will require no further adjustment, and you can go pour yourself a cup of Coffee and stare at your faster CoCo!

OTHERWISE, check the following:

- If the screen flashes garbage and then goes blank, chances are your CPU is not generating a video signal, and your crystal is either too fast or too slow for the CPU to function right.
- If the screen is just blank and flashes nothing, then your crystal is probably bad, and you should try another one.
- If the monitor is not displaying. Try the following: (Assuming you've gone into RS-DOS.)

Type CLS1 and press ENTER. The "garbage" should change it's general color. Try it again with CLS2 or CLS3. If the garbage is changing colors then you have a working CPU and a monitor not displaying properly.

If CLSX does NOT work, then shut your CoCo off. this crystal is either too fast or too slow for your CPU.

STEP 4: MONITOR ADJUSTMENTS

There are a number of monitors popular with CoCo users. I will only list two here specifically, along with some general instructions.

CM-8 --
Open up the CM-8 monitor case. This entails removing six screws: Two at the top rear, two at the bottom rear, and two in the rear panel which the cables come out of.

Once open, you will see lots of stuff, most of which should be ignored. What you are looking for is a small "pot" that is (looking from the top front) located on the left edge of the circuit board, and will be exposed without requiring you to pull the circuit board all the way out. There should be three pots located here, and the one that you will want to play with is labeled **"H. Hold"**. With the CoCo on, take a small screwdriver and adjust this pot in either direction until the "garbage" turns into a spinning display (the kind that requires V-Hold adjustment).

After getting the spinning display, adjust your **V-Hold knob** on the front panel until the display locks into place. At this point, you may want to play around with the pot next to it called **V.Size** until you get the vertical size up to a normal level (easiest done in Width 80). Once this is done, you can close up your monitor, and you're done for the hardware!

SONY (MultiSync)--
The Sony monitor had an interesting display problem. At higher crystal values, the display would be readable, but hourglass-shaped when the CoCo was turned on. This was rectified as follows:

Open up the Sony monitor. The back cover is held on by four screws, two on top and two on the bottom. Once the screws are loosened, take the cover completely off. With the CoCo on, locate a hole on the image board (the most complex and largest one), marked **"H.Sync"**. It is near the front, and about halfway up. Insert a small, straight-edged screwdriver into the hole and rotate until the hourglass suddenly pops out to a normal display. V.Sync, as you will see, automatically adjusts (it's a MultiSync monitor... <grin>). Close the monitor, and you're done with the hardware.

OTHER MONITORS --
Generally, there are two values you will need to adjust. On both MultiSync and other monitors, you will need to adjust the Horizontal sync rate, commonly labeled as **H.Sync"**, **"H.Hold"**, or **"Horiz.** S.". On NON-MultiSync monitors ONLY, you will also need to adjust **"V.Sync"**, **"V.Hold"**, or **"Vert.** S.", otherwise known as the Vertical syc rate.

STEP 5: SOFTWARE INSTALLATION

At this point, exclusive RS-DOS users can stop. OS9 users, however, will experience a little bit of trouble. Because OS9 derives its CLOCK timing from the crystal oscillator's fixed rate, the system clock, with a faster crystal, will run fast (the 38 MHz crystal causes about a 2 minute gain every five real minutes!). The remedy for this is the **CLOCK module by Eddie Kuns, edition #9**. This is a full-fledged GIME toggle clock, faster than the stock OS9 clock, and with versions for Disto and Burke and Burke RTC's, so have no fear.

First thing you will need to do is go through the standard procedure for creating and OS9 bootfile (refer to the OS9 manual, and to the millions of documents telling you how to do so), replacing the standard *Clock.60Hz* with your appropriate version of the *Clock #9* module (depending on whether you're using a B&B, Disto, or software clock).

Boot with this new clock, and then, using modpatch (or for us Burke and Burke people, EZGen), you will need to change the clock module as follows:

1. Divide the value of the original crystal (28.63636) by 60.
2. Then, divide the value of your new crystal with this value.
3. Round it off to the nearest decimal number, and, using a calculator or computerese friend, convert it to hex.
4. Change the value at location $7B from $3C to the new value.

5. Do the cobbler thing, or for EZGeners or KWIKGeners, save the new Clock to your bootfile (or write the bootfile).

YOU ARE DONE! Enjoy.
==Aron Hsiao==

Add a CLOCK to your CoCo-XT

Four years ago, when it was time to add a hard drive to my Color Computer 3, I looked around at the current advertisements in *The Rainbow* Magazine. There was hardware setups available from Disto, Owl Ware, FHL and Burke & Burke. Let me interject at this time that I claim to be of Scotch heritage (whether its true or not, I'm not sure). In other words I'm cheap! Or at least "penny wise and pound foolish". I chose to go with the Burke & Burke Interface because I already had an MFM Drive and the price was right: $69 dollars for the ***CoCo XT*** package and $99 for the ***CoCo XT-RTC*** package.

OK....Do I buy the $69 *XT* or pay an additional $30 just to get the **R**eal **T**ime **C**lock. Hey! I'm not that lazy. Typing in the current date each time I boot OS-9 is no big deal.

YEAH, SURE! "No big deal". Brave words for 10 lazy fingers. You can figure out the rest. Now I want the clock.

I borrowed an *RTC* board from the Seattle 68xxxMUG and compared the differences between the clock version (RTC) and my non clock version. The circuit boards were identical. What was missing on the non clock version was a 3 volt battery, battery holder, clock chip, crystal and capacitor. The battery holder was unusual and I had no idea of the frequency of the crystal, so I called Terry Laraway.

Terry Laraway to the rescue

If you don't know Terry Laraway, then pay attention. He has every weird part that ever came out for the CoCo and he is constantly researching new sources for parts that we may need to upgrade our little machine to a Super Computer. His efforts and stock are available under the name of *CoCo Etc.* He even has those KEL AM 34 pin "male" IDC type connector for plugging ribbon cable into the CoCo's ROM port. They are a "must have" item when installing a CoCo into a PC Case. I've included Terry's address and phone number at the end of this article.

Anyway, I called Terry to see if he knew where I might be able to pick up the specific parts Burke & Burke used for their *CoCo XT RTC Hard Drive Interface.* Terry not only new the specific part items but just happened to have them in stock. Best news of all was that the price was right...cheap!

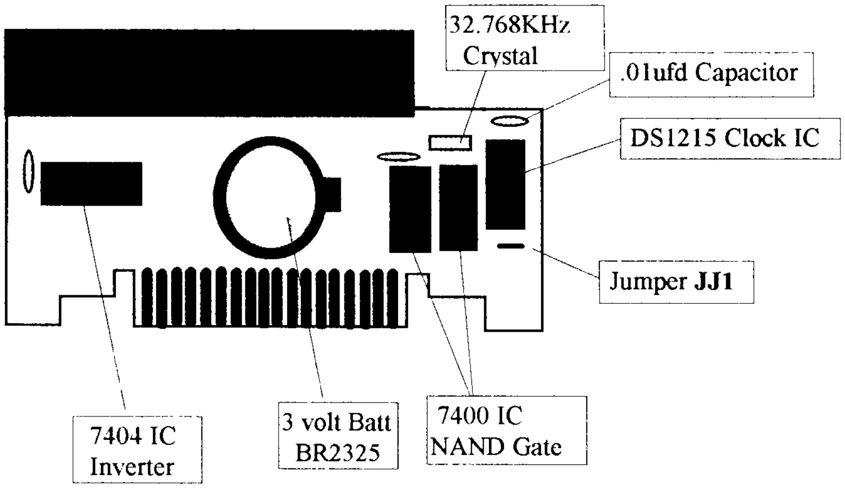
I ordered the parts from Terry and found them in the next day's mail (talk about service)! I soldered in the 16 pin socket in the pre drilled holds on my *CoCo XT* circuit board. Then I soldered in the battery holder and the .01ufd capacitor. I plugged in the clock chip and the 3 volt battery, plugged in my hard drive controller card and plugged the complete unit into my multipak and turned on the power.

It didn't smoke, but then it didn't work either? As OS-9 was booting up, I noticed that the hard drive access light was blinking in a random manner. What the heck?

I pulled out the interface and put on my glasses to do a pin by pin comparison between the two boards. Pin 1 and 2 on the clock chip go to the crystal, pin 3 goes to Address Line 2 (pin 21 on the CoCo's I/O bus), pin 4 goes to the battery, pin 5 goes to ground, pin 6 goes to Address Line 0 (pin 19 on the CoCo's I/O bus), pin 7 goes to Data Line 0 (pin 10 on the CoCo's I/O bus), pin 8 goes to ground, pin 9 jumpers to pin 15, pin 10 and 11 go to a jumper on the board located just below the chip and then to pin 9 on one of the two 7400 nand gate chips located next to the clock chip. Pin 12 goes to pin 7 on the other 7400 chip, pin 13 goes to the 7404 inverter chip (the inverted signal goes to pin 5 on the CoCo's I/O bus), pin 14 goes to ground, pin 15 jumpers to pin 9 and pin 16 goes to 5 volts.

Everything checked out but it still didn't work. On my third examination I noticed that there was a jumper etched on the top side of the board underneath the left 7400 chip. The bottom of the jumper was hidden under the socket, but on the reverse side it appeared that the jumper connected to pin 8 of the 7400 chip, but the feed through hole looked to be damaged and there was no electrical connection between the jumper and pin 8 on the chip. **WARNING!** Apparently the lack of electrical connection was on purpose. It seems that the board had a design mistake and the feed through hole was intentionally drilled out to eliminate the conductor material. Of course I tried to repair the jumper resulting in a totally dead interface.

In the end it turned out that the only modification that needed to be done was to remove the jumper wire located below the clock chip. It is labeled clearly on the circuit board as **JJ1**. That's' it!



PARTS LIST

16 pin IC socket	$ .15
32.768KHz Crystal	$1.00
DS1215 Clock IC	$12.50
Battery Holder	$1.00
BR2325 3v Battery	$1.50
.01ufd Ceramic Cap.	$ .35
Total:	$16.50

Source of parts for this project are available from Terry Laraway's ***Small Grafx Etc.*** 4a N.W. Doncee Drive, Bremerton, WA 98310. Phone (206) 692-5374

Rodger Alexander

Appendix A

General OS-9 Commands

FORMAT - Prepare media for writing (Replaces DSKINI in DECB)
BACKUP - Copies disk to identical media sector by sector (Same as DECB)
COPY - Copies a file from one location (name) to another (Same as DECB)
DEL - Removes a file from the disk (Same as KILL in DECB)
RENAME - Changes a file's name (Same as DECB)
MAKDIR - Creates a subdirectory
DELDIR - Removes a subdirectory

BUILD - Creates a text file from standard input (STDIN, usually the keyboard)
LIST - Sends a file to standard output (STDOUT, usually the screen)
DISPLAY - Sends specified hexadecimal values to STDOUT
ECHO - Sends the following information to STDOUT (best used in a shell script or redirected)
PROCS - Displays information on all currently active processes

FREE - Shows available space on disk in blocks, of 256 bytes (Similar to FREE() in DECB)
MFREE - Shows available RAM memory and fragmentation (Similar to PRINT MEM in DECB)

HELP - Displays brief help information on a particular topic
ERROR - Displays error message when given message number
SETIME - Sets the system time and date
MONTYPE - Tells OS-9 what type of monitor is attached to your system (Level II only)
XMODE - Configures port protocols. Can be used to set printer or terminal baud rate, etc.

Appendix B

6809 OS-9 Level II Error Messages

1 - Unconditional Abort
2 - Keyboard Abort
3 - Keyboard Interrupt
4 - Window Change

BASIC09 Error Messages*

10 -- Unrecognized Symbol
11 -- Excessive Verbage
12 -- Illegal Statement Construction
13 -- I-code Overflow
14 -- Illegal Channel Reference
15 -- Illegal Mode (read/write/update)
16 -- Illegal Number
17 -- Illegal Prefix
18 -- Illegal Operand
19 -- Illegal Operator
20 -- Illegal Record Field Name
21 -- Illegal Dimension
22 -- Illegal Literal
23 -- Illegal Relational
24 -- Illegal Type Suffix
25 -- Too-large Dimension
26 -- Too-large Line Number
27 -- Missing Assignment Statement
28 -- Missing Path Number
29 -- Missing Comma
30 -- Missing Dimension
31 -- Missing DO Statement
32 -- Memory Full
33 -- Missing GOTO
34 -- Missing Left Parenthesis
35 -- Missing Line Reference
36 -- Missing Operand
37 -- Missing Right Parenthesis
38 -- Missing THEN statement
39 -- Missing TO
40 -- Missing Variable Reference
41 -- No Ending Quote
42 -- Too Many Subscripts
43 -- Unknown Procedure
44 -- Multiply-defined Procedure
45 -- Divide by Zero
46 -- Operand Type Mismatch
47 -- String Stack Overflow
48 -- Unimplemented Routine
49 -- Undefined Variable
50 -- Floating Overflow
51 -- Line with Compiler Error
52 -- Value out of Range for Destination
53 -- Subroutine Stack Overflow
54 -- Subroutine Stack Underflow
55 -- Subscript out of Range
56 -- Parameter Error
57 -- System Stack Overflow
58 -- I/O Type Mismatch
59 -- I/O Numeric Input Format Bad
60 -- I/O Conversion: Number out of Range
61 -- Illegal Input Format
62 -- I/O Format Repeat Error
63 -- I/O Format Syntax Error
64 -- Illegal Path Number
65 -- Wrong Number of Subscripts
66 -- Non-record-type Operand
67 -- Illegal Argument
68 -- Illegal Control Structure
69 -- Unmatched Control Structure
70 -- Illegal FOR Variable
71 -- Illegal Expression Type
72 -- Illegal Declarative Statement
73 -- Array Size Overflow
74 -- Undefined Line Number
75 -- Multiply-defined Line Number
76 -- Multiply-defined Variable
77 -- Illegal Input Variable
78 -- Seek Out of Range
79 -- Missing Data Statement

OS-9 Error Messages

183 - Illegal window type
184 - Window already defined
185 - Font Not found
186 - Stack Overflow
187 - Illegal Argument
188 - unused
189 - Illegal Coordinates
190 - Internal Integrity check
191 - Buffer size is too small
192 - Illegal Command
193 - Screen or Window Table is Full
194 - Bad/Undefined buffer number
195 - Illegal window definition
196 - Window undefined
197-199 are unused
200 - Path Table Full
201 - Illegal Path Number
202 - Interrupt Polling Table Full
203 - Illegal Mode
204 - Device Table Full
205 - Illegal Module Header
206 - Module Directory Full
207 - Memory Full
208 - Illegal Service Request
209 - Module Busy
210 - Boundary Error
211 - End of File
212 - Returning non-allocated memory
213 - Non-existing Segment
214 - No Permission
215 - Bad Path Name
216 - Path Name Not Found
217 - Segment List Full
218 - File Already Exists
219 - Illegal Block Address
220 - Phone Hangup-Carrier Detect lost

6809 OS-9 Level II Error Messages

221 - Module Not Found
223 - Suicide Attempt
224 - Illegal Process Number
226 - No Children
227 - Illegal SWI Code
228 - Process Aborted
229 - Process Table Full
230 - Illegal Parameter Area
231 - Known module
232 - Incorrect Module CRC
233 - Signal Error
234 - Non-existent Module
235 - Bad Name
236 - Bad Module Header
237 - RAM Full
238 - Unknown Process ID
239 - No task number available
240 - Unit Error
241 - Sector Error
242 - Write Protect
243 - CRC Error
244 - Read Error
245 - Write Error
246 - Not Ready
247 - Seek Error
248 - Media Full
249 - Wrong Type
250 - Device Busy
251 - Disk ID Change
252 - Record is locked-out
253 - Non-sharable file busy
254 - I/O Deadlock Error

* Messages 5-182 vary according to the application

Appendix C

OS-9 Command Summary

ATTR
Syntax : Attr filename [permissions]
Usage : Examine or change the security permissions of a file
Opts : -perm = turn off specified permission
perm = turn on specified permission
-a = inhibit printing of attrs after change
Perms : d - directory file
s - non-sharable file
r - read permit to owner
w - write permit to owner
e - execute permit to owner
pr - read permit to public
pw - write permit to public
pe - execute permit to public

BACKUP
Syntax : Backup [e][s][-v][dev][dev]
Usage : Copies all data from one device to another
Opts : e = exit if read error occurs
s = single disk backup
-v = do not verify writes

BASIC09
Syntax : Basic09
Usage : Basic language package

BUILD
Syntax : Build filename
Usage : Builds short text files from standard input, blank line ends input

CHD
Syntax : Chd <pathlist>
Usage : Change working directory to specified path

CHX
Syntax : Chx <pathlist>
Usage : Change execution directory to specified path

CMP
Syntax : Cmp filename1 filename2
Usage : File comparison utility

COBBLER
Syntax : Cobbler devname
Usage : Creates OS-9 bootstrap file from current boot on new disk

CONFIG
Syntax : Config
Usage : Create custom boots and system disks

COPY
Syntax : Copy pathname pathname [-s]
Usage : Copies data from one file to a second file
Opts : -s = single drive copy

DATE

Syntax : Date [t]
Usage : Specifies current system time and date
Opts : t = specify time also

DCHECK

Syntax : Dcheck [-opts] <devname>
Usage : Check disk file structure
Opts : -w = path = pathlist to directory for work files
-p = print pathlists for questionable clusters
-m = save allocation map work files
-b = suppress listing of unused clusters
-s = display count of files and directories only
-o = print options for DCheck

DEINIZ

Syntax : Deiniz <devname> {<devname>}
Usage : Detach a device(s)

DEL

Syntax : Del [-x] filename [...]
Usage : Deletes the specified file(s)
Opts : -x = delete relative to execution directory

DELDIR

Syntax : Deldir directory name
Usage : Deletes entire directories

DIR

Syntax : Dir [e][x][dir or path]
Usage : Displays formatted list of the file names in a directory
Opts : e = print extended directory
x = print execution directory

DISPLAY

Syntax : Display <hex>[...]
Usage : Displays converted characters to standard output

DSAVE

Syntax : Dsave [-opts][dev][pathname]
Usage : Generates procedure file to copy all files in a directory system
Opts : -b = make a system disk by using OS9boot if present
-b=<path> = make system disk using path as source
-i = indent for directory levels
-l = do not process below the current level
-m = do not include makdir commands in procedure file
-s<num> = set copy size to num K

ECHO

Syntax : Echo <text>
Usage : Echo entered text to standard output

EDIT

Syntax : Edit {<path>}
Usage : Standard line oriented text editor

ERROR
Syntax : Error errno [...]
Usage : Outputs text error messages for given error numbers

EX
Syntax : Ex <modname>
Usage : Chain to the given module

FORMAT
Syntax : Format <devname>
Usage : Initializes an OS-9 diskette
Opts : R - Ready
L - Logical format only
"disk name"
1/2 - number of sides
'No. of cylinders' (in decimal)
:Interleave value: (in decimal)

FREE
Syntax : Free [devname]
Usage : Displays number of free sectors on a device

GFX
Syntax : RUN GFX(<funct> <args>)
Usage : Graphics interface package for BASIC09 to do compatible VDG graphics commands

GFX2
Syntax : RUN GFX2([path]<funct> <args>)
Usage : Graphics interface package for BASIC09 to handle enhanced graphics/windowing commands.

GRFDRV
Syntax : none
Usage : Graphics Driver module, needs to be loaded to handle graphics/windowing commands

HELP
Syntax : Help [subject][-?]
Usage : Give on-line help to users will prompt if no subjects given
Opts : -? give list of help topics

IDENT
Syntax : Ident <pathlist> [-opts]
Usage : Displays header information from OS-9 memory modules
Opts : -m = look at module in memory
-s = use single line output
-v = do not verify module CRC
-x = pathlist begins at execution directory

INIZ
Syntax : Iniz <devname>{<devname>}
Usage : Attach a device

INKEY
Syntax : RUN INKEY([path],strvar)
Usage : BASIC09 subroutine to input a single key stroke

KILL
Syntax : Kill <procId>
Usage : Send an abort to the process specified

LINK
Syntax : Link <modname>
Usage : Link to a memory module

LIST
Syntax : List filename [...]
Usage : Lists the contents of text files

LOAD
Syntax : Load <pathname> [...]
Usage : Loads modules into memory

MAKDIR
Syntax : Makdir <pathname>
Usage : Creates a new directory file

MDIR
Syntax : Mdir [e]
Usage : Displays the present memory module directory
Opts : e = print extended module directory

MERGE
Syntax : Merge <path> [...]
Usage : Copies multiple input files to standard output

MFREE
Syntax : Mfree
Usage : Displays the amount of free RAM memory

MODPATCH
Syntax : Modpatch <filename> [opts]
Usage : patch a module in memory from command file
Opts : -s = silent mode
-w = suppress warnings
-c = compare module only, do not change
-? = receive help
Cmds : L modname = link to module
C off obyte nbyte = change obyte at off(set) to nbyte
V = verify module
M = mask IRQs
U = unmask IRQs

MONTYPE
Syntax : Montype [opt]
Usage : Set monitor type
Opts : r = rgb monitor
c = composite monitor
m = monochrome monitor

OS9GEN
Syntax : OS9Gen devname [-s]
Usage : Creates and links an OS-9 bootstrap file, reading module names from STDIN
Opts : -s = single drive option

PROCS
Syntax : Procs [e]
Usage : Displays a list of processes running in the system
Opts : e = display all processes in the system

PWD
Syntax : Pwd
Usage : Prints the current data directory path

PXD
Syntax : Pxd
Usage : Prints the current execution directory path

RENAME
Syntax : Rename <filename> <new filename>
Usage : Gives the file or directory a new name

RUNB
Syntax : Runb <i-code module>
Usage : BASIC09 run time package

SETIME
Syntax : Setime [yy/mm/dd/hh:mm:ss]
Usage : Sets and activates the system clock

SETPR
Syntax : Setpr <procId> <num>
Usage : Sets the priority of the specified process to num

SHELL
Syntax : Shell <arglist>
Usage : OS-9 command interpreter

TMODE
Syntax : Tmode [.pathname] [params]
Usage : Displays or changes the operating parameters of the terminal

TUNEPORT
Syntax : Tuneport </t1 or /p> [value]
Usage : Adjust the baud value for the serial port

UNLINK
Syntax : Unlink <modname>
Usage : Unlinks module(s) from memory

WCREATE
Syntax : Wcreate [opt] or /wX [-s=type] xpos ypos xsiz ysiz fcol bcol [bord]
Usage : Initialize and create windows
Opts : -? = display help
-z = read command lines from stdin
-s = type = set screen type for a window on a new screen

XMODE
Syntax : XMode <devname> [params]
Usage : Displays or changes the parameters of an SCF type device (modifies device descriptor)
Param : baud=h where
0 = 110 3 = 1200 6 = 9600

1 = 300	4 = 2400	7 = 19,200 (ACIAPAK only)
2 = 600	5 = 4800	8 = 32,000 (SIO Only)

NOTE: There are many more parameters allowed (see chapter 6 in the manual)

Appendix D

Software Upgrades

The following list are the most standard Upgrades to OS-9 Level Two and are included on the *Upgrades* side of the Floppy Disk that is included with this book:

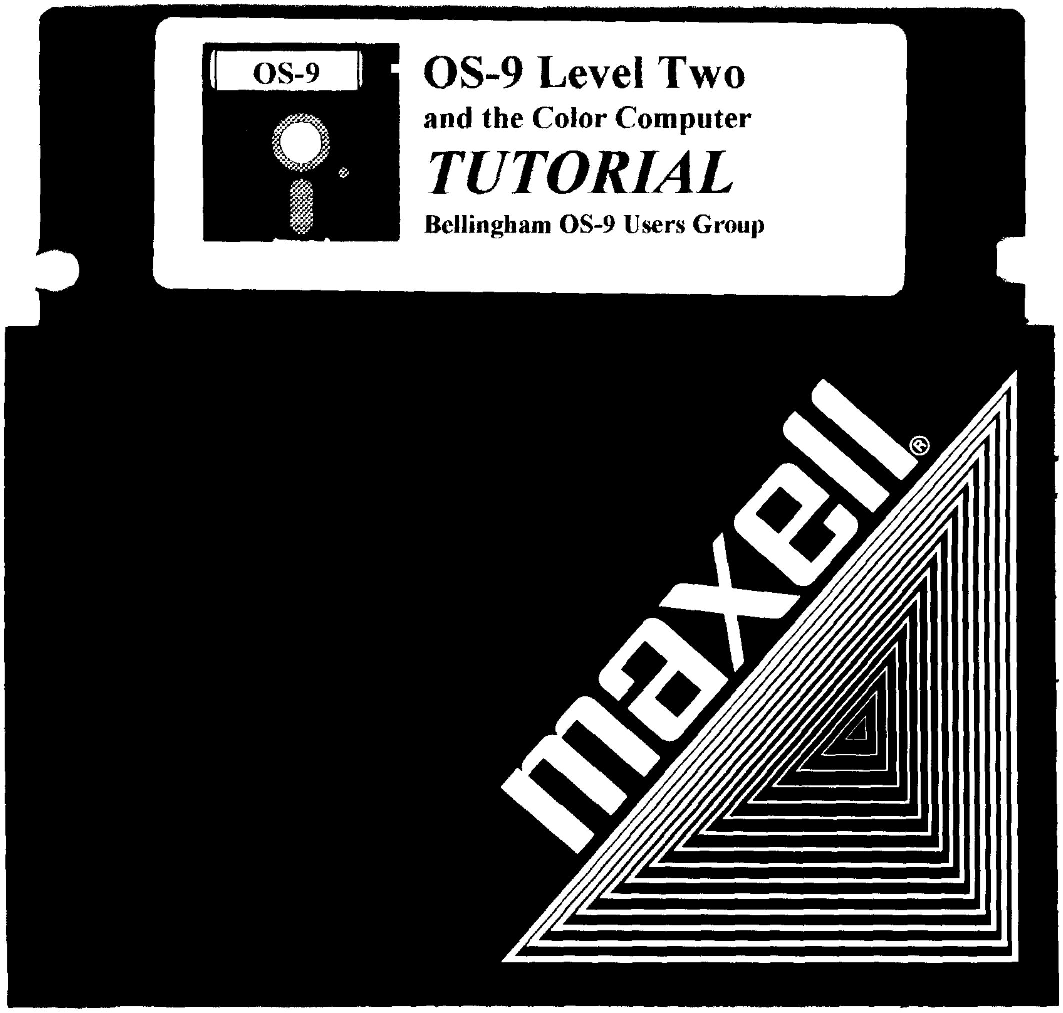
- **GRFDRV** by Kevin Darling. This IPatch file will increase the speed of *GRFDRV*by a factor of 16.
- **SHELL+** (Shell2.1) Replaces the original *shell* file in your CMDS directory. Features include modified prompts; variable assignments and storage; boolean logic programing within script files; wilcards; redirection, etc. See included documentation for complete description of all features.
- **GSHELL+** This is an IPatch file that Upgrades Multivue's *GShell* module providing more options and all known bugs corrected as well as improved performance. Requires 512K of RAM.
- **GFX2+** This is a replacement file for the *GFX* file included with *Basic09*. Features more advance mouse and cursor manipulations, more commands and improved color pallette control.
- **PROC** by Kevin Darling. This is a replacement for the *Procs* file in your CMDS directory. Identifies active processes by NAME as well as by number.
- **CC3DISK** (version 11) This is a replacement for the original *CC3Disk*. Permits 512K file sectors so that other formats, such as "Standard OS-9" and "MS-Dos" can be read under OS-9 Level Two.

Not included on this disk, but a must for OS-9 Level Two is the *Level Two Development System.* This is a collection of files, utilities and applications that was not included on the OS-9 Level Two System Disk, but are in fact a part of the complete OS-9 Level Two Operating System. Available from Radio Shack via their Customer Order Service. *OS-9 Level Two Development System* Catalaog # 26-3032.

Color Computer-3 Hardware Upgrades

The following is a suggested list of the most common modifications to the Color Computer-3 to improve it's overall performance. The below listed upgrades do not include the upgrades featured in this book.

- **PAL Chip Upgrade**. This is required for using the multipak under OS-9 Level Two. The upgrade may be performed by the customer or by Radio Shack's Technical Support. Prices range from $8 to $15.
- **512K Memory Upgrade Kit.** Memory board kits are available from Radio Shack National Parts, or other third party sources, such as Small Grafx Etc., N.W. 41 Doncee Drive, Bremerton, WA 98310 (206)692-5374.
- **IBM Keyboard Interface** by John Puppa. Permits the use of an XT type IBM Keyboard with the Color Computer. Not currently manufactured but third party production may be in the future.
- **No Halt Floppy Disk Controller**. Manufactured by Disto Products, 1710 DePatie, St. Laurent, QC H$L 4A8, Canada. On board processor and RAM oversees disk access operations. This allows the computer's processor to continue it's operationg without the need of "waiting" for the disk drive.
- **Hard Drive Interface** permits the connection of a standard IBM type hard drive to the CoCo. Manufactures include Burke and Burke ($69); PO Box 733, Maple Valley, WA 983038; and Disto Products (See above).



OS-9 Level Two
and the Color Computer
TUTORIAL
Bellingham OS-9 Users Group