



# Virtual Color Computer User Guide (VCC)

---

This guide is written using github markdown and is available online at <https://github.com/VCCE/VCC/wiki>. The online version is updated frequently and is likely to be more up-to-date, but might describe features not yet in the release you are using. The MarkText program ([github.com/marktext/marktext](https://github.com/marktext/marktext)) is used to generate the PDF version of the guide.

VCC is a Tandy Color Computer 3 emulator that runs on the Windows operating system (Windows XP or greater). It allows software written for this 30+ year old computer to run on modern hardware. Please take the time to read this guide to discover some of the features and shortcomings of this emulator. Vcc is a CoCo3 emulator only, it does not emulate a CoCo2 or CoCo1.

You should be aware that VCC by itself only emulates the Coco 3 with no peripherals. Peripherals are added by run-time plug-ins that emulate the various add-in cards that were or are available. See the section on Loadable Modules for an explanation of this.

The Color Computer 3 was the last of a line of micro-computers designed by, and distributed through Radio Shack stores. Released in 1986, it developed a rather large and loyal following that continues to this day. Some of the system's specs are as follows:

- 64k system memory, 128k via bank switching.
- 512K upgrade available from Radio Shack as well as 1, 2. & 8 meg memory upgrades from 3rdparty vendors.
- 32, 40, and 80 column hardware text modes.
- Hi-res graphics resolutions up to 320x225 at 16 colors and 640x225 at 4 colors.
- Software selectable clock speeds of 0.89 and 1.78 MHz.
- Super Extended Color Basic ROM.
- Color Computer 2 compatibility mode.

Unfortunately, in 1991, Tandy decided to discontinue the line.

Lack of support didn't seem to deter the fans. Third-party vendors stepped in to fill the void. New products were and are still being developed. Today, you can buy 512k memory cards, IDE and SCSI hard disk interfaces, and faster, more powerful CPUs in the HD63B09E. There is even a free, user-supported real-time multi-tasking multi-User operating system 'NitrOS9' available for this more than-three-decade old 8-bit machine.

The home of "VCC" is <https://github.com/VCCE/VCC/releases/>. There you will find the latest binary downloads as well as the sources for VCC.

# Copyrights

---

VCC Copyright 2012, Joseph Forgeone

VCC is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License or (at your option) any later version.

VCC is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

The code that comprises this emulator was written by Joseph Forgeone, with the following exceptions:

The RGB to composite color conversion algorithm, floppy disk CRC algorithm, and the 32x16 font set were borrowed from MESS. The MESS source is a valuable source of technical information, without it, this emulator would not be as good as it is today.

The "Becker Port" implementation was patched by Aaron Wolfe and David Ladd, giving "VCC 1.4.3b" the ability to communicate to the host PC via the DriveWire4 protocol, and since then, in VCC 2.0.1, the Becker Port was moved to a cartridge emulation "becker.dll" and must be loaded into the MPI.

The "MS Visual Studio 2015 Community" port of the old "MS Visual C++ 6" sources was done by Gary Coulborne (thanks Gary!) and is allowing VCC to now move forward on the more "modern" versions of Windows.

Various bugs cropped up in the move from VS6 to VS2015, and Wes Gale stepped in to lend a hand at getting the code usable.

The HD6309 code was unfinished and incomplete and was finally put in place by Walter Zambotti.

More recently VCC has been ported to "MS Visual Studio 2022" and a legacy build has been added to continue Windows XP support.

VCC version 2.1.9.2 has been switched from Direct X to Open\_GL for graphics. This has improved support for newer video cards and has alleviated some nagging video issues. Many thanks to

Craig Allsop for doing this conversion.

There have been various additions, enhancements, and bug fixes by Bill Pierce, Gary Coulborne, Wes Gale, Walter Zambotti, James Ross, Peter Westburg, James Rye, EJ Jaquay, Trey Tomes, Mike Rojas, and Craig Allsop.

We welcome anyone interested in the programming and enhancement of the VCCE project. Contact us on the "Color Computer" Facebook group with your ideas, and we will determine if they fit the direction we are headed with the VCC project.

## Credits

---

First and foremost, Joseph Forgeone, the original author of the VCC Color Computer 3 emulator, without all his hard work, this wonderful emulator would have never come about.

Robert Gault for the "RGBDOS" implementation, the emudsk Nitros9 driver, and all his testing and suggestions during the development of VCC. Robert has contributed more to the Coco community than anyone could know.

Aaron Wolfe and David Ladd for the addition of the "Becker Port" for using VCC with DriveWire4. This has allowed VCC to communicate with the outside world and has opened many doors that were closed before.

Gary Colbourn for his work in converting Joseph's Visual Studio 6 C++ code to compile in Visual Studio 2015 Community Edition.

Wes Gale for his bug hunting and squashing during the initial release of VCC to open source.

Walter Zambotti for his contributions to the HD6309 CPU emulation and for inspiring us to add a second hard drive.

David Johnson for an extensive proofreading of the VCC manual.

Mike Rojas for his great work on the VCC Debugger. This one has been a long time coming, and we hope it will get even better in the future.

Bill Pierce for keeping the project alive, for all his great insights, contacts, and knowledge of how the Color Computer is being used by the community. Without Bill VCC would not have advanced much beyond Joseph Forgeone's original work.

Craig Allsop for many recent improvements to VCC and fixes to some long standing issues.

Especially James Ross, James Rye, Peter Westburg (R.I.P.), Ed Jaquay, and Trey Tomes for their endless work on the enhancements and bug fixes of the last couple of releases. Without them, we would not be making this release.

Last but not least, everyone in the Coco community. The help and support of the community is what has kept the interest in the Color Computer going strong and made VCC possible in the first place!

VCC is maintained by the VCCE Open Source Project Team:

Administrators:

- Joe Forgeone (absent)
- Bill Pierce
- Ed Jaquay
- Gary Coulborne (absent)
- Wes Gale (absent)

Programming Contributors:

- Walter Zambotti
- James Ross
- Peter Westburg (R.I.P.)
- James Rye
- Ed Jaquay
- Trey Tomes
- Mike Rojas
- Craig Allsop

VCC Manual, Repository Maintenance, Packaging, and Distribution:

- Bill Pierce
- Ed Jaquay

Inspiration and continued support:

- The Tandy Color Computer Community

# Forward

---

VCC version 1.4.2 was originally written by Joseph Forgeone. In or about 2012, several members of the Coco Community contacted Joseph about VCC since he hadn't updated the software in several years. Joseph informed us that he would like to release the VCC source files to the public as an open source project so that it could be enhanced by others. His desire was to remove the Tandy ROMs (for copyright reasons) and make small changes in his copyrights before releasing the sources. He told us he should have it ready for release in a couple of months. As a token of good faith he offered to send us the sources to "play with" until he made his release.

After that, Joseph seemed to disappear from the map. He had not replied to many emails, nor could any posts be found on any of the forums he was a regular on. He seemed to have disappeared into thin air.

As the events above describe, several of us ended up with sources for VCC and started "playing" with them as Joseph had suggested. The main goal being to add support for the Becker Port system developed by Gary Becker as well as to fix a few minor known bugs. Since Joseph could no longer be contacted, the decision was made to release some changes to VCC only in binary form therefore keeping the sources from public view. Thus VCC 1.4.3b Beta w/Becker Port was born.

The "Becker Port" patch eventually started causing a few *internal* problems in VCC so the Becker Port was to an external cartridge, therefore removing all influences of the Becker Port from VCC proper. So now, to use the Becker Port, you must load becker.dll into either VCC's cartridge slot or a slot in VCC's MPI. We find this new implementation of the Becker Port to be much more stable than the original version.

Several members of the Coco community decided after a couple of years of not hearing from Joseph, that they could follow his intentions and release the VCC sources as open source. In doing so, we decided to make a public announcement in advance to allow anyone with claims to the copyrights of VCC to step up. The first person to reply was Joseph Forgeone!

Not only did we get the "VCC Open Source Project" started, but we did so with Joseph on board and contributing to the project. We have now moved the VCC sources from the old "MS Visual Studio 6" programming environment into the "MS Visual Studio 2015 Community" programming suite, which is a free download from the Microsoft website. The VCC sources can now be compiled for Windows 7, 8, 10, and 11.

We now have an improved VCC which is more compatible with modern versions of Windows and Visual Studio. There are many new enhancements and bug fixes. The future is bright for

VCC!

One last note: VCC is continuously in a beta state and by no means is a finished product. Quite a few things have been changed from the original. There are still bugs to be worked out and enhancements to be added. It is for this reason that the VCC Color Computer 3 emulator has been released as open source. It is hoped that outside contributors and/or programmers will get involved and help this become the best Color Computer 3 emulator available!

# System Requirements

---

VCC runs under most versions of Microsoft Windows. It runs on Linux systems using the Wine Windows emulation package for Linux and on Mac using Winebottle.

The Windows requirements are:

1. Microsoft Windows XP SP2 or greater (Windows Vista, 7, 8, 10, & 11 are supported).
2. Storage large enough for the Windows OS and the VCC installation, as well as space for any Coco ".dsk" (virtual floppy disk images) and ".vhd" (virtual hard drive images) files you plan to be using. Also, you will need space for DriveWire4 if you are planning to use this system as well. Today's computers should have no problem with storage space.
3. On older Windows systems (XP), We suggest at least 512 meg of memory. On newer systems (Vista and greater), We suggest at least 1 gigabyte or memory, and 4 gigs for heavy users, especially if using DriveWire4.
4. DirectX 9 or OpenGL. Most Windows PC graphics systems will work with VCC. The known incompatibilities seem to affect full screen mode. Even if full screen mode has issues, Vcc should run well in windowed mode. Newer releases of VCC (2.1.9.2 and up) are built with OpenGL instead of DirectX. There are flags in BuildConfig.h (see sources) that can be adjusted to build either OpenGL or DirectX.
5. If you are using VCC with DriveWire4, an internet connection is also desirable.



# VCC Installation

---

VCC is distributed in a self-installing executable package. Just run VCC setup and follow the prompts. Windows will install all components and provide a Desktop icon for starting VCC. If you want to install VCC in a different location than the default, choose the "Custom" option when installing, otherwise, VCC will normally install to "C:\Program Files (x86)\VCC" on your hard drive.

Alternately, you can download the VCC-2.x.x.x.zip file and install VCC as you please. This can be unzipped to your desired folder. All .dll's and supplied .rom files should be kept in the same folder as Vcc.exe. Newer versions of Windows may require you to mark the files as "safe."

VCC allows several setup configurations with RGB or Composite monitor emulation, 6809 or 6309 CPU, from 128k to 8 megabytes of memory, and many others. In these next sections we will try to step through each option and explain its function.

VCC normally keeps its settings in the "%AppData%\VCC\vcc.ini" file which allows VCC to update and keep settings without special permissions. You can also save and load settings with different files so multiple configurations can be kept.

**IMPORTANT:** VCC versions starting with 2.1.9.0 will no longer run on XP unless built using a depreciated version of Visual Studio. For convenience a zip file containing a "legacy" version that will run on XP can be found on github. For VCC version 2.1.9.0 the vcc-2.1.9.0\_legacy.zip contains binaries built for XP but the installer and the vcc-2.1.9.0.zip file contains binaries build for newer Windows systems.

## The VCC Installer

---

The VCC team uses Inno to create the installer. Inno is an open source project that is used by thousands of software distributors, including many commercial software houses. Care has been taken to make the VCC installation as easy as possible.

When first run, the VCC installer requests the usual license agreement. After the license agreement, a text file is displayed, explaining a few of the distribution items contained in the package and how to use them. It is best, if you have not read this article, to do so now. You can click "Cancel" at any time to cancel the VCC installation.

Next, you are given the chance to select your installation directory path. The default path is "C:\Program Files (x86)\VCC 2.x.x.x". To install in a different directory, click the "Browse" button to the right and select your directory.

The next dialog let's you choose the components to be installed. Checking or unchecking the boxes determines which components the installer will use. These choices are:

- VCC Coco 3 Emulator
  - Support Files for the VCC Coco 3 6809-6309 Emulator
- Vcc User Guide
- Coco File Utilities for PC
  - WImg Tool for OS9 Disk Images
  - WImg Tool for MSDOS Disk Images

Basically, you want to install all files, as all in the first category are needed, and the manual includes updates to the current version being installed. This manual will be installed to "Documents\Coco Manuals". Clicking "Next" continues the installation.

The next dialog allows you to select the name of the "Start Menu" group in which the VCC shortcut will reside. This also determines the name given to your VCC shortcut. Unless you have more than one installation of VCC, we suggest this be left as is. Click "Next" to continue.

This dialog lets you select if you want a "Desktop Shortcut" or not. The shortcut will be named using the above selected name. Click "Next" to continue.

This final dialog is your last chance to cancel, or change your VCC setup. A review of your previous choices is shown to make sure this is what you want.

Clicking "Back" will take you to previous dialogs, allowing you to make changes to your setup. Once you click "Install", the installer will install the VCC files in the manner you have chosen.

In the installation, if another VCC folder of the same name is present, the installer will overwrite any previously installed files with the new installation, and your old version of VCC will be updated. Any old configurations from the previous version will be maintained.

Before the installation proceeds, you may or may not be greeted by a standard Windows permission dialog asking if you want to allow this installation to proceed. Just click "Allow" or "OK" and installation will continue.

The VCC Color Computer 3 installation is complete.

## Package Contents

---

The VCC installation package and VCC Zip package contain these files:

- VCC 2.x.x.x Coco 3 Emulator (Installed to the VCC program folder)

- Vcc.exe (The VCC emulator itself)
- wimgtool-os9.exe (OS9 Disk Utility)
- wimgtool-rsdos.exe (RSDOS Disk Utility)
- acia.dll (Tandy RS232 pak emulation)
- becker.dll (Becker Port Cartridge for DriveWire4)
- fd502.dll (Tandy FD-502 Disk Controller)
- harddisk.dll (Generic Hard Drive Controller)
- mpi.dll (Tandy Multi-Pak Interface)
- orch90.dll (Orchestra90cc Program Pack)
- Ramdisk.dll (512k External Ram Disk Cartridge)
- SuperIDE.dll (Cloud9 IDE Hard Drive Controller)
- GMC.dll (Game Master Cart by John Linville)
- SDC.dll (A simulation of the SDC floppy emulator)
- coco3.rom (the Coco 3 BASIC ROM, auto loaded)
- disk11.rom (the Disk BASIC Rom, used by the FD502.dll, auto loaded)
- hdbdw3bc3 w-offset 5A000.rom (variant of the HDBDOS ROM, see ROM descriptions)
- hdbdw3bc3.rom (variant of the HDBDOS ROM, see ROM descriptions)
- hdbdw3bck w-offset 5A000.rom (variant of the HDBDOS ROM, see ROM descriptions)
- hdbdw3bck.rom (variant of the HDBDOS ROM, see ROM descriptions)
- hdblbba.rom (variant of the HDBDOS ROM for SuperIDE, see ROM descriptions)
- orch90.rom (Orchestra90 ROM used by Orc90.dll, auto loaded)
- rgbdos.rom (RGBDOS ROM for VCC's harddisk.dll)
- rs232.rom (Rom for Tandy RS232 program pak)
- cyd\_gmc.rom (optional demo rom for the GMC cart)
- sdc-dos.rom (Rom for SDC simulator)
- license.txt (License Agreement)
- ReadMe.txt (ReadMe file displayed in installation)
- VCC-UserGuide (this document)
- Release-Notes.txt (a brief listing of changes with each VCC version)

# VCC's Menus

---

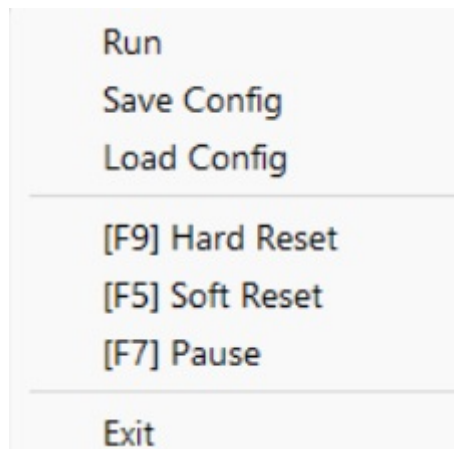
Through it's additional modules (Cartridge and DLL files) and configuration settings, VCC can be set up to mimic a variety of custom Color Computer 3 setups. Vcc supports a limited number of command-line parameters but none are needed to fully configure Vcc.

The VCC MenuBar resides across the top of the VCC window. Here you'll find menus and choices for configuring the various functions of the VCC emulator. All of VCC's options start here. Any changes you make are remembered and recalled each time you run VCC.

Following is a discussion of the menus and choices for configuring VCC.

## VCC's File Menu

---

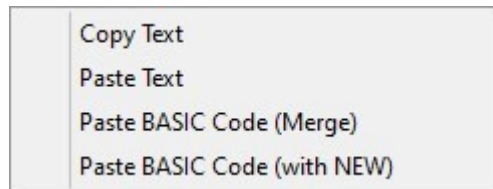


- **Run** - Turns on the VCC Color Computer 3's emulation. It is not necessary if you have "Auto Start Emulation" check in "Cpu Config."
- **Save Config** - Saves you current configuration to the "VCC.ini" file or you can save a custom ini file of your own to recall at any time.
- **Load Config** - You can load any custom configuration file you may have saved. As the default, "VCC.ini" is loaded automatically on startup.
- **[F9] Hard Reset** - Toggles the Coco off and then on, which hard resets the emulation. Anything in memory will be lost. If the F9 key is used it must be pressed twice: once for off, and again for on.
- **[F5] Soft Reset** - This is the same as hitting the reset button on your Coco 3.
- **[F7] Pause** - Pauses the emulation. (Toggles)
- **Exit** - This ends the VCC emulation and returns you to Windows. Any unsaved data or programs will be lost as the program ends with no prompts and control returns to the

Windows OS.

## VCC's Edit Menu

---



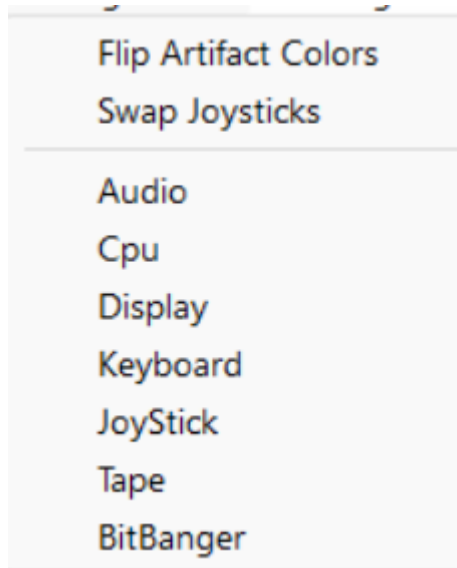
The **Edit Menu** enables text editing options for VCC. It contains probably the most requested feature for VCC: copy and paste

- **Copy Text** - Clicking this item immediately copies *ALL* text on the VCC screen into your PC's clipboard for pasting into your PC text documents. (RSDOS and OS-9/NitrOS9)
- **Paste Text** - Click this item will paste text from your PC's clipboard onto your VCC's screen as if you typed it in. Good for command line entries and works well in most text editors. This function does not work well for pasting BASIC listings; see below. (RSDOS and OS-9/NitrOS9)
- **Paste BASIC Code (Merge)** - Clicking this will paste a BASIC listing from your PC's clipboard onto VCC's screen just as if you typed it, merging your file with any file that may already be in memory. It works very similar to DECB's "MERGE" command. (RSDOS only)
- **Paste BASIC Code (with NEW)** - Clicking this item will first issue a **NEW** command to wipe any program in memory and then pasting the contents of your PC's clipboard. (RSDOS only)

**NOTE:** Copy and paste **ONLY** work on the Coco 3's hardware text screens (both RSDOS and OS-9/NitrOS9). You **cannot** copy or paste from or to a Coco 3 graphics screen.

## VCC's Configuration Menu

---



After installation, when you start VCC, you will be greeted with the familiar "green screen" and Tandy/Microsoft/ Microware Extended BASIC logo. You have basically started a Tandy Color Computer 3 (Coco3) just as it came out of the box with 128k and *nothing* attached. From here you have *many* choices and they all start with the "Configuration" and "Cartridge" menus at the top of the VCC window.

First, you want to configure the Coco3 to the basic machine you desire; whether you run RSDOS, OS-9, or both, it really doesn't matter. Maybe you want to emulate your actual Coco3 setup, or maybe you want to setup that *Dream* System you never had. This starts with the "Configuration" menu:

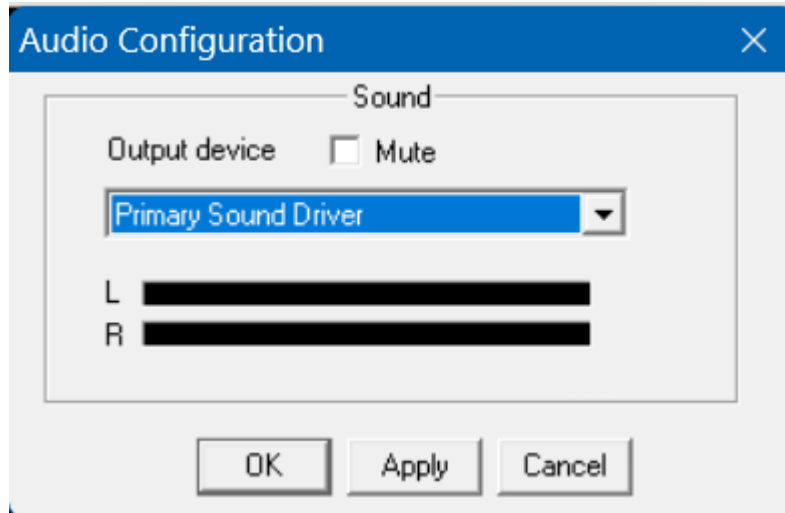
### Flip Artifact Colors

This will toggle PMODE artifact colors to the alternate set. Handy for games. Shifted F6 key will also toggle artifacts.

### Swap Joysticks

This will swap the joystick settings from left to right. Shifted F7 will also swap joysticks.

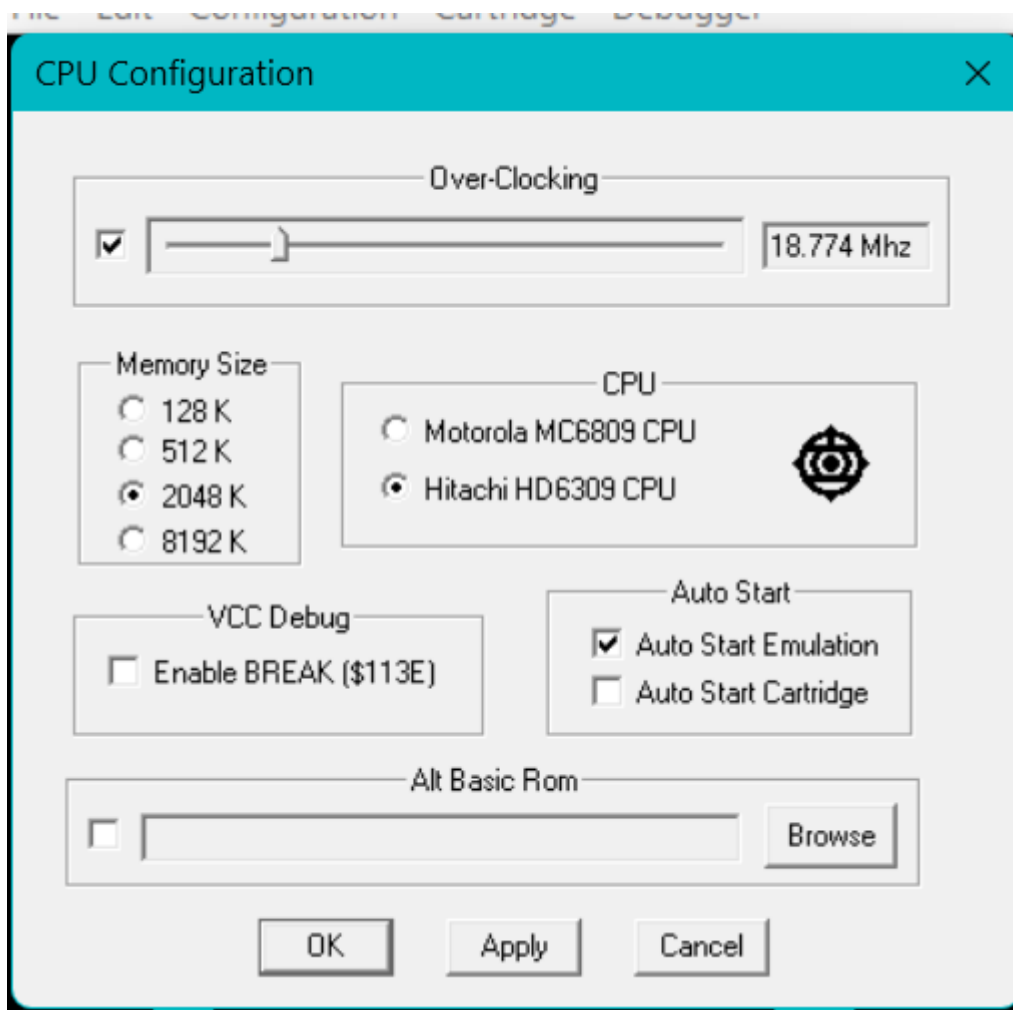
## Audio



This can usually be left as is. It automatically recognizes your Windows computer defaults and should not need to be changed. However there are those users who may have more than one audio output or card and may want to change these setting to suit their needs. For instance, a musician & recording engineer might have several sound options on the computer.

- **Output Device** - A drop down menu that shows your PC system's choices for sound output devices. Most will use the "Primary Sound Drive" used by most Windows programs.
- **Mute** - Disables audio output.
- **VU Meters** - (horizontal bars) These meters are a visual representation of the sound volume in VCC.

## CPU



This allows you to select various Coco 3 system defaults, such as memory, CPU type, and overclocking.

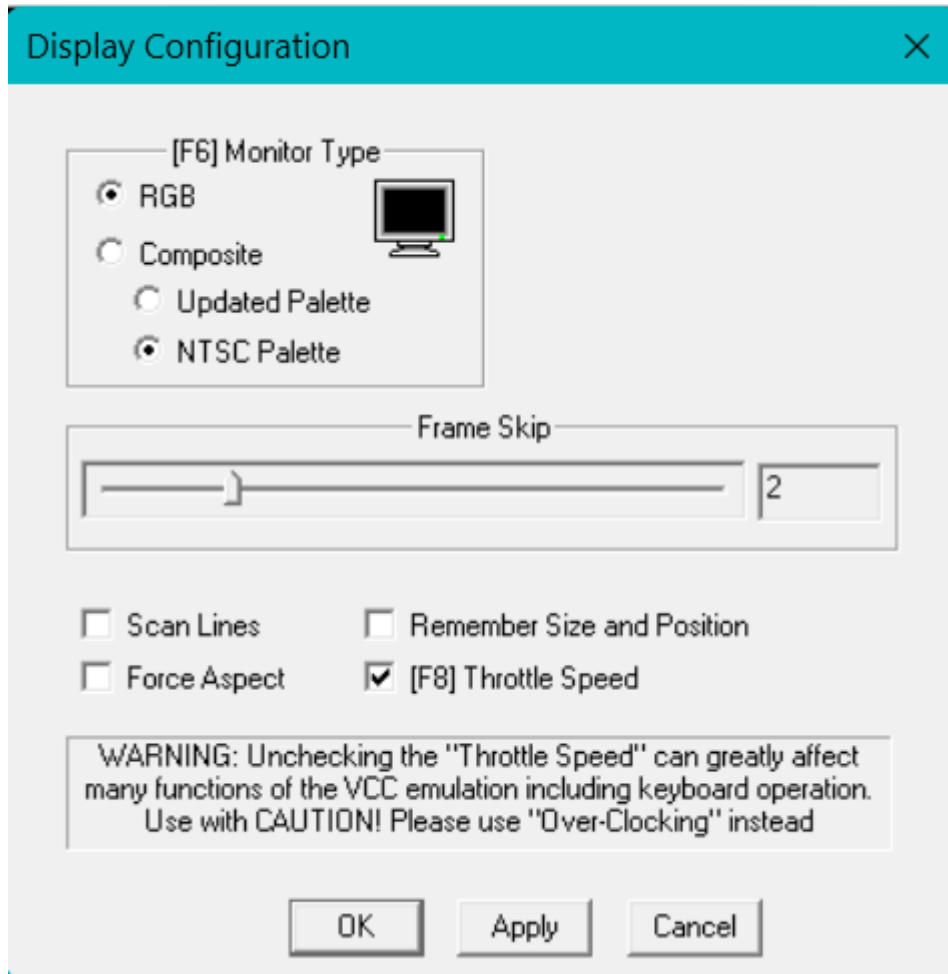
- **Over-Clocking** - The slider will set the speed at which the CPU operates when double speed is enabled. Minimum is 1.778Mhz, maximum is 89.400Mhz. This can also be accessed via the F3 and F4 keys. Checkbox or shifted F8 key can be used to toggle overclocking enable.
- **Memory Size** - Set the maximum amount of memory your Coco3 will be allowed to use.
  - **128k** - Emulates a stock Coco3 straight out of the box
  - **512k** - Emulates the 512k upgrade available through Tandy and many 3rd party vendors
  - **2048k** - Emulates the "Disto" 2 meg upgrade
  - **8192k** - Emulates Paul Barden's 8 meg board
- **CPU**
  - **Motorola MC6809 CPU** - Emulates the stock CPU installed in the Coco3.
  - **Hitachi HD6309 CPU** - Emulates the aftermarket Hitachi 6309 instruction/speed enhanced CPU.



- **Enable Break** - Checking this box enables the lwasn BREAK instruction (\$113E). If checked this instruction causes the emulator to halt.
- **AutoStart Emulation** - Unchecked starts VCC in an "Off" state. You must press "F9" or use the File Menu to turn the emulation on. Checked, VCC will start with the power "On".
- **AutoStart Cartridge** - Unchecked, program paks will not "autostart". This simulates disabling the cartridge detect pin on a cart. This is useful for using the Orchestra90 cart emulation for stereo sound. When using the MPI (MultiPak Interface) module, the carts will only autostart when the MPI slot switch is set to the slot in which the cart is inserted. When using the "Quick Load" command line argument for .CCC or .ROM files autostart must be checked for Vcc to autostart these cart images.
- **\*\* Alt Basic Rom \*\*** - Allows setting an alternate coco3.rom. Useful for trying out a custom Coco3 rom or custom rom location.

**WARNING: If the CPU type or memory size is changed the Coco3 will reset and all memory contents will be erased.**

## Display

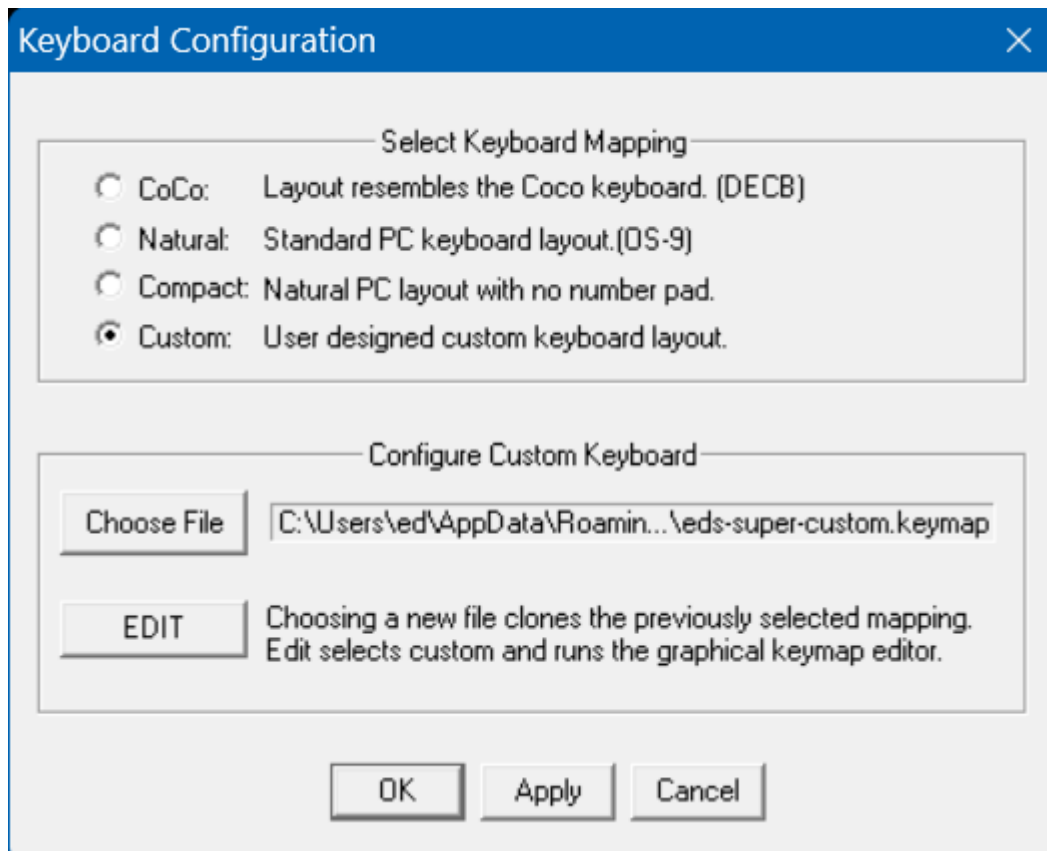


Allows you to set your VCC windowed PC display to match your choice of standard Coco 3 display types

- **[F6] Monitor Type:** - Allows you to choose your monitor type.
  - **RGB** - Selects an RGB color monitor such as the Tandy CM-8.
  - **Composite** - Selects a composite video monitor such as a color TV.
- **Frame Skip** - Number of frames to skip in rendering the Coco3 screen (I've seen no use for this and assume it was from the days of slower computers, we may remove it in the future).
- **Scan Lines** - Allows VCC to "blank" the odd scans to the screen (as a real Coco would).
- **Force Aspect** - Forces VCC to keep the screen in the proper proportions when resized. The "Full Screen" mode will NOT retain it's aspect and will stretch.
- **Remember Screen Size** Allows you to resize the VCC window and exit, the VCC will remember your previous screen size and return to it the next time you run VCC.
- **[F8] Throttle Speed** - Turns off all speed restraints and allows VCC to run at the speed of the host PC, including the keyboard rollover, which may cause pressed keys to repeat at an

unusable rate.

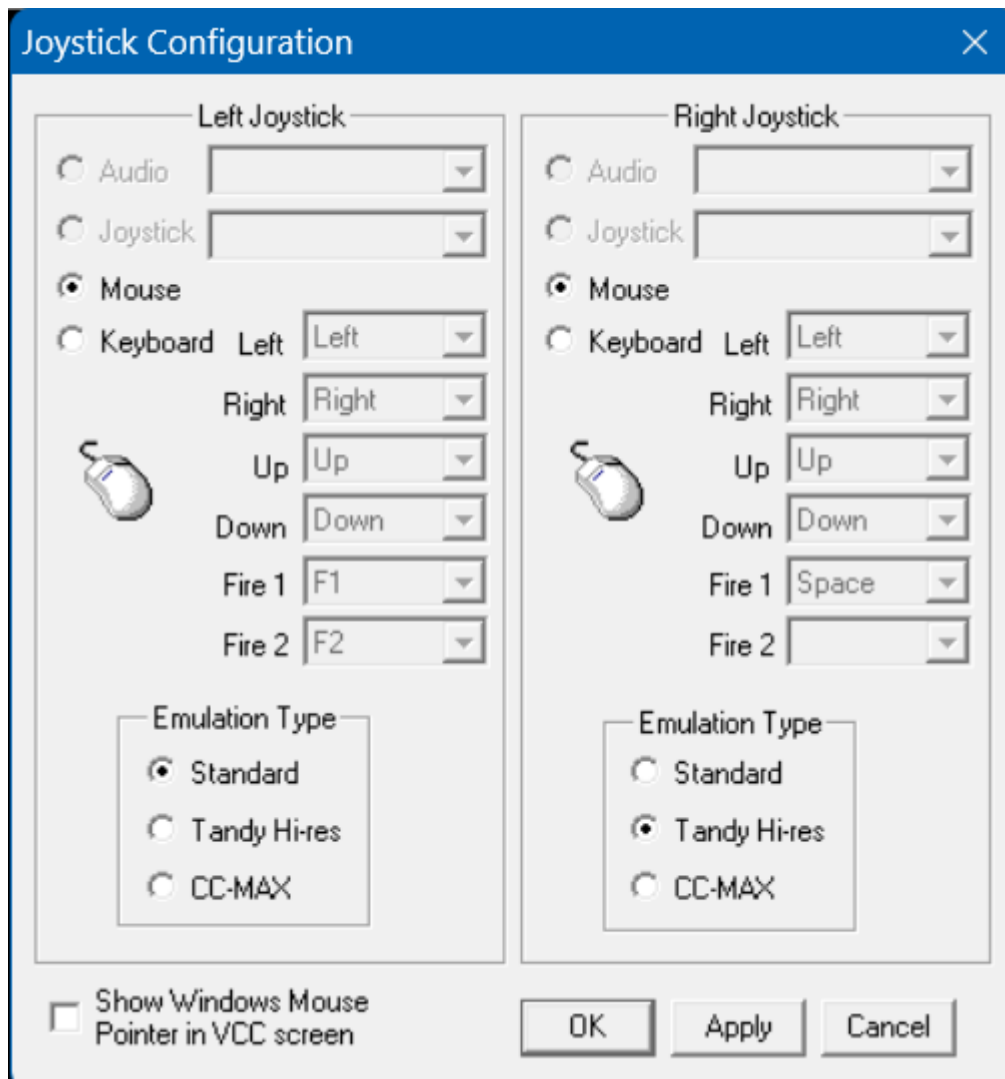
## Keyboard



Allows you to remap your PC keyboard to different keymaps to emulate the layout of a real Coco 3 keyboard, or use it similar to an AT or PS/2 keyboard interface. Keyboard mapping is selected by choosing one of the radio buttons. *(See the "VCC Keyboard Map" section for actual keymaps)*

- **Coco (DECB)** - The PC keyboard is remapped to emulate the layout of an actual Coco 3 (as much as possible)
- **Natural (OS-9)** - The PC keyboard is mapped "as is" with the exception of a few special keys.
- **Compact (OS-9)** - Same as above but mapped with consideration of laptop keyboards with compact key sets
- **Custom** - Allows loading of custom keymaps (see "KeyMap Editor") To edit the custom map one of the following buttons can be clicked:
  - **Choose File** - Load a Custom KeyMap file
  - **Edit** - Opens the Custom KeyMap Editor

## Joysticks



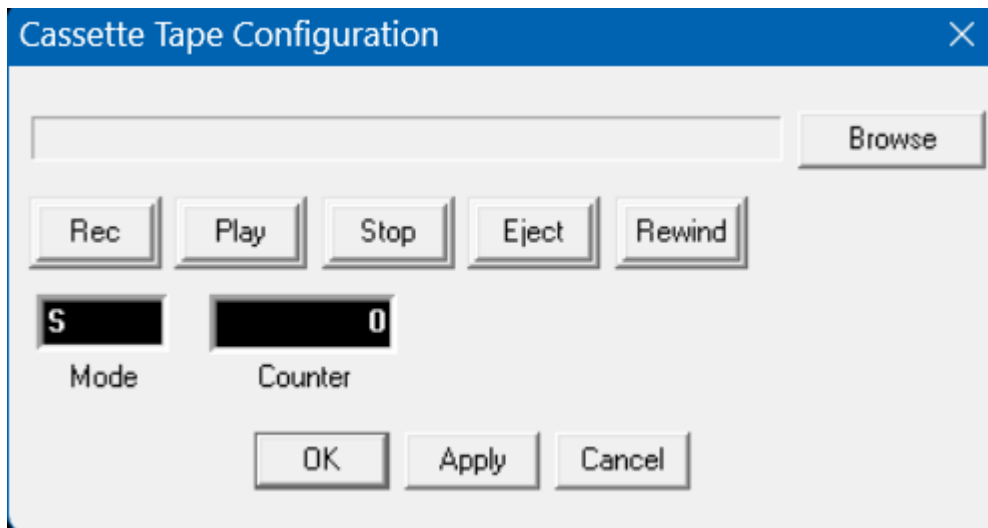
**Joysticks** contains controls for using the mouse or keyboard as Coco joysticks. There are two panels for settings, one for the left joystick and one for the right joystick.

- **Audio** - Disabled. The intent is this will be used to emulate 6 bit audio sampling through the Joystick port similar to some Coco software sampling packages.
- **Joystick** - Allows selection of any joystick connected to the PC.
- **Mouse** - Selects the PC mouse as a joystick
- **Keyboard** - Defines keys on the PC keyboard to be used as a 2 button joystick. WARNING this will disable the normal use of these keys.
  - **Left** - Select key to define Left joystick movement (default <Left Arrow>)
  - **Right** - Select key to define Right joystick movement (default <Right Arrow>)
  - **Up** - Select key to define Up joystick movement (default <Up Arrow>)
  - **Down** - Select key to define Down joystick movement (default <Down Arrow>)

- **Fire 1** - Select key to define Fire button 1 (default <F1 key>)
- **Fire 2** - Select key to define Fire button 2 (default <F2 key>)
- **Emulation Type** - Allows the use of the Tandy & CocoMax3 hi-rez interface emulation
  - **Standard** - Standard Tandy joystick emulation
  - **Tandy Hi-Res** - Tandy Hi-Res interface emulation
  - **CC Max** - CocoMax3 Hi-Res interface emulation
- **Show Windows Mouse Pointer in VCC Screen** - If unchecked the windows mouse pointer is hidden while in the Vcc screen.

**WARNING:** *If you set the Arrow Keys (or any other keys for that matter) to act as Joysticks, those keys will be unusable for anything other than joysticks.*

## Tape



**Tape** is for cassette tape emulation in VCC. Note that the BASIC commands "MOTOR ON/OFF" and "AUDIO ON/OFF" do not affect the VCC tape recorder as on a real Coco.

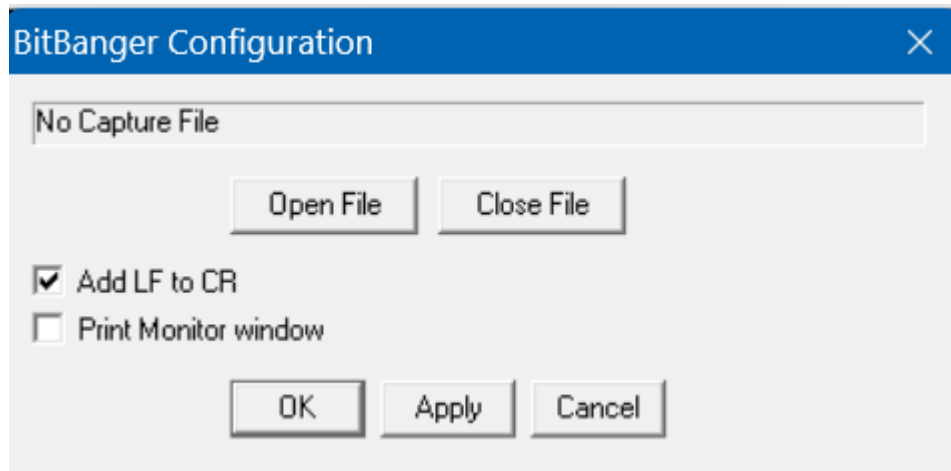
- **Browse** - this button allows you to *browse* your PC for ".cas" or ".wav" files of recorded cassette tape format programs. The 2 types differ in their formats as follows:
  - **wav** - An 8 bit, 44.199 kHz audio recording of the actual cassette file. The file **MUST** be 8 bit & 44.199 kHz (see below).
  - **cas** - A specially encoded, digital representation of the program data. This format is much more compact than wav and loads much faster.>
- **Record** - Sets the tape emulation to record. The tape will stay on "pause" until you issue a "CSAVE" or "CSAVEM" command from basic.
- **Play** - Sets the tape player to "play" and as above, stays on pause until a "CLOAD" or "CLOADM" command is issued fro BASIC.
- **Stop** - This will stop the recorder, just as on a real tape deck. It will not respond to any BASIC commands while stopped. The current "cas" or "wav" file remains in the buffer.
- **Eject** - This button closes then "unloads" the tape image from VCC. You will no longer have access to this image until it's loaded again.
- **Rewind** - This will rewind the "tape" image to the beginning of the tape.
- **Mode** -- Displays the current mode of the tape recorder (Play, Stop, Record Etc)
- **Counter** - Displays the position of the tape file just as a real tape counter would. Handy for tape images containing multiple programs

**WARNING:** If you use ".wav" files, be warned!! If the ".wav" file is NOT recorded in "44,100 kHz, 8-bit", older VCC versions will mangle the file just by loading it, you don't even have to "play" it.

Most "standard" wave files are in 16-bit, 44,199 kHz. These will not work in old VCC versions. You were warned



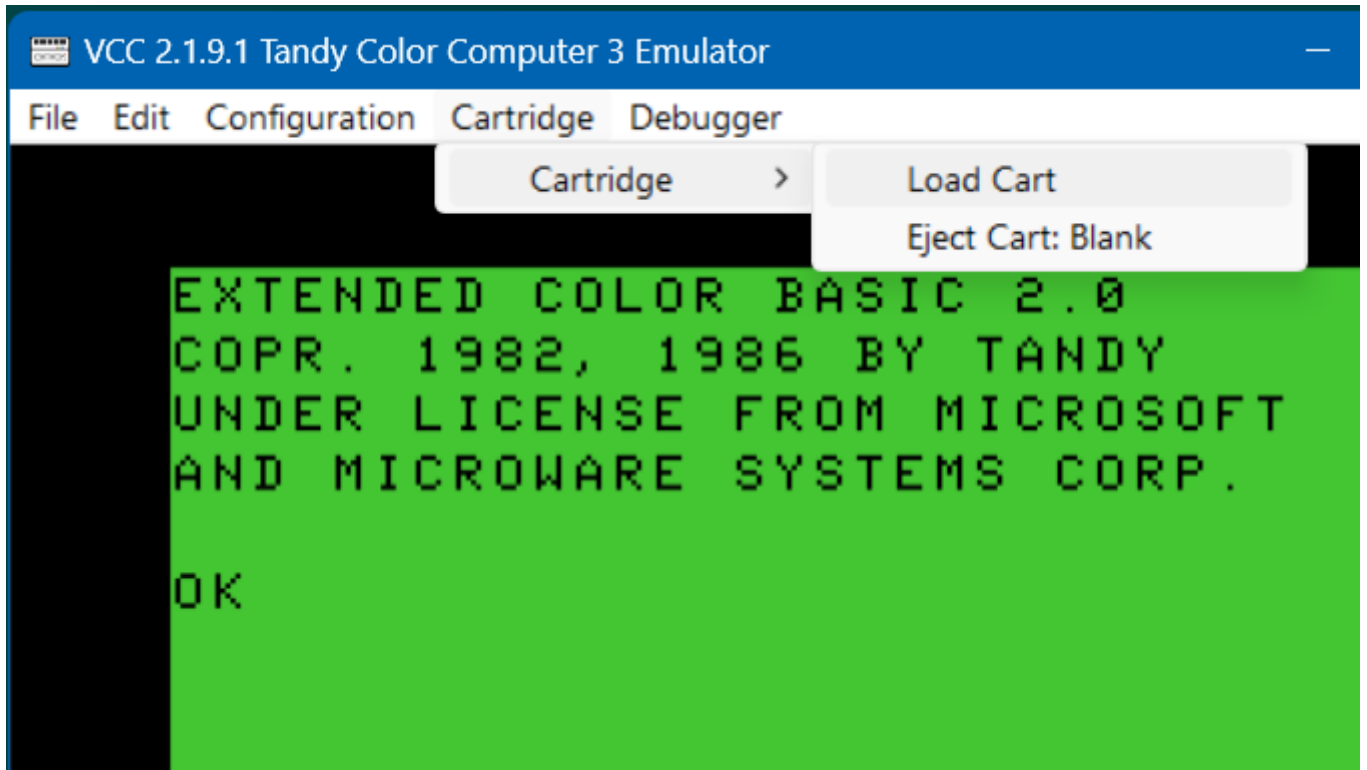
## BitBanger



The BitBanger port is included as a means for VCC to print a Coco text file to a PC text file. All transfers are in raw ASCII text.

- **Open** - This button "opens" a selected capture file. Any data sent to the serial port will be sent to this file in raw ASCII format.
- **Close** - This closes the serial capture file.
- **Add LF to CR** - Checking this item will cause VCC to add a line feed character to every carriage return VCC encounters while sending data to the capture file. PC text files use CR/LF as a standard where the Coco uses CR alone.
- **Print Monitor Window** - Opens an extra window panel on your Windows desktop that will display any data sent to the bitbanger port.

## VCC's Cartridge Menu



The "Cartridge" selection is an emulation of the Coco's cartridge slot. Here you insert cartridge .ROM or .DLL files that emulate various Coco controller carts, such as, the Multipak Interface, Orchestra 90, the FD-502 disk controller, several HD controllers, the Becker Port cart, and RS232 program paks.

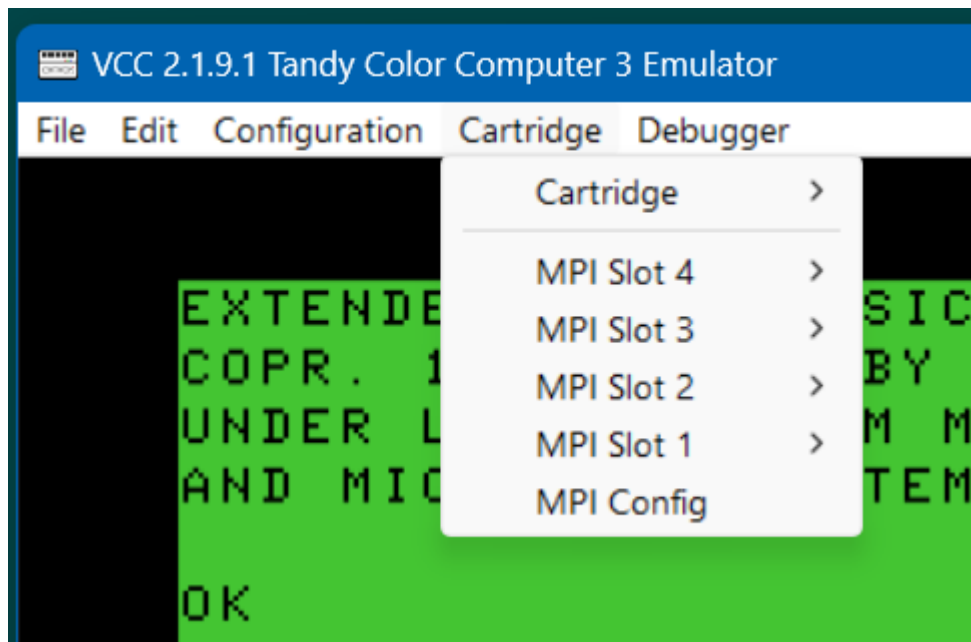
Most Coco cartridges contained read only memory (ROM) for a Coco program that would run as soon as the cartridge was started. This is typical for game cartridges. These are inserted into the Vcc cartridge menu as .ROM image files.

Other cartridges, such as the Multipak interface, Disk controllers, Orchestra 90, and others, also contained extra hardware needed to interface external devices. These are inserted into the Vcc cartridge menu as DLL files. The DLL file contains Windows software that emulates the extra hardware. If the DLL requires a .rom file it will try to automatically install it.

The Cartridge Menu is dynamic, as DLL cartridges are added the menu expands to include items for configuring them.

The DLL files included with Vcc that add to the Cartridge menu are as follows:

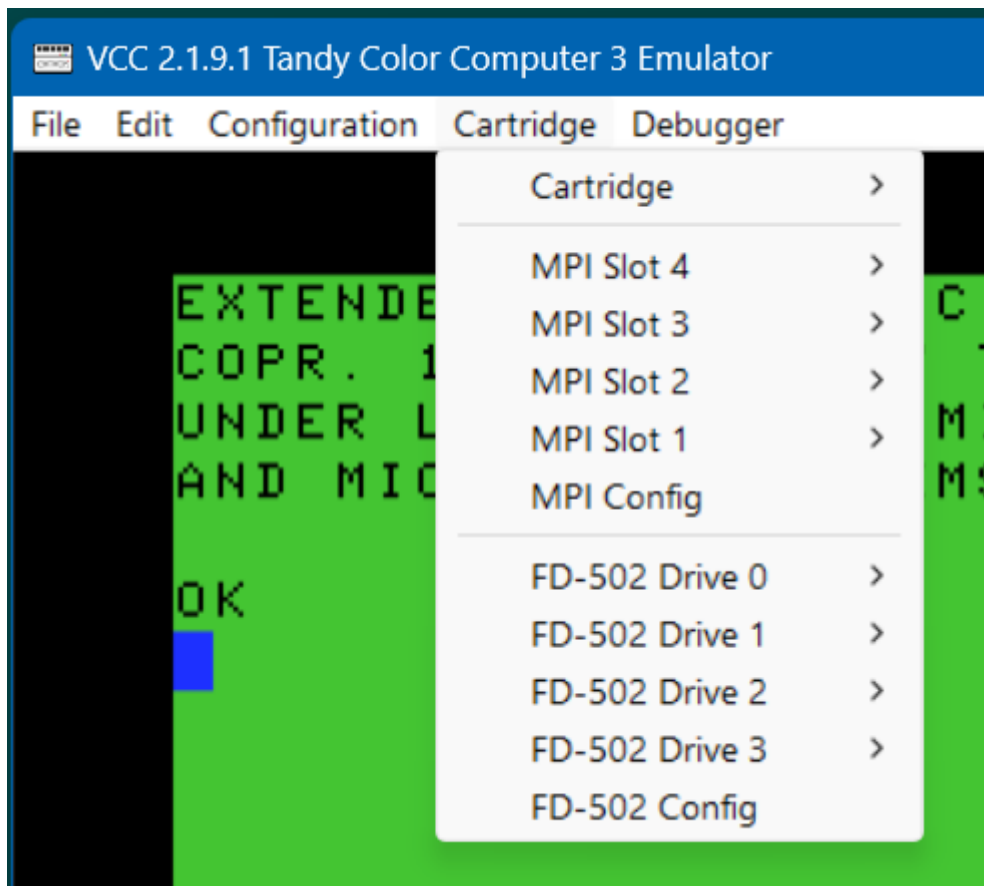
## mpi.dll



The Tandy Multipak Interface. Allows insertion of up to 4 additional ROMS or DLL files. Inserting the mpi.dll module into the emulator cartridge slot add the following entries to the Cartridge menu:

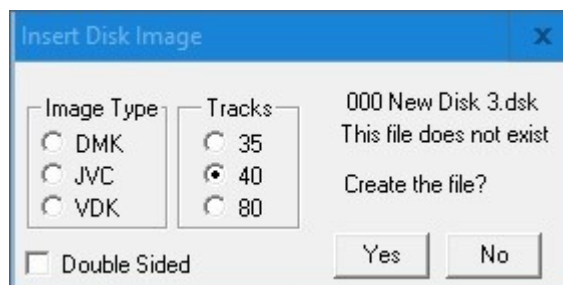
- **Slot 4** - Browse Cart images to load into Slot 4
- **Slot 3** - Browse Cart images to load into Slot 3
- **Slot 2** - Browse Cart images to load into Slot 2
- **Slot 1** - Browse Cart images to load into Slot 1
- **MPI Config** - Displays which carts are inserted and configures the MPI settings
  - **Slot Select** - Selects the Slot to become the *active* Slot. The active slot controls which module will load the rom needed for the module to function using Coco Basic. This action may reset the Coco 3 based on the auto start cartridge setting in Config/Cpu . There is no advantage to selecting a module that does not contain or install rom.
  - **Persistent Pak Images** - Allows VCC to *remember* which carts you have selected and reloads them each time you run VCC. This works unless you remove and reinsert the MPI.
  - **Disable Cart Select Signal** - The Cart Select Signal controls which slot has access to floppy ports. If this checkbox is checked all slots have access to floppy ports.

fd502.dll



Tandy FD-502 Disk controller. Inserting this adds entries to the menu that allow you to insert virtual disk images (.DSK) into drive slots and configure the controller.

- **FD-502 Drive 0** - Browse virtual disk images (.dsk) into Slot 0
- **FD-502 Drive 1** - Browse virtual disk images (.dsk) into Slot 1
- **FD-502 Drive 2** - Browse virtual disk images (.dsk) into Slot 2
- **FD-502 Drive 3** - Browse virtual disk images (.dsk) into Slot 3
- **New Disk Image**



This menu comes up when a non-existing filename is entered when inserting a disk into a slot. Here you can specify an image type and track count for a new disk image and VCC will create

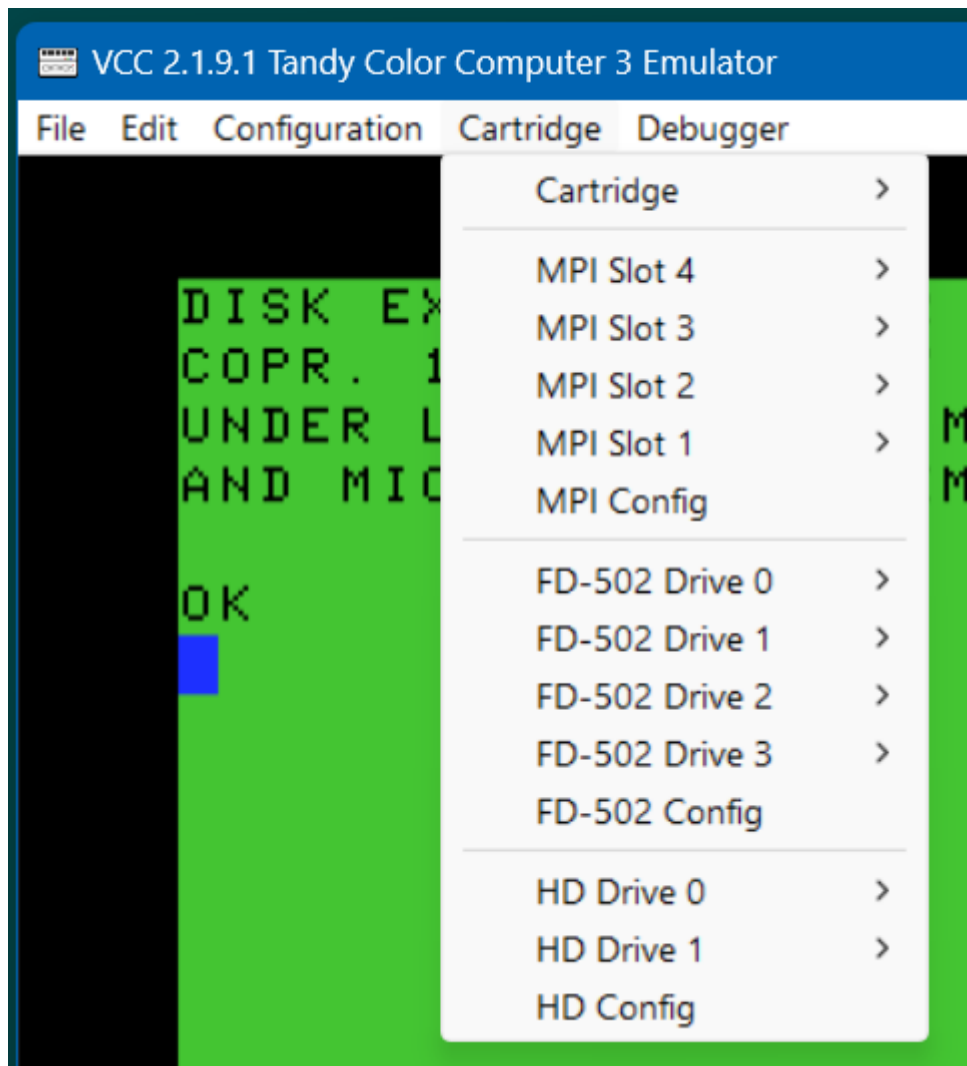
the file.

- **DMK** - DMK virtual disk image type
- **JVC** - JVC virtual disk imagetype (preferred)
- **VDK** - VDK virtual disk image type
- **35** - Creates a 35 track virtual disk image
- **40** - Creates a 40 track virtual disk image
- **80** - Creates an 80 track virtual disk image
- **Double Sided** - Checking this creates a double sided disk

Push "Yes" to confirm our choices or "No" to cancel. Vcc does not format the new disk; the disk must still be formatted using DECB (dskini) or OS9/NitrOS9 (format) commands.

- **FD-502 Config** - Configure the FD-502 controller
  - **DOS Image** - Selects what DOS rom is to be used
    - **External ROM Image** - Loads an external DOS ROM (HDBDOS, RGBDOS, etc)
    - **Disk BASIC** - Use the DECB ROM (disk11.rom)
    - **RGB DOS** - Use the RGBDOS ROM for use with "harddisk.dll"
  - **OverClock Disk Drive** - Checked, allows VCC to use it's virtual drives at PC speeds. Unchecked, simulates normal Coco drive speeds. (default is ON)
  - **Persistent Disk Images** - Forces VCC to load the last disk used when starting VCC (default is ON)
  - **Clock at 0xFF50-51** - Moves the RTC port to \$FF50-\$FF51 for use with some Hard Drive controllers (default is On)
  - **Physical Disks** - Originally meant to use the FDRAWREAD driver for Windows that allowed VCC to read/write to real Coco floppy disks. It is unlikely any modern PC's can access these disks any longer due to a change in format with the introduction of the High Density 1.44meg floppy disk. This function most likely be removed in the future.
    - **A** - Selects Physical Drive A
    - **B** - Selects Physical Drive B
- **External Disk ROM Image** - Browse for an external Disk ROM image to use as an alternative Disk Operating System.

## harddisk.dll



This dll enables hard drive emulation. To use this dll fd502.dll should also be installed if a version of Disk Extended Basic is needed (Such as DOS or RGB DOS)

- **Hard Drive 0** - Insert hard drive images (.vhd) here.
- **Hard Drive 1** - Insert hard drive images (.vhd) here.
- **Create Hard Disk Image**

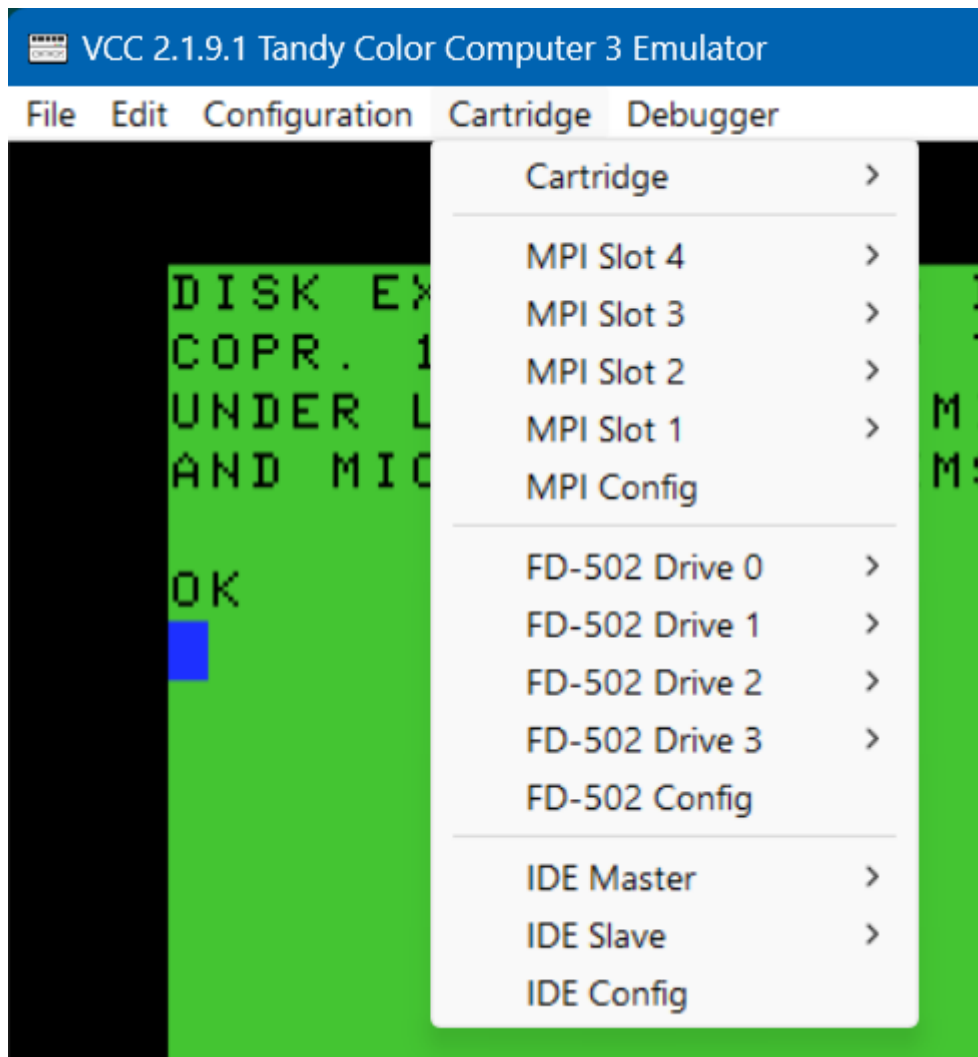


If a non-existing file is typed into a VHD "Insert" selection Vcc will bring up the "Create Hard Disk Image" menu in which you can create a new VHD of that name.

- **Create New File** - Creates the VHD of the specified size.
- **Cancel** - Cancels the operation
- **Initial Disk Size** - Input the size (in Kilobytes) that you want for your VHD. The default size is 132,480k, which is the standard size for an HDBDOS HD with 256 RSDOS disks and a (approximately) 92meg OS-9 HD, with HDBDOS set at an offset of \$5A000. This size can be changed to any size you want, but remember to set your OS-9 HD driver to accommodate your VHD size.

When VHD files are created by Vcc they still need to be formatted using the appropriate utility in DECB or OS9/NitrOS9.

## SuperIDE.dll



Glenside IDE / SuperIDE emulation for using CF card images

- **IDE Master** - Insert CF (master) card image here (.img)
- **IDE Slave** - Insert 2nd (slave) CF card image here
- **IDE Config** - IDE interface configuration settings
  - **Base Address** - Base port address used by the controller. Must NOT conflict with Clock address in FD502 Configuration
  - **Clock at 0xFF70** - RTC port address. Must NOT conflict with clock address in FD502 Configuration.
  - **Clock is Read-Only** - Unchecked allow the clock to be set by the OS.

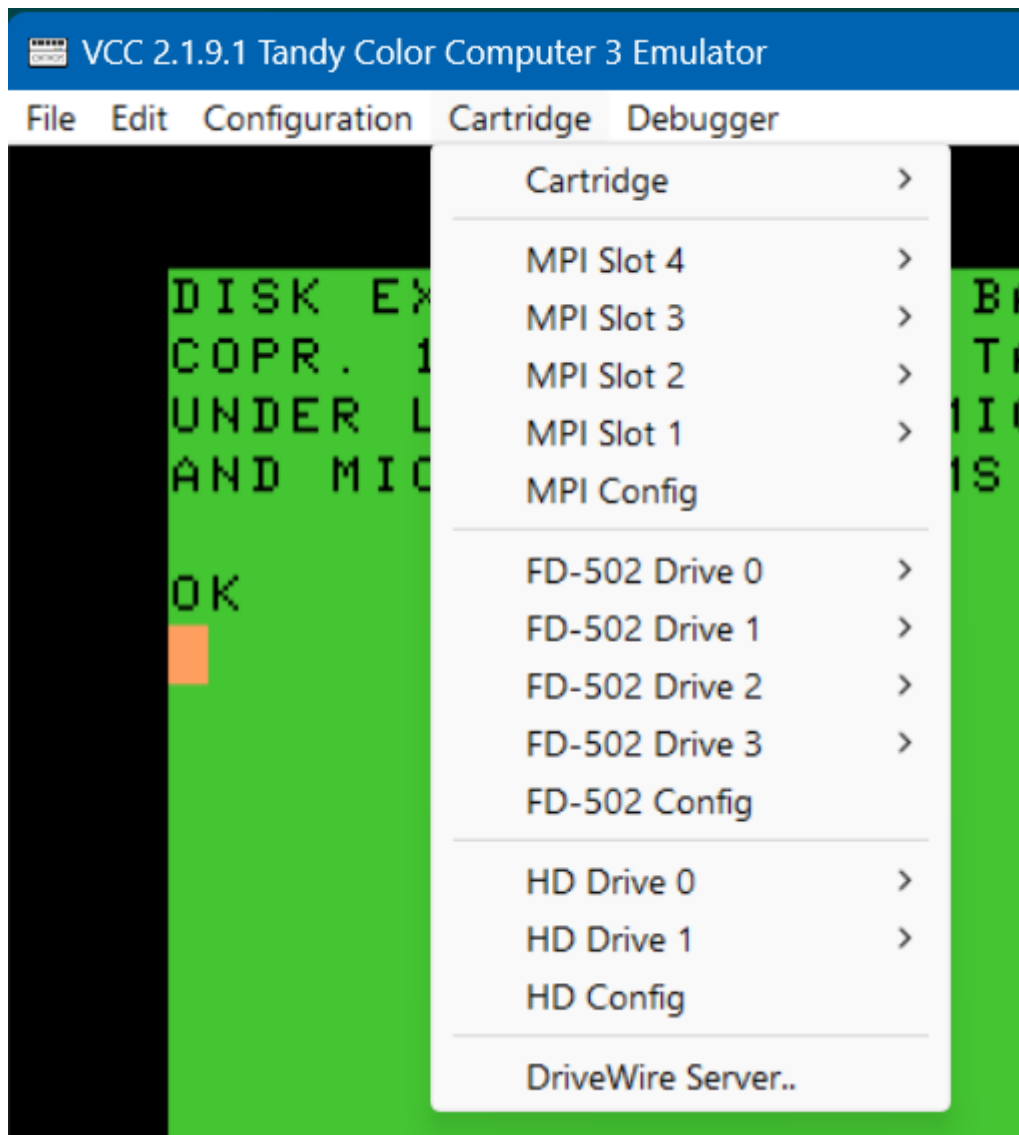
## GMC.dll



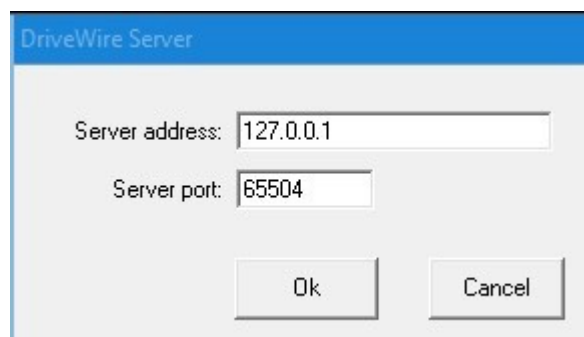
The GMC dll emulates John Linville's "Game Master Cart" (GMC) for custom game cart design with "Syd" type sound & music. **This cart is still experimental, so please report any problems.**

- **Select GMC ROM** - Select and insert the GMC compatible game/sound ROM.

becker.dll



Becker Port emulation for DriveWire4 communication



**DriveWire Server** Selects Drivewire server address and port. Defaults are 127.0.0.1 (localhost) and 65504. The port number must match that used by the Drivewire server to communicate with

the Becker Port. Note that this is a different port than that used by other external applications to communicate with the Drivewire server.

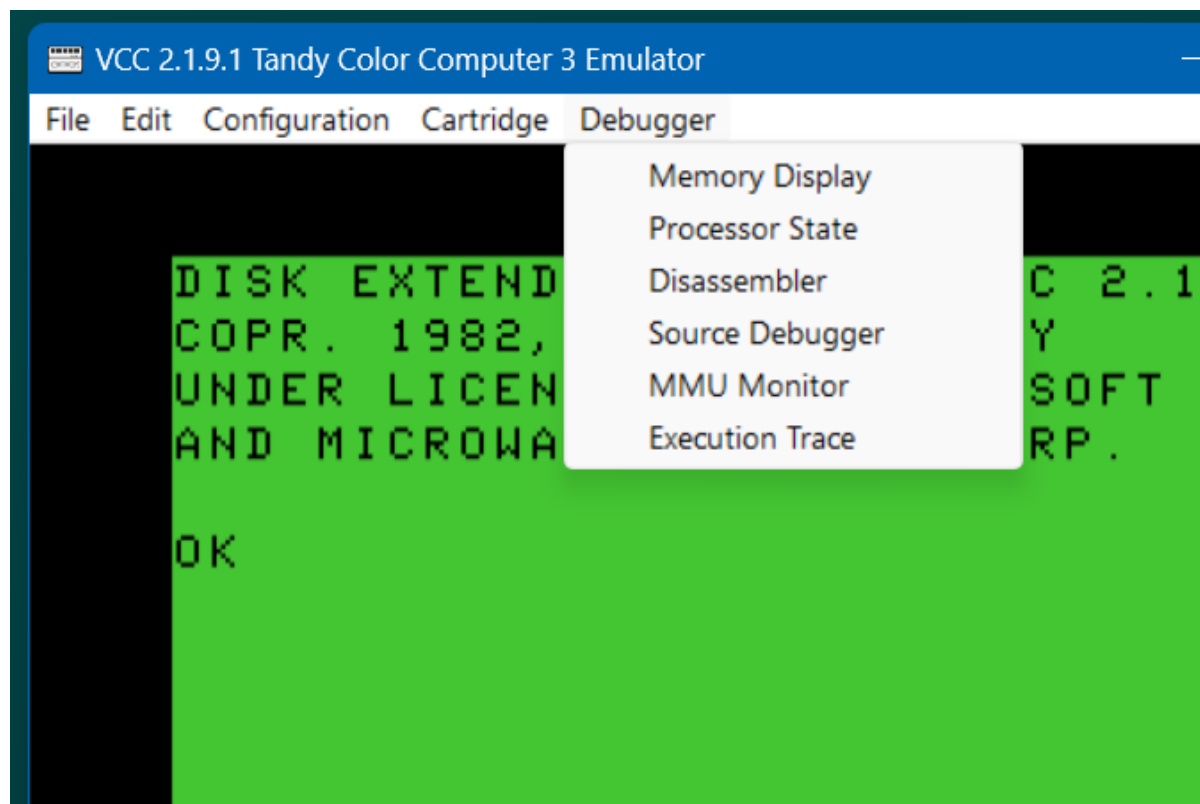
## acia.dll

Cartridge	>
MPI Slot 4	>
MPI Slot 3	>
MPI Slot 2	>
MPI Slot 1	>
MPI Config	
FD-502 Drive 0	>
FD-502 Drive 1	>
FD-502 Drive 2	>
FD-502 Drive 3	>
FD-502 Config	
HD Drive 0	>
HD Drive 1	>
DriveWire Server..	
ACIA Config	

This DLL attempts to emulate the sc6551 Asynchronous Communication Interface Adapter found in the Tandy Deluxe RS232 and Direct Connect Modem Paks. See the ACIA Guide for more details.

- Type: Select the data flow type needed
  - Console
  - File Read
  - File Write
  - TCPIP
  - COMx
- Acia Address - Select the Coco port address to be used
  - \$FF68 (RS-232 Pak)
  - \$FF6C (Modem Pak)
- Name - Used for selecting port modes (see ACIA section in this manual)
- Port - Used for selecting port modes (see ACIA section in this manual)
- Text - Turns on/off the sending of CR/CRLF in the Read/Write file modes.(see ACIA section in this manual)

## VCC's Debugger Menu



The VCC Debugger is one of the latest additions to VCC, and is probably one of the 2nd or 3rd most requested features. With the Debugger, you can view the inner workings of the Color Computer 3 as you run software. This is great for debugging your projects or just viewing how it all works.

**Note:** Interrogating the machine's internals and displaying the results in real time *may* impact the frame rate of the emulator. On reasonably fast machines this will not be an issue. However, if you are playing a graphically intensive game and want to maintain 60 FPS, keep the debugger windows closed until you need them. When the windows are closed, the emulator's framerate is not impacted.

## Memory Display

VCC 2.1.9.1 Tandy Color Computer 3 Emulator

Memory

Close

Find

CPU

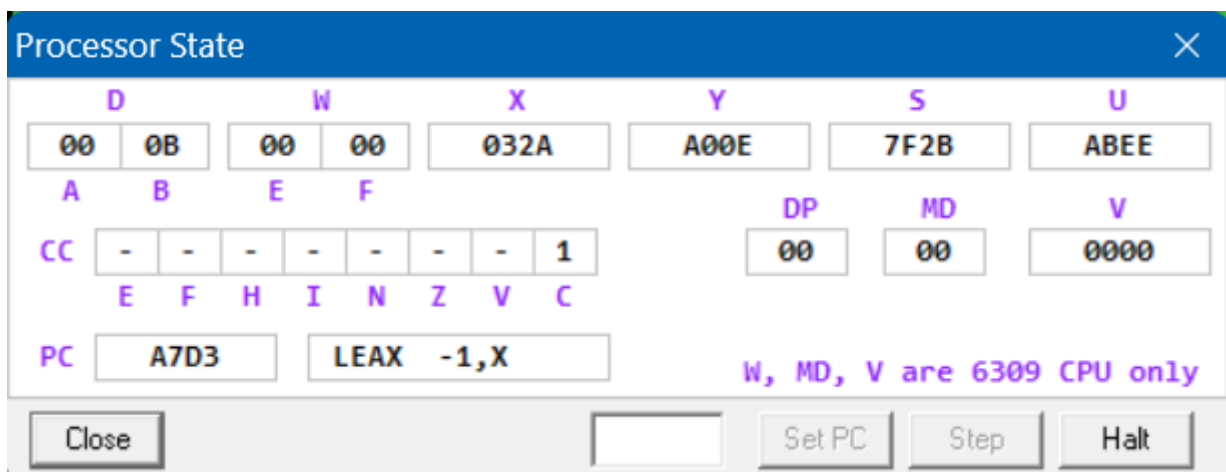
Help

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ASCII
000000	00	22	22	00	00	00	FF	00	00	00	01	A9	01	A4	00		. "" .....
000010	00	00	00	00	00	00	00	00	00	26	01	26	03	26	03	26	....._
000020	03	7F	37	7F	FF	00	00	7F	FF	00	00	00	00	00	00	00	..7.....
000030	00	00	00	26	00	00	00	00	00	00	00	00	00	00	00	00	...._.....
000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000050	00	00	01	A9	00	00	03	00	AB	EE	00	00	00	00	00	00	.....
000060	00	00	AB	EE	AB	F1	00	00	00	00	10	10	00	20	00	00	.....
000070	00	55	C0	E7	7F	FF	00	00	00	00	00	00	00	00	00	00	.U.....
000080	00	00	00	00	00	00	00	00	04	C0	00	00	00	00	00	12	.....
000090	18	0A	00	80	06	00	58	00	01	10	70	84	00	B4	4A	0C	.....X...p...J.
0000A0	A7	26	02	0C	A6	B6	26	00	7E	AA	1A	00	00	00	00	00	....._~.....
0000B0	09	5F	03	00	00	00	00	14	00	10	0E	00	0E	00	00	00	._.....
0000C0	00	00	00	00	00	00	00	00	80	00	60	00	00	00	00	00	.....^.....
0000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	02	BA	.....
0000E0	42	04	02	00	00	00	00	00	00	00	00	00	00	00	00	00	B.....
0000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000100	3B	3B	3B	3B	3B	3B	00	00	00	7E	D8	A1	7E	D8	AF	7E	;;;;;;;;;...~..~..~
000110	A0	F6	78	DB	4A	80	4F	C7	52	59	FF	04	5E	7E	84	89	..x.J.O.RY..^~..
000120	35	AA	66	AB	67	14	AB	1A	AA	29	19	81	83	CF	0A	0E	5.f.g....).....
000130	82	1E	CF	32	14	C1	92	C2	38	06	C2	19	C2	4E	00	4A	...2....8....N.J
000140	B4	B2	77	00	B4	4A	B2	77	B4	4A	B4	4A	B4	4A	B4	4A	..w..J.w.J.J.J.J
000150	B4	4A	FF	FF	FF	FF	FF	FF	FF	FF	00	00	00	00	7E	C4	.J.....~.
000160	4B	7E	C8	88	7E	C8	93	7E	CC	1C	7E	C5	BC	7E	C8	48	K~..~..~..~..H
000170	7E	C8	4B	7E	CA	E9	7E	CA	F9	7E	8E	90	7E	CD	35	7E	~.K~..~..~..~.5~
000180	C8	A9	7E	C6	E4	7E	CA	E4	7E	C9	0C	7E	CE	D2	7E	C6	..~..~..~..~..~.
000190	E4	7E	C2	65	7E	CA	3E	7E	87	E5	7E	C8	B0	39	39	39	..~.e~.>~..~..999
0001A0	7E	C2	B2	7E	83	04	39	39	39	03	00	AB	EE	00	00	00	~..~..999.....
0001B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0001D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0001F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

Selecting the memory display gives a hex memory dump. The CPU dropdown allows you to select CPU, Real, ROM, or PAK to control which is displayed. "CPU" is the memory the cpu sees after the Memory Management Unit has mapped it. "Real" is the actual physical memory. "ROM" is the Coco3.rom before it was copied to physical RAM. "PAK" is the contents of the active cartridge.

Entering an address in the find box will scroll the display to select the byte at that address for editing (if permitted). The byte to edit can also be selected by clicking on the cell containing it. A new value can then be entered and the memory will be modified. PAK contents CANNOT be modified unless a .dll associated with the cartridge enables it.

## Processor State



The Processor State window displays the following registers and values:

D		W		X	Y	S	U
00	0B	00	00	032A	A00E	7F2B	ABEE
A	B	E	F				

CC	E	F	H	I	N	Z	V	C	DP	MD	V
-	-	-	-	-	-	-	1		00	00	0000

PC	Instruction
A7D3	LEAX -1,X

W, MD, V are 6309 CPU only

Buttons: Close, Set PC, Step, Halt

This displays the current state of the processor. Register values are displayed and updated in real time. In addition, you can halt, step, and run the processor at any time. When the processor is halted you can also change the PC register and step or run from that point.

## Disassembler

Disassembler

☐ Real Address

☐ Os9 Mode

Decode

Address

☒ Auto track PC

Help

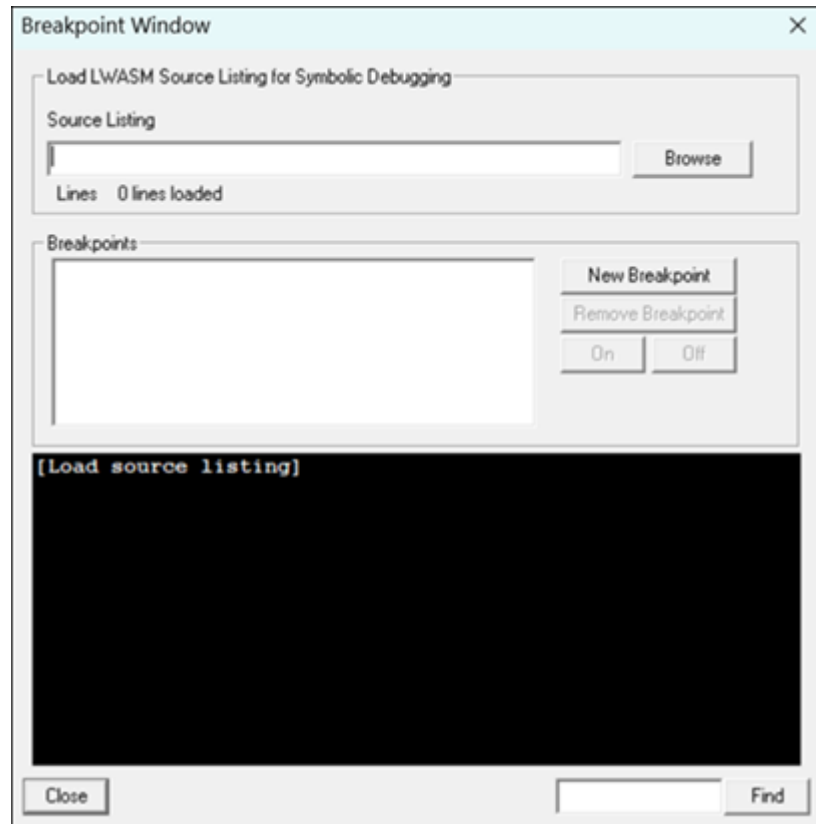
Disassembly is tracking the PC

00A7D3	301F	LEAX	-1,X	0.
00A7D5	26FC	BNE	*-\$02	&
00A7D7	39	RTS		9
00A7D8	1A50	ORCC	#\$50	.P
00A7DA	8DEE	BSR	*-\$10	.n
00A7DC	9E92	LDX	<\$92	..
00A7DE	8D48	BSR	*+\$4A	.H
00A7E0	301F	LEAX	-1,X	0.
00A7E2	26FA	BNE	*-\$04	&z
00A7E4	39	RTS		9
00A7E5	8DF1	BSR	*-\$0D	.q
00A7E7	8D0B	BSR	*+\$0D	..
00A7E9	1CAF	ANDCC	#\$AF	./
00A7EB	B6FF21	LDA	\$FF21	6.!
00A7EE	84F7	ANDA	#\$F7	.w
00A7F0	B7FF21	STA	\$FF21	7.!
00A7F3	39	RTS		9
00A7F4	1A50	ORCC	#\$50	.P
00A7F6	D67D	LDB	<\$7D	V}
00A7F8	D781	STB	<\$81	W.
00A7FA	967D	LDA	<\$7D	.}
00A7FC	2707	BEQ	*+\$09	'.
00A7FE	9E7E	LDX	<\$7E	.~
00A800	AB80	ADDA	,X+	+..
00A802	5A	DECB		Z
00A803	26FB	BNE	*-\$03	&{
00A805	9B7C	ADDA	<\$7C	.
00A807	9780	STA	<\$80	..
00A809	9E7E	LDX	<\$7E	.~
00A80B	8D1B	BSR	*+\$1D	..
00A80D	863C	LDA	#\$3C	.<
00A80F	8D19	BSR	*+\$1B	..
00A811	967C	LDA	<\$7C	.
00A813	8D15	BSR	*+\$17	..
00A815	967D	LDA	<\$7D	.}
00A817	8D11	BSR	*+\$13	..
00A819	4D	TSTA		M
00A81A	2708	BEQ	*+\$0A	'.
00A81C	A680	LDA	,X+	&.
00A81E	8D0A	BSR	*+\$0C	..



This window disassembles a selected range of addresses and allows the setting of breakpoints. It also tracks and displays a disassembly of code that was executing when the CPU was paused. This provides machine level debugging for a loaded program. If you enter an address and press the "Apply" button the window will show you the disassembled code for that address. Addresses must be entered in hexadecimal. The disassembled code can be used to set breakpoints. In addition, if the processor is paused (halted) while the Disassembly Window is open it will automatically disassemble code starting at the paused CPU address. A disassembler tutorial can be found at <https://github.com/VCCE/VCC/wiki>.

## Source Debugger



This window allows you to load an LWASM assembler listing and set breakpoints. This provides a source level debugging capability for a loaded program. To create the listing, you must use `lwasm`. For example:

```
lwasm \--format=decB \--list=sr3.lst \--output=sr3.bin sr3.asm
```

This produces a listing file like this:

6000 1A50	( sr3.asm):00047	START	ORCC	#\$50 * Turn off interrupts
6002 7FFF40	( sr3.asm):00048		CLR	\$FF40
6005 7FFFDE	( sr3.asm):00049		CLR	\$FFDE
6008 FC0112	( sr3.asm):00050		LDD	\$112
600B FD64CC	( sr3.asm):00051		STD	RNUM
600E FD64CE	( sr3.asm):00052		STD	RNUM+2
6011 8655	( sr3.asm):00053		LDA	#\$55
6013 9771	( sr3.asm):00054		STA	<\$71
6015 8E601D	( sr3.asm):00055		LDX	#STRT
6018 9F72	( sr3.asm):00056		STX	<\$72
601A 171560	( sr3.asm):00057		LBSR	HNAM

When the listing is loaded, it scans the file looking for the 4-byte address (6000 in the case of the first line). When it finds a line with an address it will make note of the listing line number. When a breakpoint address is sent to the CPU and the CPU's Program Counter (PC) matches the breakpoint address, the CPU will halt.

When the CPU halts, the breakpoint window will scan the source listing and locate the source line with the matching address. This allows you to see the source exactly as it appears in the listing including any comments.

**Note 1:** The source listing is used as the only means of source debugging. No disassembly of machine code is performed. This means that if the program uses self-modifying code or data as instructions, the source listing window may not reflect exactly what the machine is executing. The memory window will contain the exact bytes seen by the processor, but the source listing may not. However, the breakpoint window will try its best to stay relevant to the area of the code being executed.

**Note 2** The Source Debugger only works for programs currently loaded in memory accessible to the processor. Programs not mapped by the MMU can not easily be debugged with this tool.

## MMU Monitor

MMU Monitor

Real	MAP 0	CPU Memory	MAP 1	Real
070000	38	<—> 0000	38	070000
072000	39	<—> 2000	30	060000
074000	3A	<—> 4000	31	062000
076000	3B	<—> 6000	32	064000
078000	3C	<—> 8000	33	066000
07A000	3D	<—> A000	3D	07A000
07C000	3E	<—> C000	35	06A000
07E000	3F	<—> E000	3F	07E000

FF90:b6 Enable

1

FF90:b3 RAM Vector

1

FF91:b0 MMU Task

0

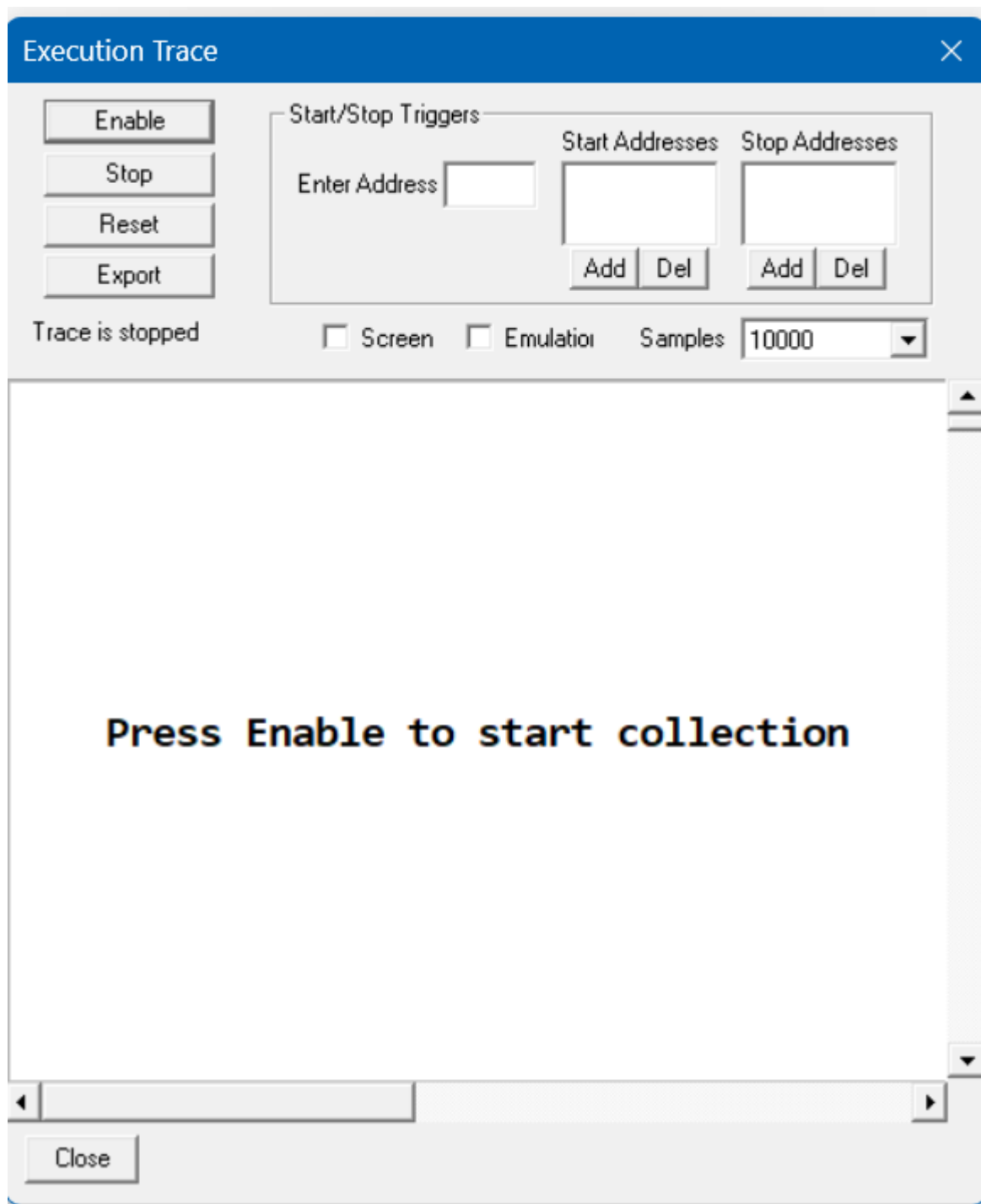
FF90:b1-0 ROM Map

0

Close

Selecting the MMU Monitor option will allow you to see the current state of the Memory Management Unit (MMU). This device handles the mapping of the machine's real memory (128KB or 512KB in most CoCo3s) to the CPU's 64KB memory space. The MMU will map real memory into CPU memory in 8KB pages (8192 bytes). The page numbers are placed in the MMU's 8 registers to make a mapping. The CoCo3's MMU has two sets of 8 registers, so that two maps can always be ready to be used. The mapping used is determined by the MMU Task Bit (0 = MAP 0, 1 = MAP 1). The monitor will display which map is in effect by drawing lines from the MAP column to the CPU memory ranges.

## Execution Trace



This window allows you to record a detailed trace of instructions as they are executed. Pressing 'Enable' will start capture. Stop/Start triggers can be set to control where trace starts or starts. After collecting a trace you can export it to a file.

## Help Menu

---

This menu selection has three choices:

- **About VCC** - Displays the VCC "About Box" which contains general VCC information and copyright and information.
- **Function Keys** - Brings up a help popup showing the VCC function key mappings (Fn)
- **User Guide** - Launches the windows system browser with a link to this document on-line.

## VCC Function Keys



VCC version 2.1.9.2 adds a bunch of new Function keys shortcuts.

- **F1 & F2** - These are mapped to the F1 and F2 keys on the standard coco keyboard. They also can be mapped to Fire 1 and Fire 2 when using the keyboard to emulate joystick input.
- **F3** Decrease the overclock CPU speed. Key repeats until minimum speed is reached
- **F4** Increase the overclock CPU speed. Key repeats until maximum speed is reached
- **F5** Soft Reset. Same as pressing the reset button on a real machine.
- **F6** RGB toggle. Temporarily switches between RGB and Composite. Does not change the ini file setting.
- **F7** Toggle Pause/Run. Stops game play or can be used to step code in the debugger.
- **F8** Toggle Throttle. Normally the emulator is throttled to display 60 frames per second to match a real Coco. Toggling throttle off allows the emulation to run as fast as possible.
- **F9** - Start/Start toggle. When emulation is stopped a "static" screen is shown.
- **F10** - When in full screen mode causes the Cartridge menu to come up allowing user to switch disks, etc without leaving full screen mode.
- **F11** - Toggles between windowed and full screen mode.
- **F12** - Brings up function key help (this).
- **Shifted F5** - Hard Reset. Reloads ROMS and casues restart of CoCo Basic.
- **Shifted F6** - Flip Artifacts. Toggles Red/Blue artifact colors when in Composite mode.

- **Shifted F7** - Swap Joysticks. Left and Right Joystick settings are swapped.
- **Shifted F8** - Toggle Overclocking enable. When disabled high speed poke sets cpu to 1.788 Mhz. When enabled cpu speed can be much higher. A plus "+" is appended the cpu speed on the status line if overclocking is enabled.
- **Shifted F10** - When in full screen mode cause the config menu to come up allowing VCC configuration changes without leaving full screen mode.
- **Shifted F11** - When in full screen mode Toggles a VCC status line at the top of the screen.
- **Shifted F12** - Takes a screenshot.

## Keyboard Configuration and Key Mapping

---

The default VCC keyboard layout tries to resemble the physical layout of the Color Computer 3 keyboard. This works well with Disk Extended Color Basic (DECB) but not so well with the more powerful OS-9 and NitrOS-9 operating systems. Also some PCs do not have number pads or have uncommon layouts. For these reasons VCC provides three built-in keyboard layouts (CoCo, Natural, Compact) and a user-modifiable custom layout.

The Coco (DECB) layout maps the PC's keyboard in a layout similar to an actual Coco 3 keyboard.

Natural (OS-9) maps the PC's keyboard to match the keycaps on a US QWERTY keyboard with the exception of a few special keys.

Compact (OS-9) is similar to Natural, but with some keys altered to compensate for missing the number pad keys on smaller laptops.

The Custom layout is stored in a separate file, while the built-in layouts are hardcoded within the VCC executable. The mapping is selected by a radio button on the VCC configure keyboard window. Also on the configure keyboard window there are two buttons in the Custom Keyboard section, as follows:

The "Choose File" button changes the custom keymap file. If the file exists the keymap is loaded. If the file does not exist a new file is created containing the currently selected keymap. This allows user to create a file that contains a modifiable copy of a built-in keymap.

The "EDIT" button modifies the Custom keymap. Changes made in the editor take immediate effect and are automatically saved to the keymap file unless Cancel is selected within the editor. Using the keymap editor will cause the Custom keymap to be automatically selected.

It is possible to modify custom keymap files using a text editor. This should not be done while VCC is running. Otherwise VCC might overwrite the changes.



## Custom Keyboard Map File

The keyboard map file is used to map host PC key codes into Color Computer 3 keys. This file has the default name "custom.keymap" and normally resides in the user's Vcc appdata directory. The file contains lines of text, each contains four fields (two pairs) separated by one or more spaces or tabs. Blank lines, lines starting with "#", or anything on a line beyond four fields are comments. All other lines are key definitions. The first two fields of a key definition are the PC code and its modifier. The next two fields are the CoCo key and its modifier.

PC keynames start with "DIK\_" and CoCo key names start with "COCO\_"

Key modifiers are specified with a single digit as follows:

0=not modified, 1=shift, 2=control, 3=alt. A list of valid PC and CoCo key names can be found in keynames.h in Vcc sources. Here are some example entry lines:

```
# PC key name Mod  CoCo name  Mod
# -----
DIK_EQUALS      0  COCO_MINUS  1 # "="  Coco
DIK_MINUS       1  COCO_MINUS  2 # "-"  NitroS-9
```

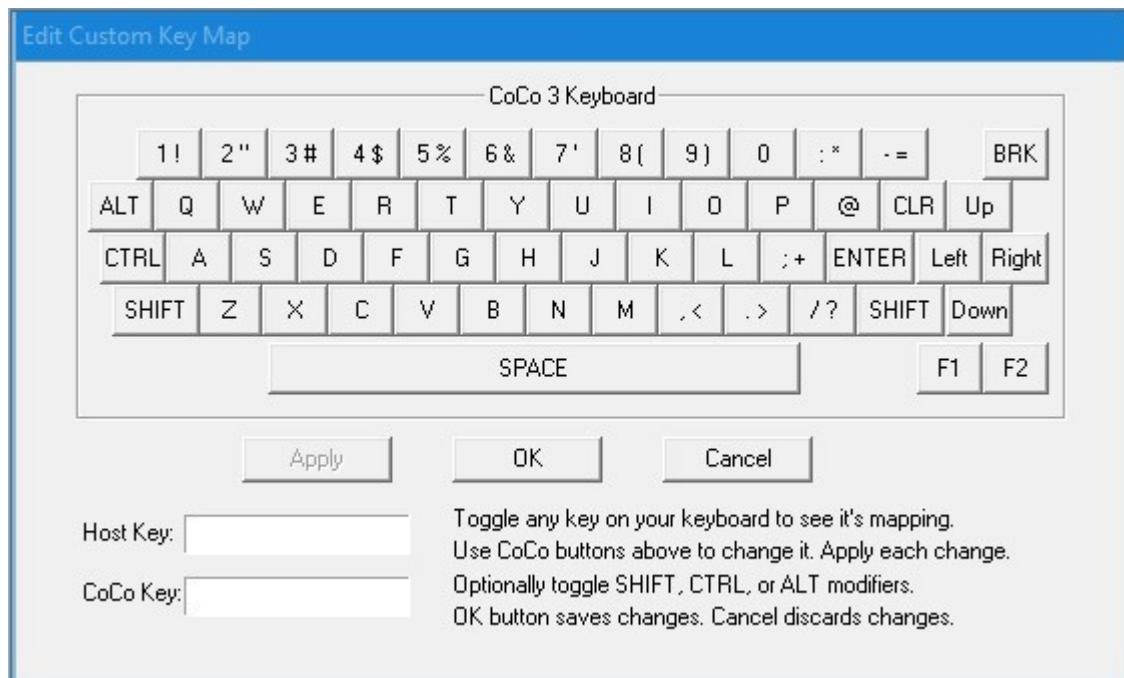
The above example maps PC '=' to 'Shift+' and PC 'Shift-' key to 'Control-' on the emulated CoCo.

Any CoCo key that is not mapped in the keymap file will be dead in Vcc. However shifted letter keys do not need to be mapped to make them uppercase because both DECB and OS-9 will handle this internally.

It is important to note that some PC keys can not be mapped. Specifically the F3-F11 keys can not be mapped, they are used for Vcc control. Keys selected for joystick input via the Joysticks config tab can not be mapped. Adding mappings for any of these keys in the keymap file will have no effect. Also both the right shift and right control keys are automatically mapped to match their left counterparts so they can not be independently mapped.

Whenever Vcc updates the keymap file all comments after the first valid map entry will be removed. This means user added comments should be confined to the head of the file.

## Edit Custom Key Map screen



The Edit Custom Key Map screen is activated by clicking the "EDIT" button on the configuration menu Keyboard tab. The screen shows a virtual CoCo 3 keyboard, a "Host Key" text box, a "Coco Key" text box, and three buttons:

- Apply - To apply an individual key modification.
- OK - To save the changes and exit the editor.
- Cancel - To undo changes and exit the editor.

Pressing a key on the PC keyboard causes it to be shown in the "Host Key" text box and if the key is mapped that will be shown in the "CoCo Key" text box and the key on the virtual keyboard will appear as pressed. Pressing a different PC key (other than a modifier) deselects the previous key and selects the new one. Modifier keys can be toggled either before or after non-modifier keys. This functionality allows easy toggling through various PC key combinations to see the CoCo mapping of each.

Mouse clicks (or touches on a touch screen) on the virtual keyboard are used to modify the PC key mapping. The Shift, Alt, and Control modifier keys can be combined with the PC or CoCo keys being pressed. Clicking a key button when no PC key is selected causes an warning popup to be issued.

Once a key mapping is changed the "Apply" button is enabled. Pressing either the "Apply" or the "OK" button sets the new mapping. "OK" exits the editor so "Apply" should be used if more than one key is to be mapped. All changes are automatically saved to the keymap file when "OK" is pressed and are discarded if "Cancel" is.

## VCC Default Keyboard Layouts

When the keyboard layout is set to "Coco (DECB)", the PC keyboard is reconfigured to that of the Coco 3 as closely as possible. The "Natural (OS-9)" mode is meant to be the full PC keyboard with a few slight modifications. The "Compact (OS-9)" mode is similar to the Natural mode, but configured for laptop computers with limited keys. The layout looks something like this:

### PC Keyboard:

```
[Esc] [F1][F2][F3][F4][F5][F6][F7][F8][F9][F10][F11][F12] [Prnt][Scr][Paus]
[`] [1!][2@][3#][4$][5%][6^][7&][8*][9()][0]] [-_][=+][BkSpc] [Inst][Hom][PgUp]
[Tab][Qq][Ww][Ee][Rr][Tt][Yy][Uu][Ii][Oo][Pp][[{][}]][\|] [Dlet][End][PgDn]
[ Caps][Aa][Ss][Dd][Ff][Gg][Hh][Jj][Kk][Ll][;:] ['"][Enter]
[ Shift ][Zz][Xx][Cc][Vv][Bb][Nn][Mm][,<][.>][/?][ Shift ] [UpA]
[Cntl][Win][LAlt][ Space ][ ][Win][Prp][Cntl] [LftA][DnA][RgtA]
```

### VCC Coco (DECB) Keyboard:

```
[ ][F1][F2][ ][ ][Rst][RGB][ ][Thr][Pwr][StB][FSc][ ][ ][ ][ ]
[ ][1!][2"][3#][4$][5%][6&][7'] [8()][9)][0] [[:*][-=][BkSpc] [ ][Clr][ ]
[ ][Qq][Ww][Ee][Rr][Tt][Yy][Uu][Ii][Oo][Pp][[{][}]][\|] [ ][Brk][ ]
[ Caps][Aa][Ss][Dd][Ff][Gg][Hh][Jj][Kk][Ll][;:] ['"][Enter]
[ Shift ][Zz][Xx][Cc][Vv][Bb][Nn][Mm][,<][.>][/?][ Shift ] [UpA]
[Cntl][ ][LAlt][ Space ][ ][ ][ ][Cntl] [LftA][DnA][RgtA]
```

### VCC Natural (OS-9) Keyboard:

```
[BRK][F1][F2][ ][ ][Rst][RGB][ ][Thr][Pwr][StB][FSc][ ][ ][ ][ ]
[`] [1!][2@][3#][4$][5%][6^][7&][8*][9()][0]] [-_][=+][BkSpc] [INST][Clr][PgUp]
[Tab][Qq][Ww][Ee][Rr][Tt][Yy][Uu][Ii][Oo][Pp][[{][}]][\|] [DEL][EOL][PgDn]
[ Caps][Aa][Ss][Dd][Ff][Gg][Hh][Jj][Kk][Ll][;:] ['"][Enter]
[ Shift ][Zz][Xx][Cc][Vv][Bb][Nn][Mm][,<][.>][/?][ Shift ] [UpA]
[Cntl][ ][LAlt][ Space ][ ][ ][ ][Cntl] [LftA][DnA][RgtA]
```

### VCC Compact (OS-9) Keyboard

```
[ ][F1][F2][ ][ ][Rst][RGB][ ][Thr][Pwr][StB][FSc][ ][ ][ ][ ]
[BRK~][1!][2@][3#][4$][5%][6^][7&][8*][9()][0]] [-_][=+][BkSpc][ ][ ][ ]
[ CLR][Qq][Ww][Ee][Rr][Tt][Yy][Uu][Ii][Oo][Pp][[{][}]][\|] [ ][ ][ ]
[ Caps][Aa][Ss][Dd][Ff][Gg][Hh][Jj][Kk][Ll][;:] ['"][Enter]
[ Shift ][Zz][Xx][Cc][Vv][Bb][Nn][Mm][,<][.>][/?][ Shift ] [UpA]
[Cntl][ ][LAlt][ Space ][ ][ ][ ][Cntl] [LftA][DnA][RgtA]
```

Some changes have been made to the "Natural (OS-9)" keyboard layout to better facilitate its use in NitroS9 and some text editors. These changes are:

- END has been changed from BRK to SHIFT-RIGHT to facilitate the "End Of Line" sequence in NitroS9 and ShellPlus. Also, in the "Ed 3.1" text editor, this key will now produce a true TAB.
- ESC is now mapped as BRK

- ESC has also been mapped to F12 for use with CTRL-BRK in some games and utilities.
- INSERT has been mapped to CTRL-RIGHT
- DELETE has been mapped to CTRL-LEFT
- PAGE UP has been mapped to SHIFT-UP
- PAGE DOWN has been mapped to SHIFT-DOWN

NOTE: The LEFT ALT key is mapped as the ALT key for the Coco 3 emulation. The RIGHT ALT key is mapped to Windows for controlling the Menu Bar in VCC.

## Coco & PC key names used for KeyMaps

PC Scan Code	Value	Comment	PC Scan Code	Value	Comment
DIK_ESCAPE	0x01		DIK_NUMPAD0	0x52	
DIK_1	0x02		DIK_DECIMAL	0x53	. on numeric keypad
DIK_2	0x03		DIK_OEM_102	0x56	<> or \  RT 102-key keyboard
DIK_3	0x04		DIK_F11	0x57	
DIK_4	0x05		DIK_F12	0x58	
DIK_5	0x06		DIK_F13	0x64	(NEC PC98)
DIK_6	0x07		DIK_F14	0x65	(NEC PC98)
DIK_7	0x08		DIK_F15	0x66	(NEC PC98)
DIK_8	0x09		DIK_KANA	0x70	(Japanese keyboard)
DIK_9	0x0A		DIK_ABNT_C1	0x73	/ ? on Brazilian keyboard
DIK_0	0x0B		DIK_CONVERT	0x79	(Japanese keyboard)
DIK_MINUS	0x0C	- on main keyboard	DIK_NOCONVERT	0x7B	(Japanese keyboard)
DIK_EQUALS	0x0D		DIK_YEN	0x7D	(Japanese keyboard)
DIK_BACK	0x0E	backspace	DIK_ABNT_C2	0x7E	Numpad . on Brazilian keyboard
DIK_TAB	0x0F		DIK_NUMPADEQUALS	0x8D	= on numeric keypad (NEC PC98)
DIK_Q	0x10		DIK_PREVTRACK	0x90	(DIK_CIRCUMFLEX on Japanese KB)
DIK_W	0x11		DIK_AT	0x91	(NEC PC98)
DIK_E	0x12		DIK_COLON	0x92	(NEC PC98)
DIK_R	0x13		DIK_UNDERLINE	0x93	(NEC PC98)
DIK_T	0x14		DIK_KANJI	0x94	(Japanese keyboard)
DIK_Y	0x15		DIK_STOP	0x95	(NEC PC98)
DIK_U	0x16		DIK_AX	0x96	(Japan AX)
DIK_I	0x17		DIK_UNLABELED	0x97	(J3100)
DIK_O	0x18		DIK_NEXTTRACK	0x99	
DIK_P	0x19		DIK_NUMPADENTER	0x9C	Enter on numeric keypad
DIK_LBRACKET	0x1A		DIK_RCONTROL	0x9D	
DIK_RBRACKET	0x1B		DIK_MUTE	0xA0	Mute
DIK_RETURN	0x1C	Enter on main keyboard	DIK_CALCULATOR	0xA1	Calculator
DIK_LCONTROL	0x1D		DIK_PLAYPAUSE	0xA2	Play / Pause
DIK_A	0x1E		DIK_MEDIASTOP	0xA4	Media Stop
DIK_S	0x1F		DIK_VOLUMEDOWN	0xAE	Volume -
DIK_D	0x20		DIK_VOLUMEUP	0xB0	Volume +
DIK_F	0x21		DIK_WEBHOME	0xB2	Web home
DIK_G	0x22		DIK_NUMPADCOMMA	0xB3	, on num keypad (NEC PC98)
DIK_H	0x23		DIK_DIVIDE	0xB5	/ on numeric keypad
DIK_J	0x24		DIK_SYSRQ	0xB7	
DIK_K	0x25		DIK_RMENU	0xB8	Right Alt
DIK_L	0x26		DIK_PAUSE	0xC5	

PC Scan Code	Value	Comment
DIK_SEMICOLON	0x27	
DIK_APOSTROPHE	0x28	
DIK_GRAVE	0x29	accent grave
DIK_LSHIFT	0x2A	
DIK_BACKSLASH	0x2B	
DIK_Z	0x2C	
DIK_X	0x2D	
DIK_C	0x2E	
DIK_V	0x2F	
DIK_B	0x30	
DIK_N	0x31	
DIK_M	0x32	
DIK_COMMA	0x33	
DIK_PERIOD	0x34	. on main keyboard
DIK_SLASH	0x35	/ on main keyboard
DIK_RSHIFT	0x36	
DIK_MULTIPLY	0x37	* on numeric keypad
DIK_LMENU	0x38	left Alt
DIK_SPACE	0x39	
DIK_CAPITAL	0x3A	
DIK_F1	0x3B	
DIK_F2	0x3C	
DIK_F3	0x3D	
DIK_F4	0x3E	
DIK_F5	0x3F	
DIK_F6	0x40	
DIK_F7	0x41	
DIK_F8	0x42	
DIK_F9	0x43	
DIK_F10	0x44	
DIK_NUMLOCK	0x45	
DIK_SCROLL	0x46	Scroll Lock
DIK_NUMPAD7	0x47	
DIK_NUMPAD8	0x48	
DIK_NUMPAD9	0x49	
DIK_SUBTRACT	0x4A	- on numeric keypad
DIK_NUMPAD4	0x4B	
DIK_NUMPAD5	0x4C	
DIK_NUMPAD6	0x4D	
DIK_ADD	0x4E	+ on numeric keypad
DIK_NUMPAD1	0x4F	
DIK_NUMPAD2	0x50	
DIK_NUMPAD3	0x51	

PC Scan Code	Value	Comment
DIK_HOME	0xC7	Home on arrow keypad
DIK_UP	0xC8	UpArrow on arrow keypad
DIK_PRIOR	0xC9	PgUp on arrow keypad
DIK_LEFT	0xCB	LeftArrow on arrow keypad
DIK_RIGHT	0xCD	RightArrow on arrow keypa
DIK_END	0xCF	End on arrow keypad
DIK_DOWN	0xD0	DownArrow on arrow keypad
DIK_NEXT	0xD1	PgDn on arrow keypad
DIK_INSERT	0xD2	Insert on arrow keypad
DIK_DELETE	0xD3	Delete on arrow keypad
DIK_LWIN	0xDB	Left Windows key
DIK_RWIN	0xDC	Right Windows key
DIK_APPS	0xDD	AppMenu key
DIK_POWER	0xDE	System Power
DIK_SLEEP	0xDF	System Sleep
DIK_WAKE	0xE3	System Wake
DIK_WEBSEARCH	0xE5	Web Search
DIK_WEBFAVORITES	0xE6	Web Favorites
DIK_WEBREFRESH	0xE7	Web Refresh
DIK_WEBSTOP	0xE8	Web Stop
DIK_WEBFORWARD	0xE9	Web Forward
DIK_WEBBACK	0xEA	Web Back
DIK_MYCOMPUTER	0xEB	My Computer
DIK_MAIL	0xEC	Mail
DIK_MEDIASELECT	0xED	Media Select

// Alternate scancodes used for transition from DOS.

DIK_BACKSPACE	DIK_BACK	backspace
DIK_NUMPADSTAR	DIK_MULTIPLY	* on numpad
DIK_LALT	DIK_LMENU	left Alt
DIK_CAPSLOCK	DIK_CAPITAL	CapsLock
DIK_NUMPADMINUS	DIK_SUBTRACT	- on num pad
DIK_NUMPADPLUS	DIK_ADD	+ on num pad
DIK_NUMPADPERIOD	DIK_DECIMAL	. on num pad
DIK_NUMPADSLASH	DIK_DIVIDE	/ on num pad
DIK_RALT	DIK_RMENU	right Alt
DIK_PGDN	DIK_NEXT	PgDn
DIK_PGUP	DIK_PRIOR	PgUp
DIK_UPARROW	DIK_UP	Up on arrow pad
DIK_LEFTARROW	DIK_LEFT	LeftA on arrow pad
DIK_RIGHTARROW	DIK_RIGHT	Right on arrow pad
DIK_DOWNARROW	DIK_DOWN	Down on arrow pad

## Default Vcc key mapping

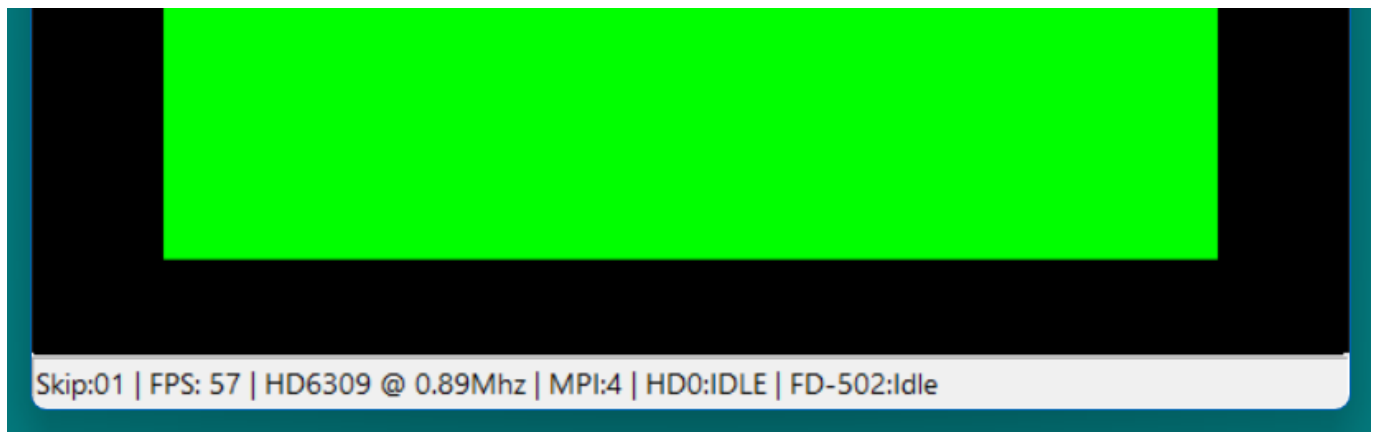
Modifiers: 0=none, 1=shift, 2=control 3=alt

Scancode	Mod	Coco Key	Mod	Comment	Scancode	Mod	Coco Key	Mod	Comment
DIK_A	0	COCO_A	0		DIK_5	1	COCO_5	1	%
DIK_B	0	COCO_B	0		DIK_6	1	COCO_7	2	^
DIK_C	0	COCO_C	0		DIK_7	1	COCO_6	1	&
DIK_D	0	COCO_D	0		DIK_8	1	COCO_COLON	1	*
DIK_E	0	COCO_E	0		DIK_9	1	COCO_8	1	(
DIK_F	0	COCO_F	0		DIK_0	1	COCO_9	1	)
DIK_G	0	COCO_G	0		DIK_SEMICOLON	0	COCO_SEMICOLON	0	;
DIK_H	0	COCO_H	0		DIK_SEMICOLON	1	COCO_COLON	0	:
DIK_I	0	COCO_I	0		DIK_APOSTROPHE	0	COCO_7	1	'
DIK_J	0	COCO_J	0		DIK_APOSTROPHE	1	COCO_2	1	"
DIK_K	0	COCO_K	0		DIK_COMMA	0	COCO_COMMA	0	,
DIK_L	0	COCO_L	0		DIK_PERIOD	0	COCO_PERIOD	0	.
DIK_M	0	COCO_M	0		DIK_SLASH	1	COCO_SLASH	1	?
DIK_N	0	COCO_N	0		DIK_SLASH	0	COCO_SLASH	0	/
DIK_O	0	COCO_O	0		DIK_EQUALS	1	COCO_SEMICOLON	1	plus
DIK_P	0	COCO_P	0		DIK_EQUALS	0	COCO_MINUS	1	equals
DIK_Q	0	COCO_Q	0		DIK_MINUS	0	COCO_MINUS	0	-
DIK_R	0	COCO_R	0		DIK_UPARROW	0	COCO_UP	0	
DIK_S	0	COCO_S	0		DIK_DOWNARROW	0	COCO_DOWN	0	
DIK_T	0	COCO_T	0		DIK_LEFTARROW	0	COCO_LEFT	0	
DIK_U	0	COCO_U	0		DIK_RIGHTARROW	0	COCO_RIGHT	0	
DIK_V	0	COCO_V	0		DIK_NUMPAD8	0	COCO_UP	0	
DIK_W	0	COCO_W	0		DIK_NUMPAD2	0	COCO_DOWN	0	
DIK_X	0	COCO_X	0		DIK_NUMPAD4	0	COCO_LEFT	0	
DIK_Y	0	COCO_Y	0		DIK_NUMPAD6	0	COCO_RIGHT	0	
DIK_Z	0	COCO_Z	0		DIK_SPACE	0	COCO_SPACE	0	
DIK_0	0	COCO_0	0		DIK_RETURN	0	COCO_ENTER	0	
DIK_1	0	COCO_1	0		DIK_NUMPAD0	0	COCO_CLEAR	0	
DIK_2	0	COCO_2	0		DIK_ESCAPE	0	COCO_BREAK	0	
DIK_3	0	COCO_3	0		DIK_NUMPADPERIOD	0	COCO_BREAK	0	
DIK_4	0	COCO_4	0		DIK_F1	0	COCO_F1	0	
DIK_5	0	COCO_5	0		DIK_F2	0	COCO_F2	0	
DIK_6	0	COCO_6	0		DIK_BACK	0	COCO_LEFT	0	
DIK_7	0	COCO_7	0		DIK_LBRACKET	0	COCO_DOWN	1	[
DIK_8	0	COCO_8	0		DIK_RBRACKET	0	COCO_RIGHT	1	]
DIK_9	0	COCO_9	0		DIK_BACKSLASH	0	COCO_CLEAR	1	\
DIK_1	1	COCO_1	1	!	DIK_CAPSLOCK	0	COCO_0	1	DECB Tog case
DIK_2	1	COCO_AT	0	@	DIK_LSHIFT	0	COCO_SHIFT	0	
DIK_3	1	COCO_3	1	#	DIK_LCONTROL	0	COCO_CTRL	0	
DIK_4	1	COCO_4	1	\$	DIK_LMENU	0	COCO_ALT	0	



## The Status Line

---



This is a view of the VCC screen showing the "status line". The status line contains useful information about the status of the emulation.

- Current frame skip setting. It means draw every nth frame. So 1 is every frame , 2 is every other frame etc.
- Average frames per second. Should always read 60. if it consistently reads lower try selecting a higher frame skip or turning on scan lines. See the configuration dialogs section.
- CPU type currently being emulated, MC6809 or HD6309, and clock speed. .89 and 1.79 MHz are stock. Over-clocking up to 89 MHz is supported.
- MPI cart number selected
- Cartridge data fields. Pluggable cart dlls can also display information on the status line:
  - Hard Drive info is displayed if the "harddisk.dll" cart is loaded
  - FD-502 info is displayed if the "fd502.dll" is loaded
  - DriveWire4 info field. Displays "ConOK" if there is a DW4 connection. The "R" & "W" fields display data as DW4 is accessed
  - Acia status is displayed if the "acia.dll" is loaded

## Configuring VCC

---

This section will try to explain the options to configure VCC for normal use.

Setting up VCC should be as easy as double clicking the VCC setup file icon and following the onscreen prompts. We have tried to make the *default* as close to a "stock" Tandy Color Computer 3, just as you would have bought it in 1986.

### VCC Defaults

Here are the default settings for VCC.

- Audio - Primary Sound Driver. This should default to whatever your PC is using.
- CPU/CPU - Motorola MC6809
- CPU/Memory - 128k (up to 8 meg).
- CPU/Overclocking - 1.788 MHz (stock Coco 3 speed). VCC will actually start at .89MHz with 1.788MHz available through software POKEs.
- CPU/AutoStart Emulation - Checked
- CPU/AutoStart Cartridge - Checked
- Display/Monitor Type - RGB
- Display/Frame Skip - 1
- Display/Scan Lines - Unchecked
- Display/Allow Resize - Checked
- Display/Throttle Speed - Checked
- Keyboard/Keyboard Mapping - Coco (DECB), emulates the Coco3 keyboard
- Joysticks/Left Joystick - Mouse
- Joysticks/Right Joystick - Mouse
- BitBanger/Add LF to CR - Checked
- BitBanger/Print Monitor Window - Unchecked
- Cartridge - Empty

You can change these settings to suit your needs. The CPU overclocking is particularly handy when running an assembler or compiling "C" code in OS-9. Overclocking also speeds up screen scrolling in text editors, but it is not advised to use overclocking while running games. It may speed up the emulator enough to make the game unplayable.

## Setting Up A "Custom" VCC to Suit Your Needs

---

The emulator can be configured many ways and emulates some of the more unique Color Computer 3 setups.

VCC saves most settings you make in the Vcc.ini.

See the VCC Quick Start guide for common scenarios.

## Using the MultiPak Interface

Included with VCC is an emulation of the Tandy MultiPak Interface (MPI). To use the MPI, Click the "Cartridge" menu selection at the top of the emulator window. Select "Cartridge" (actually,



the only choice initially). Select "Load Cart". If the navigation panel does not start in your VCC installation folder, you must navigate to this folder, usually "C:\Program Files (x86)\VCC xxxx" (or wherever you installed it, "xxxx" being the version number). Select the "mpi.dll" and click "Open". On some more *modern* versions of Windows (Vista, 7, 8 & 10), the ".dll" extension may be hidden from view. If this is the case, just select "mpi".

Once you have loaded the MPI, you will notice now that the "Cartridge" menu has expanded. You will see 4 "Cartridge Slots" and an "MPI Config" selection. These "slots" are where you insert the custom "ROM carts" included with the VCC installation or most any of the "ROM carts" you can find and download from any of the Color Computer archive websites. To use a ROM cart, you must *select* the proper slot in the "MPI Config". Just move the "Slot Select" slider to the slot in which you have inserted the cart.

The Color Computer came with two kinds of cartridges. Most only contained rom images of programs that the Coco would run. This was typical of game and utility carts. VCC expects these to be stored as .ROM files. The other kind of cartridge contained peripheral hardware in addition to rom. VCC requires these to be custom .DLL files written specifically for it.

VCC comes with several of these DLL carts:

- FD-502 Disk Controller (fd502.dll) - Standard Tandy FD-502 Disk controller.
- Hard Drive Controller (harddisk.dll) - An emulation of the a typical MFM HD controller.
- SuperIDE Hard Drive Controller (SuperIDE.dll) - An emulation of Cloud9's "SuperIDE" HD w/CF cards.
- Becker Port Cart - A custom designed cart of the "Becker Port" for use with DriveWire4.
- Orchestra90cc (orch90.dll) - An emulation of the Tandy Orchestra90cc cart.
- RamDisk (ramdisk.dll) - An emulation of a 512k RAM card similar to those made by Disto.
- Game Master Cart (GMC.dll) - An emulation of John Linville's "Game Master Cart"
- Rs232 Pak (acia.dll) - An emulation of RS Deluxe RS232 cart
- SDC (sdc.dll) - A simulation of Darren Atkinson's SDC floppy drive emulator

Several of these carts require an external ROM image to run (included). The ROMs should *autoload* when the modules are inserted into the MPI slot. These carts & ROMs are:

- fd502.dll - disk11.rom, rgbdos.rom, or one of the hdbdos ROMs (explained in the "Loadable Modules" section of this manual).
- orch90.dll - orch90.rom
- GMC.dll - cyd\_gmc.rom (optional, other GMC compatible ROMs may be loaded manually as well)
- acia.dll - rs232.rom

- sdc.dll - SDC-DOS.rom

The "becker.dll", "harddisk.dll", and "SuperIDE.dll" are dependent on whatever ROM image is loaded by the "fd502.dll". This is handled in the "FD502 Config" menu under the "Cartridge" menu.

The "hdbdos" and "rgbdos" ROM uses are best explained in the "HDBDOS User's Manual" available on the Cloud9 website and the "RGB-DOS User's Guide" found in "The Color Computer Archives".

The ROMs included in this installation are specifically written for use with emulators.

## Loadable Modules

As stated before, the VCC emulator alone does not know anything about the various peripherals that were available. It depends entirely on runtime loadable DLL files (Modules) to emulate them. These are windows programs that are used to emulate hardware external to the Coco. After a module is loaded it will, if needed, add its own options to the Cartridge Menu and status line. There are currently 9 modules included. They are: "mpi.dll", "orch90.dll", "harddisk.dll", "Superide.dll", "fd502.dll", "becker.dll", "acia.dll", "GMC.dll", "sdc.dll", and "ramdisk.dll". DLL Modules are loaded via the Cartridge menu option.

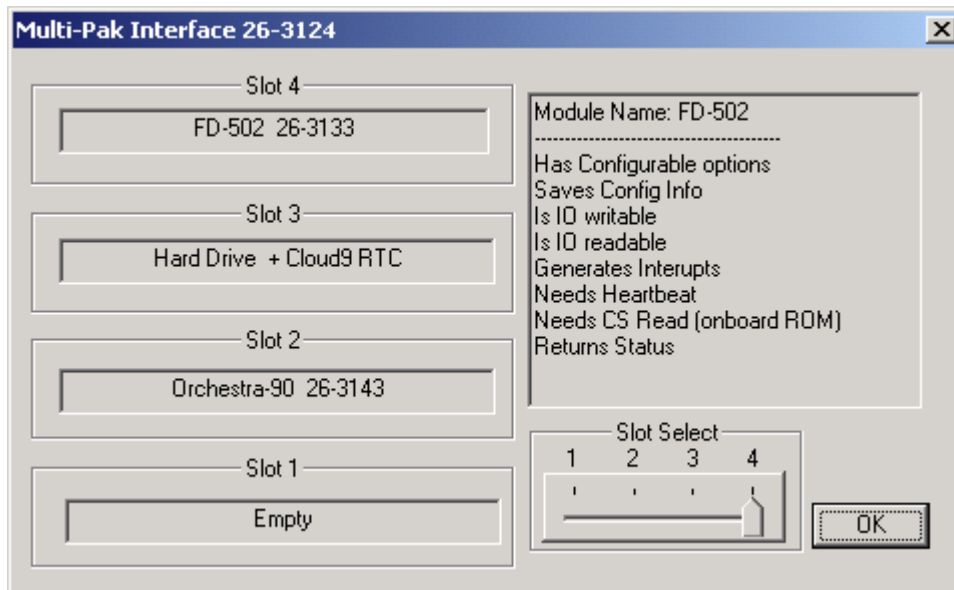
### orch90.dll

This is the simplest module. It emulates the Orchestra-90 program pack. It has no menu options and returns no status. The sound from the Orchestra-90 pak will play in full stereo through your PC's speakers. To use the Orchestra-90 as the original, you must have the Orchestra-90 rom in the VCC installation folder along with the other roms. The "Autostart Cart" checkbox must be checked in the "Config/Cpu" menu and if using the MPI, the MPI selector switch must be set to the same slot as the Orch90 pak. Hit "F5" to reset the emulator and start the cart.

### mpi.dll

The MPI module emulates the standard Tandy Multi-Pack interface. It adds 5 menu options. The first four allow Inserting/Ejecting carts from the Multi-pack. Any loaded modules will also add their own options to the menu. As with the main emulator, each slot will accept either a Module DLL or a Program ROM pack. Don't try to insert the mpi.dll module into an MPI slot, this can not work. Also don't try to insert the same module into more than one slot.

The MPI Config option is mostly informational. The Slot Select Slider picks the slot the MPI will use on startup. Displayed on the left are the names of all loaded modules. On the right is a list of API interfaces the currently selected module needs.



This module will add an MPI item to the status line. Following this are two numbers. The first is the slot the Chip select signal is routed to, The second is the destination of the Spare Select line. Following this is any status info returned from loaded modules. In the example above the MPI is set for slot 4 (internally slots are numbered 0 to 3) and the harddisk and fd502 modules are also loaded.

### **fd502.dll**

This module emulates the Tandy FD-502 Floppy disk controller with 4 Double Sided/ Double density disk drives attached. It adds 5 options to the Menu. The first 4 are simply to Insert / Eject Virtual disk (DSK) images. Normally the fd502.dll will autoload the disk11.rom file which contains Disk Extended Color Basic (DECB) which is the coco extension required to read floppy disks. Usage details can be found in the Disks Section of this manual.

### **harddisk.dll**

This module implements the use of emulator disk files that are supported in many Coco emulators. It adds a dual menu item used to Insert/Eject up to two virtual hard disk images (VHD) as HD 0 and HD 1. It also contains an implementation of the Dallas DS1315 real time clock as used by Cloud-9. This can be used under OS9/Nitros9 with the appropriate driver. Usage details can be found in the Disks Section of this manual.

### **SuperIDE.dll**

This module emulates Glenside IDE disk images. It adds a menu to insert / eject a master and a slave IDE. It also contains an implementation of the Dallas DS1315 real time clock. Usage details can be found in the Disks Section of this manual.

## **becker.dll**

The "Becker Port" cart emulation has been added to allow the user to utilize the features of the DriveWire4 file server on your PC. DriveWire4 can be used for multiple dsk/vhd access, TelNet, modem emulation (through telnet), internet FTP access, DriveWire MIDI through the PC host's soundcard, virtual printing, and much, much more! Usage details can be found in the Becker Port Section of this manual.

## **acia.dll**

Acia.dll is a Vcc add-on that attempts to emulate the Deluxe RS232 program pack. It allows connection to windows serial ports and a tcpip server. It also allows reading/writing windows files and interaction with the Vcc console. Operation of the Acia rs232 pak is detailed in the "Acia Guide"

## **ramdisk.dll**

This cart image emulates a 512k RAM cart. The cart uses a 24 bit addressing mode (3 byte) to read/write a single byte of data to/from the cart. To set the address in the cart of the data poke the high byte into FF42, the middle byte into FF41, and the low byte into FF40. To read peek FF43, to write poke FF43. A NitrOS9 driver for this cart would be a simple project.

NOTE: This cart has port conflicts with fd502. It will only work if fd502.dll is NOT installed.

## **GMC.dll**

NOTE: The GMC.dll ROM cart is experimental and has not been fully tested. We welcome anyone with compatible ROMS or software that uses this ROM cart to test this feature and let us know if it is working correctly

**Any information for this ROM cart must be obtained from elsewhere as we could find very little pertaining to it's use.**

Please refer to John Linville (or others) for information on using this cart. There seems to be no documentation and/or very little information concerning it's use.

We have provided a sample ROM image, "cyd\_gmc.rom" in VCC's installation directory. Load the GMC.dll into the cartridge slot and in the cartridge menu, click "Select GMC ROM". Navigate to the VCC installation directory and select "cyd\_gmc.rom". Once selected, press F9 twice and the demo will play music.

All that is currently know about the GMC is that it is an emulation of the "TI SN76489 Digital Complex Sound Generator" chip using the Coco port \$FF41. The single port is "write only". No data can be read from this port. All programming for the GMC sound is done through this port.

Additional information on the GMC, and some found from the "CoCopedia" WIKI site is the document "Game Master Cart Info" included in this distribution.

## **sdcard.dll**

This cart was introduced with VCC version 2.1.9.2 and simulates Darren Atkinson's SDC floppy drive emulator. The simulator is a work in progress and some sdc features have been left out. Most of these are features used to create and manage sdcard files. These functions are necessary on a real Coco3 but are more conveniently managed using the Windows OS that Vcc runs on. Some implemented features may not work properly, these are usually not intentional but are due to a mis-understanding of how a read SDC works.

SDC requires changes to mpi.dll to work properly with the floppy driver (fd502) so make sure you have the latest version of mpi.dll if you intend to also use the floppy driver. See the SDC\_Guide for usage details.

## **System and DLL ROMs**

These are ROM images that contain 6x09 programs that are required by the emulated Coco or hardware emulation DLLs. They must be installed in VCC's home directory, which is "C:\Program Files\VCC x.x.x.x" by default.

This installation of VCC comes with the following ROM images installed in the VCC installation folder along with the emulator and it's supporting software:

- **"coco3.rom"** A combined copy of the original Tandy "Color BASIC", "Extended Color BASIC", and "Super Extended Color BASIC" ROMs installed in the Color Computer 3. This ROM was compiled from the "ToolShed" repository sources and not "ripped" from a real Color Computer 3. Vcc will not start unless this coco3.rom is found in the same directory as vcc.exe
- **"disk11.rom"** A copy of the "Disk Extended Color BASIC" ROM (v1.1) supplied with the Tandy "FD-502" disk controller. This ROM was compiled from the "ToolShed" repository sources and not "ripped" from a real FD-502 disk controller.
- **"rgbdos.rom"** A copy of the RGBDOS ROM by Robert Gault and downloaded from his website. This ROM has "offsets" for dual partitioned VHD images with NitroS9 and RSDOS partitions. The offsets are set at "\$5A000" for *standard* VHD images that use a 90 megabyte OS-9 partition and a RSDOS partition with 256 virtual drives. To change this ROM for use with *single partition* VHD images, (in RSDOS) use:  
POKE&HD938,0:POKE&HD939,0:POKE&HD93A,0

Or in a Windows hex editor, you can change the values permanently by changing the ROM image values directly at locations 1938,1939, and 193A, *allto00*.

- **"hdbdw3bc3.rom"** Standard "Becker Port" HDBDOS ROM. Compiled from the HDBDOS sources at the ToolShed repository. Not to be used with *dual partitioned* VHD images.
- **"hdbdw3bc3 w-offset 5A000.rom"** Standard "Becker Port" HDBDOS ROM. Compiled from the HDBDOS sources at the ToolShed repository, but modified for using dual partitioned VHD images for OS-9 and RSDOS. The offsets are set at "\$5A000" for *standard* VHD images that use a 90 megabyte OS-9 partition and and RSDOS partition with 256 virtual drives.
- **"hdbdw3bck.rom"** "Becker Port" HDBDOS ROM without the *speedup* (2 MHz) poke installed. This ROM allows time critical Games and Apps that do not run well (or run too fast) with the speedup poke. Compiled from the HDBDOS sources at the ToolShed repository. Not to be used with *dual partitioned* VHD images.
- **"hdbdw3bck w-offset 5A000.rom"** "Becker Port" HDBDOS ROM without the *speedup* (2 MHz) poke installed. This ROM allows time critical Games and Apps that do not run well (or run too fast) with the speedup poke.
- **"rs232.rom"** The rs232.rom is a copy of the rom from the Radio Shack Deluxe RS232 program Pack. It is loaded by acia.dll.
- **"SDC-DOS.rom"** The SDC-DOS.rom is a copy of SDC-DOS version 1.75. It is loaded by sdc.dll.

The RGBDOS and HDBDOS ROMs offsets can be changed to match any size VHD images with any size partitions using the methods used above for the RGBDOS ROM. The same ROM addresses apply to HDBDOS. To calculate the proper offset for your VHD partition, We suggest using Robert Gault's "SPECS.BAS" program found on "RGBDOS Tools Disk image" found on his website (listed above). This program will calculate the size of your OS-9 partition (which must come first) and give you the proper offset to the RSDOS partition. The SPECS.BAS program can be found on the "Tools.dsk" in the RGBDOS zip available for download on his site.

NOTE:

The ROM images are provided for your convenience and to assure the proper ROMs for VCC are present.

The VCC Development Team is not responsible for the ROM contents as they are each developed by 3rd party developers, though verified to be accurate to the original ROMs of the same name. These ROM images are the same as used by most Color Computer 3 emulations.

Also note that the "hdblba.rom" used for SuperIDE on DECB that was distributed with some previous versions of VCC was found to be defective and has been replaced. Be sure to use the latest version.

## The VCC Command Line Options

---

VCC allows a few command line options. One has been available since at least VCC 1.42. It is expected more options will be added.

*NOTE: Command line options are used in the Windows Command Prompt or in Shortcut targets. You must change directory to the location of the VCC installation directory or use the complete file path to "VCC.exe" before using any command line options.*

**Quick File Load** Certain files can be automatically loaded and started when VCC starts by specifying them on the command line. The files are identified by their extensions; .CCC, .ROM, or .BIN

.CCC and .ROM files are considered to be images from Color Computer Cartridges. They are loaded and will auto start if "AutoStart is checked in Vcc Cpu Options.

.BIN files are in the format DECB uses to save binary files using "SAVEM". If you have a .dsk file containing these files you can use a coco .dsk utility such as "decb" from the "toolshed" package to copy them to windows.

To use the quick file load function you can use the command line as follows:

```
C> vcc filename.bin
```

Alternatly you can simply drag and drop one of these files onto a Vcc Shortcut on your desktop to run vcc with the binary loaded. Or you double click on any of these file types in Windows File Explorer (if already associated with another program, you may need to right click, select "Open With", then associate the extension to VCC.exe), then associate these extensions with VCC, they will open VCC and run, from any directory. Caution that if you associated these file types with a previous version of Vcc you may have de-associate them first. Windows does not include full paths in program associations.

**Auto Paste Text** - You can specify a basic command that will be auto pasted into DECB when VCC starts using the -p option followed by the command or commands seperated by colons. If there are spaces in the command string it must be surrounded by double quotes. However double quotes are allowed within the quoted string, as shown in the second example:



To auto run the dos command:

```
C> vcc -pDOS
```

Auto load a bin file from disk and exec it:

```
C> vcc -p "drive 3: loadm "sig": exec"
```

**Custom VCC.ini Load** - When VCC starts normally it loads and stores configuration settings in the initialization file "%APPDATA%\VCC\Vcc.ini". You can cause VCC to use an alternate file from the command line with the '-i' option. You should first have created such a file from manually editing a vcc.ini file and saving it under a different name (w/the ini extension), or having "Saved" the custom ini file from the "File/Save Config" feature (below). To load a custom ini from the cmd line:

```
C> vcc -i IniFileName.ini
```

You can also set your Windows shortcut to VCC to load the custom config file on startup. To do so you just: Right click on the VCC shortcut

In the "General" tab in the "Target" list, type (include quotes)

```
"C:\Program Files (x86)\VCCxxx\VCC.exe" -i "%appdata%\VCC\Custom.ini"
```

Now when you click your VCC shortcut, you will start with your custom config.

In addition to selecting the .ini file other command line options can be added to a shortcut to autoload a binary or to paste a basic command string. You can even create one multiple click shortcuts for each of your favorite VCC configurations.

## Loading and Saving Custom VCC.ini Files

You can also Load/Save custom ini files now from the "File" menu while VCC is running. To save a custom ini file, just click "File/Save Config", type in a new custom.ini name, and click Save.

To load a custom ini file (you must have created one first), Click "File/Load Config" , click the custom.ini file you want to load, then click Load. For your new configuration to boot, you'll need to hit F9 twice.

NOTE: Before creating a "custom" version of VCC, We suggest you "Save" the current configuration under the new custom name first. This is because VCC writes many of the changes to an ini file as you go, so any change you make will be saved under the current name. VCC starts under the "vcc.ini" file, so we suggest leaving that file "plain", saving under a custom



name, then make your changes. Most changes will be save to the "new" file, but don't forget to "Save Config" when you're done just to make sure. When you exit VCC, the current name is used to save the final state of your configuration.

## The VCC.ini File Format

The "vcc.ini" and any custom "ini" file you may save, will be stored in you "user/appdata/roaming/vcc" folder. The ini files are kept in a strict format and users must use caution if editing these files manually or you may ruin your VCC configuration and have to redo it from inside VCC.

The following describes the ini file fields

### [Version]

Release=x (VCC version)

### [CPU]

DoubleSpeedClock=x (speed value 0-255)

FrameSkip=x (1-6)

Throttle=x (x=0-1, 0=off, 1=on)

CpuType=x (x=0-1, 0=MC6809, 1=HD6309)

MaxOverClock=x (1-256)

### [Audio] SndCard=x (SndCardName)

Rate=x (x=0-3, 0=none, 1=11,025khz, 2=22,050khz, 3=44,100khz)

### [Video]

MonitorType=x (x=0-1, 0=RGB, 1=CMP) Scanlines=x (x=0-1, 0=off, 1=on)

AllowResize=x (x=0-1, 0=off, 1=on) ForceAspect=x (x=0-1, 0=off, 1=on)

PaletteType=0

RememberSize=1 WindowSizeX=1280 WindowSizeY=990

### [Memory]

RamSize=x (0=128k, 1=512k, 2=2048k, 3=8192k)

ExternalBasicImage=x (FileDir\FileName) (not normally used)

### [Misc]

AutoStart=1

CartAutoStart=1

KeyMapIndex=3

CustomKeyMapFile=C:\Users\BPier\AppData\Roaming\VCC\OS9.keymap

ShowMousePointer=0

[Module] OnBoot=FilePath\ROM\CCC\DLL\PAKFilename

[LeftJoyStick]

UseMouse=x (x=0-1, 0=off, 1=on)

Left=x (x=keycode)

Right=x (x=keycode)

Up=x (x=keycode)

Down=x (x=keycode)

Fire1=x (x=keycode)

Fire2=x (x=keycode)

DiDevice=x (x=DeviceName)

HiRezDevis=x (0-2, 0=none, 1=RS HiRez, 2=CocoMax)

[RightJoyStick]

UseMouse=x (x=0-1, 0=off, 1=on)

Left=x (x=keycode)

Right=x (x=keycode)

Up=x (x=keycode)

Down=x (x=keycode)

Fire1=x (x=keycode)

Fire2=x (x=keycode)

DiDevice=x (x=DeviceName)

HiRezDevis=x (0-2, 0=none, 1=RS HiRez, 2=CocoMax)

[MPI]

SWPosition=x (x=0-3 slot number) SLOT1=PakFilePath\PakFileName

SLOT2=PakFilePath\PakFileName SLOT3=PakFilePath\PakFileName

SLOT4=PakFilePath\PakFileName

[Hard Drive] VHDImage=VHDFilePath\VHDFileName

VHDImage1=VHDFilePath\VHDFileName

[FD-502]

DiskRom=x (x=0-2, 0=ExternalRom, 1=RSDOSRom, 2=RGBDOSRom)

RomPath=DiskRomPath

Persist=x (x=0-1, 0=off, 1=on)

Disk#0=DiskFileName1

Disk#1=DiskFileName2

Disk#2=DiskFileName3

Disk#3=DiskFileName4

ClkEnable=x (x=0-1, 0=off, 1=on) TurboDisk=x (x=0-1, 0=off, 1=on)

[DefaultPaths] FloppyPath=FloppyPathName

HardDiskPath=HardDiskPathName MPIPath=MPIPathName  
CassPath=CassPathName PakPath=PakPathName  
SuperIDEPATH=HardDrive Pathname

[HDBDOS/DW/Becker]

DWServerAddr=x.x.x.x (x=Server IP addresss i.e. 127.0.0.1)

DWServerPort=x (x=server port # i.e 65504)

[Glenside-IDE /w Clock] Master=IDEMasterImageName

Slave=IDESlavelImageName

BaseAddr=x (x=0-3, 0=40, 1=50, 2=60, 3=70)

ClkEnable=x (x=0-1, 0=off, 1=on) ClkRdOnly= (x=0-1, 0=off, 1=on)

[GMC-SN74689]

ROM=ROM Image Path/Name

[Acia]

AciaBasePort=104

AciaComType=0

AciaComMode=0

AciaComPort=COM3

AciaTcpPort=48000

AciaTcpHost=localhost

AciaTextMode=0

AciaFileRdPath=AciaFile.txt

AciaFileWrPath=AciaFile.txt

## Disks

---

### Virtual Disks

---

To access virtual disks the fd502.dll module must be installed. It is traditional to install this module in MPI slot 4 and select it to get DECB started.

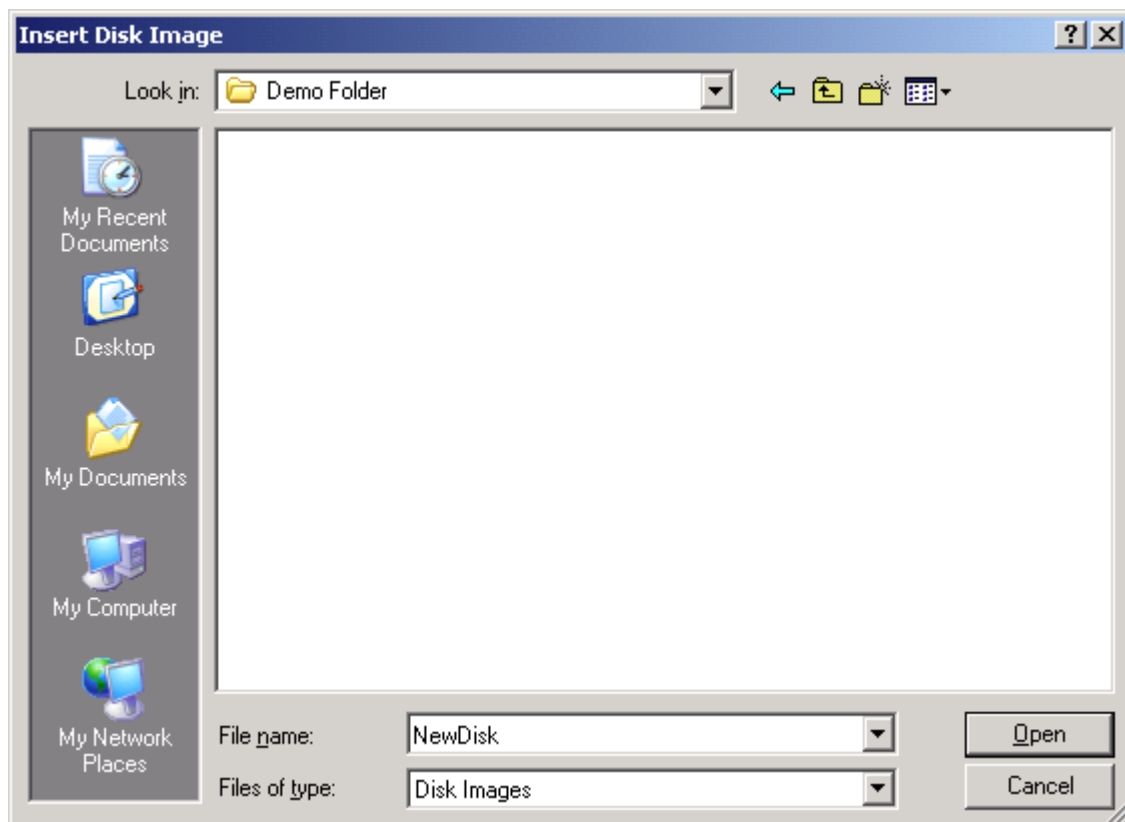
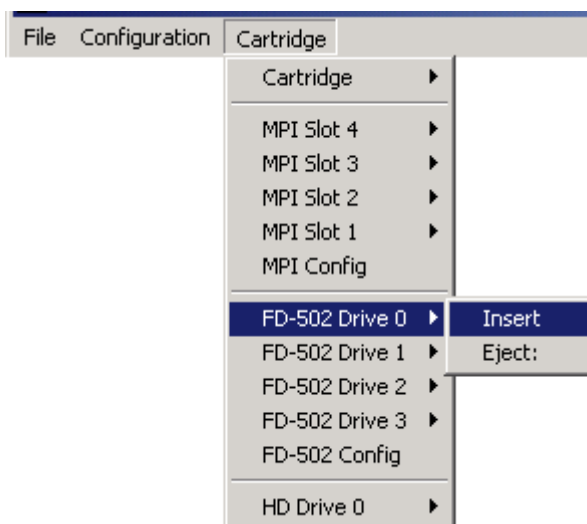
### Floppy Images ".dsk" files

Most image types are supported including DMK, OS9, JVC, DSK and VDK. To insert a virtual Disk simply select the drive you wish to put it in and select insert. Note: Only the first side of Drive 3 can be accessed.

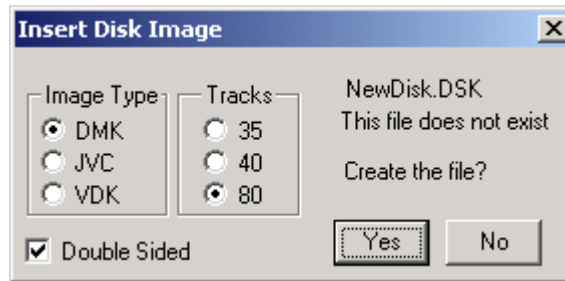
If you wish to have the disk write protected simply set the "read only" attribute from windows. This can be done by right-clicking the file, selecting Properties and checking the "read-only" box under Attributes. This will work for all image types. The DMK format uses a byte in the header to indicate "write protect". This will be respected if present but there is currently no way to change it from within this emulator.

### ***Creating a New Blank Disk:***

Creating a new blank disk will be similar to inserting an existing image. Simple click Insert, and instead of picking an existing image type the non-existing name of a new disk image you wish to create (with the extension ".dsk").



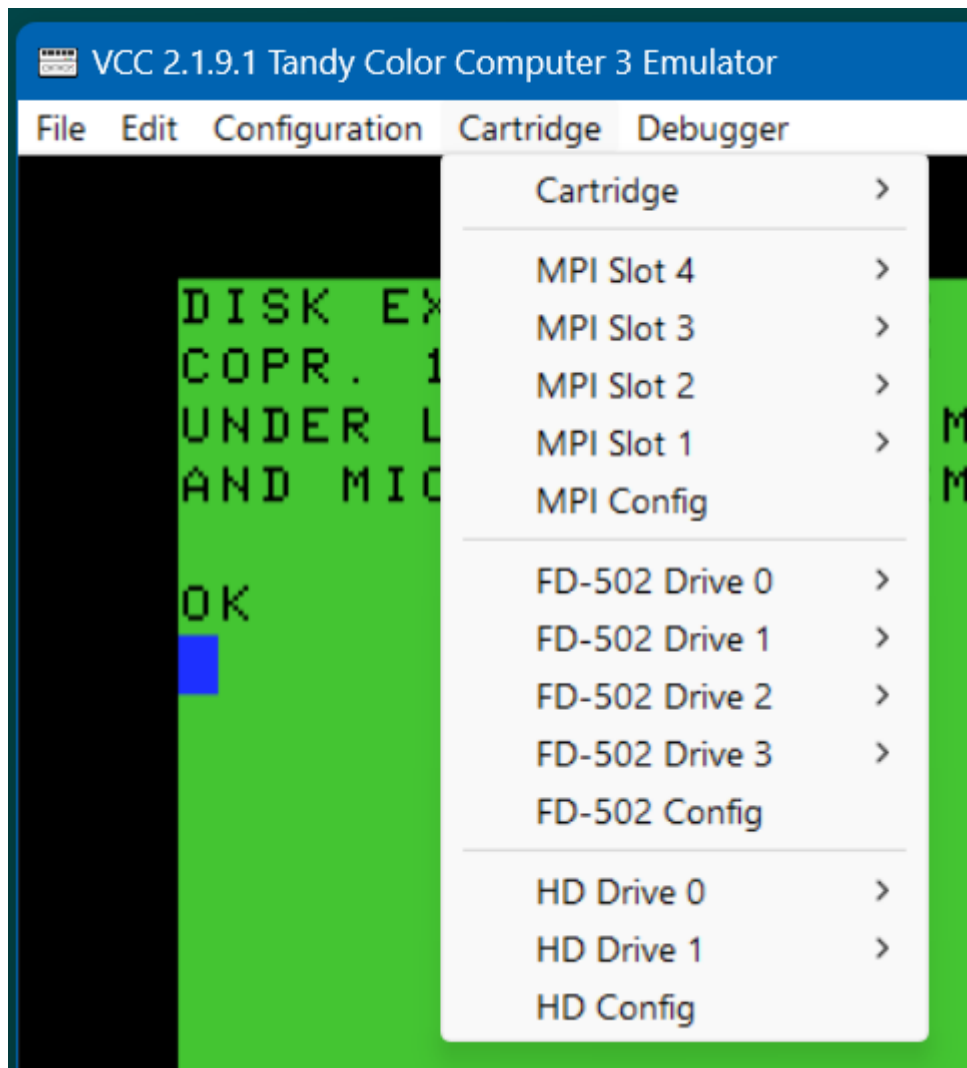
The following dialog will appear:



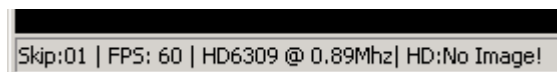
Just select the parameters for the new image and click yes. The image will be created, Zero filled and mounted in the drive you've selected. Note that unlike most other emulators the act of creating a new disk **will NOT format it**. Just as you would with a real disk you must either DSKINI (DOS) or format (OS9 et al) the image before it can be used. For most disk use, you will want to use the JVC as it is the format most emulators use. The DMK and VDK formats are not fully implemented and in some cases cause problems.

## Emulated hard disk images ".vhd" files

If haddisk.dll is installed in any MMI slot the Cartridge Menu will contain two entries for adding hard disks:



Also harddisk status will be added to the status bar and will show the following:



The status is interpreted as follows:

- HD:No Image! - Indicates that no VHD image has been selected.
- HDn:Idle - VHD image is loaded but not being accessed.
- HDn:Rd [sector] - VHD is being read.
- HDn:Wr [sector] - VHD is being written.

"n" is the drive 0 or 1 that is or last was accessed.

### ***Creating A New VHD Image***

Creating a new VHD image is much like creating a new floppy image above. In the "Insert" menu, type in a non-existent filename and the new dialog will appear.



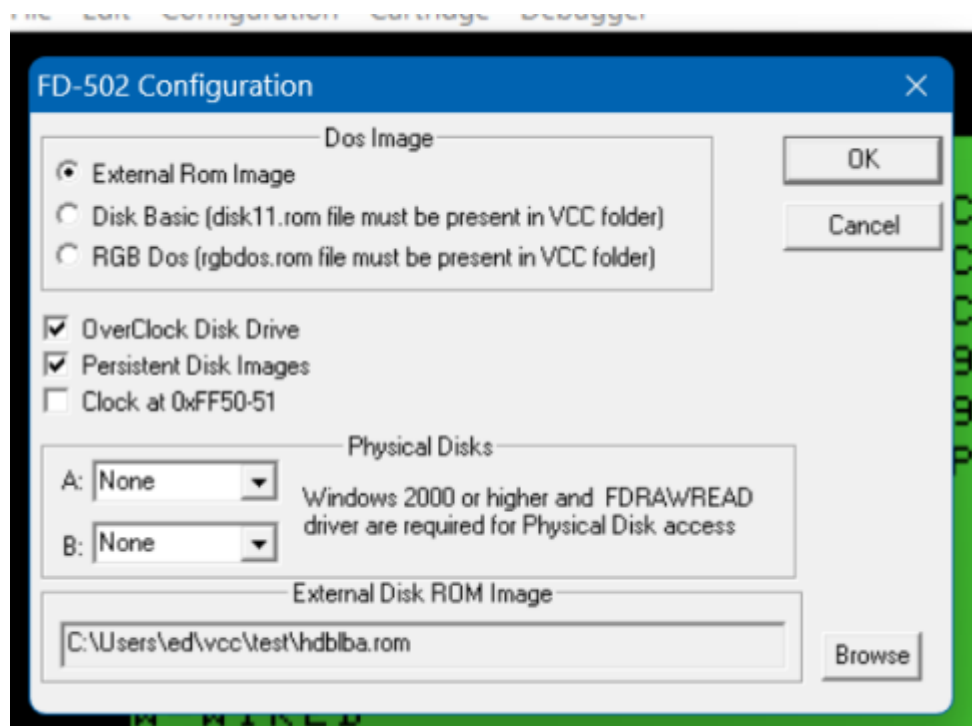
Input the size (in Kilobytes) that you want for your VHD.

The default size is 132,480k, which is the standard size for an HDBDOS/RGBDOS HD with 256 RSDOS disks and a (approximately) 92meg OS-9 HD, with HDBDOS/RGBDOS set at an offset of \$5A000 sectors. Refer to the HDBDOS manual for info in setting up this type of drive.

This size can be changed to any size you want, but remember to set your OS-9 HD driver to accomodate your VHD size.

You will have format this VHD before using it in RSDOS or NitroS9

## The FD502 Configuration Dialog



### Dos Image

There are two built in DOS ROMS to choose from. Disk Basic is the standard DOS 1.1 that shipped with the FD-502 controller. RGB Dos is a version of DOS modified by Robert Gault to take advantage of the virtual hard disk emulation described elsewhere in the document. External

Rom Image is just that. If you have a dump of a custom version of DOS (ADOS3 or HDB-DOS for example) simply click "Browse" and select it. then check the "External Rom Image" radio button. It is recommended that you keep this image in the same directory that VCC is installed to, but it's not required. To use the Becker Port cart and DW4, you will need to use this feature to mount the "hdbdw3bck.rom" image, and check "External Rom Image".

**OverClock Disk Drive**" Sounds more sophisticated than it is. By default Vcc attempts to emulate the time it would take a real disk to move the read/write head. Clicking this will reduce that time to almost nothing. If you are using the CPU over-clocking option and experience I/O errors try turning this on.

### Persistent Disk Images

When checked any Disk images mounted when the emulator is shut down will be remounted when it's started back up. Unchecking this will cause the emulator to start with empty Floppy disk drives. This is also true with any hard drive controller modules and VHD files.

**Clock at 0xFF50-51:** Selected whether the Disto RTC will be enabled or not.

### Physical Disks

**This is also experimental code. Please only use backup disks as we can't guarantee everything works 100% yet. Currently only "standard format" 18 sector per track disks are supported.**

To use this option 3 requirements must be met. The drop downs will be grayed otherwise.

- The host computer must be running Windows 2000,XP or Vista. Note this has only been tested on Windows XP.
- The host computer must have a supported Floppy disk controller chip, (uPD765a or equiv). A USB floppy drive will NOT do.
- The FDRAWCMD driver must be loaded. These drivers were written by Simon Owen and can be downloaded from his website here: [\[http://simonowen.com/fdrawcmd/\]](http://simonowen.com/fdrawcmd/)

Download and run the fdinstall.exe file.

There are two drop downs. One for each Physical disk drive you may have. Simply select the Virtual disk drive from the dropdown next to the Physical disk Letter. For Example: To Map Virtual disk 1 to real Drive A:, Select "Drive 1" from the dropdown next to A:.

To unmap a disk either select None from the dropdown or select Eject Floppy from the menu.

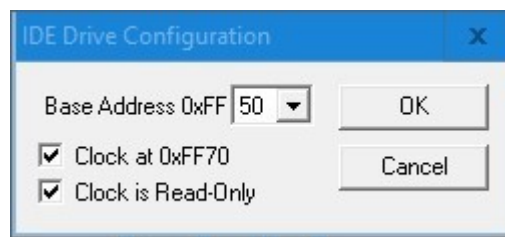
**DISCLAIMER: We have no association with either the Glenside computer club or Cloud-9. They do not endorse or support this program in any way. If you experience any issues with**



this software please **DO NOT** bother either of these entities. J.F.

## SuperIDE

This is simulation of the Glenside IDE and Cloud9's SuperIDE controller with the Dallas DS1315 RTC. It provides no more functionality than RGB-DOS and the hard disk module that are included with VCC. Its primary purpose is to allow owners of the Glenside IDE or Cloud-9 Super-IDE Controller to mount and run a backup of their CF card under VCC without any driver modifications. As such it is compatible the both HDB-DOS and the Super-Driver currently in the Toolshed and NitrOS9 repositories. It adds three menu items used to Insert/Eject backup images (.IMG files) created by SideKick Utility programs and to set the base address of the virtual controller. Note that currently the SideKick utility only allows dumping of removable CF cards, Hard disks are not currently supported for safety reasons. Sidekick and Sidekick Utilities for HDB-DOS can be found in the Color Computer Archive.



### Base Address

The base address must be set to match the address of the real hardware the CF card image was taken from.

Do not use 0xFF40 base address with the FD-502 module loaded as it uses part of this range as does the becker.dll.

The default base address is 0xFF50. This address conflicts with the RTC in FD502 so uncheck the Clock at 0xFF50-52 box in FD-502 Configuration if the default base address is used.

The 0xFF70 base address will conflict with the Cloud9 RTC used with this module, so the clock will be disabled if this address is used.

Note: the Hard Drive module also uses the Cloud9 RTC clock and some older versions of VCC have no means to disable it.

### Clock at 0xFF70

Enables the DS1315 RTC at this address. It will be unavailable if the IDE base address is set to 0xFF70.

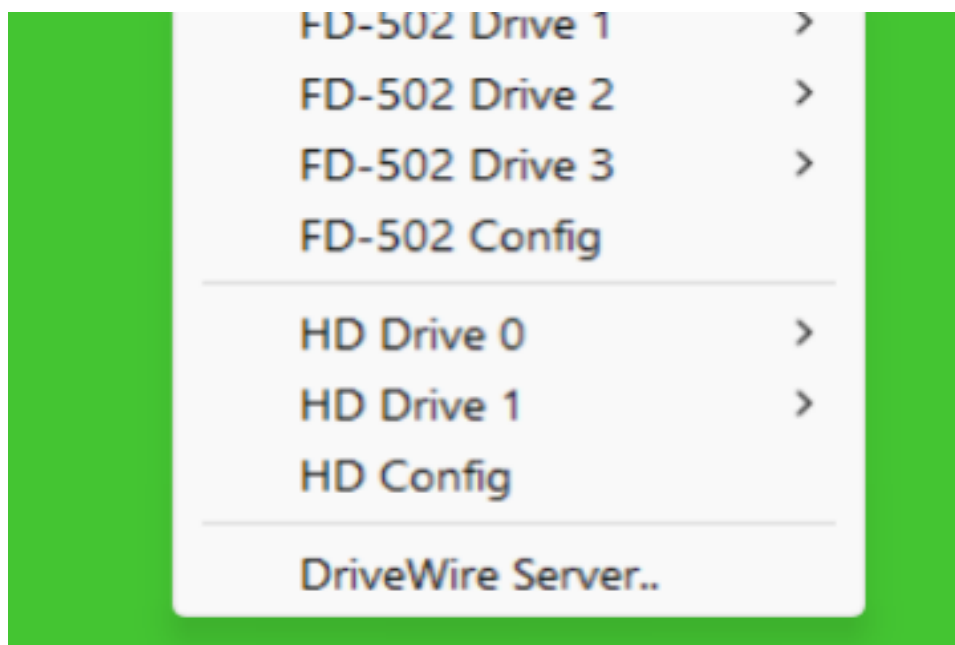
## Clock is Read-Only

If this option is not selected changing the time in the emulator will affect the time of the host computer.

## Becker Port

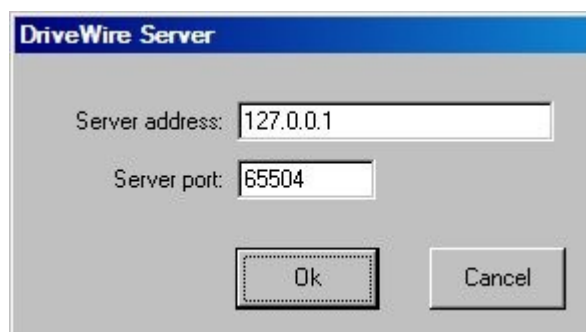
DriveWire4 (DW4) has opened the doors to the outside world to the Coco and now it's available for VCC. To use DW4 with VCC, you must insert the "becker.dll" into one of the MPI slots, but DO NOT SELECT THE SLOT. You will need to use the "External ROM" feature of the FD-502 controller to load the becker version of HDBDOS.

The becker.dll cart will add another menu option to the "Cartridge" menu:



This menu option allows the setting for the Becker port to be changed to fit the user's needs. In most cases, the default settings are what is needed and it should not have to be changed.

The default settings are:



These settings should be sufficient for most DriveWire4 uses. In the DriveWire4 server "Simple Config Wizard" utility, you will use the "Emulator or other TCP/IP" selection and use the default

settings from there.

To use DW4 from BASIC, you must have the "Cartridge/FD502 Config/External Rom" selected and the browse to your HDBDOS rom in the "Browse" box. This rom is also used when "auto booting" NitrOS9 from an HDBDOS virtual floppy drive using Robert Gault's RGBDOS menu system. If you are using "multi-partitioned" VHD images with NitrOS9 and HDBDOS images on one VHD, then you will have to use an "offset" in HDBDOS to reflect the size of the NitrOS9 partition to get to the HDBDOS partition. An explanation to this offset can be found on Robert Gault's website: [<http://aaronwolfe.com/robert.gault/Coco/Downloads/Downloads.htm>].

Scroll to the bottom of the page for RGBDOS. Almost all that applies to RGBDOS will also apply to HDBDOS (in most cases).

## VCC Bugs

VCC definitely has a few bugs, and some are more "evasive" than others.

The CPU is not exactly "cycle accurate". It's just a hair slightly faster than a real CoCo 3 and there are a couple of things that this causes problems with. For example "Popstar Pilot" has a screen flicker and a couple of SockMaster's demos will not work correctly. These programs count cycles in the render process and VCC is slightly off in timing.

When using VCC & DriveWire4, sometimes when doing a cold start (F9 twice), VCC & DW4 will lose sync, and VCC will have to be restarted.

There's a problem in the disk drive emulation in which IRQs are not being handled correctly. It doesn't bother "normal" use, but if you play around with the drive commands directly via ML, it will show it's nasty head.

If you know any other bugs in VCC, please report them to the "Issues" page on the VCC GitHub site where you downloaded this installation package.

View documents online: <https://github.com/VCCE/VCC/wiki>