# CoCo3FPGA

Users Guide for Version 5.X

For MiSTer

Ver 1.2

## Introduction

CoCo3FPGA is a Verilog/VHDL implementation in a programmable FPGA of a Tandy / Radio Shack Color Computer 3 (CoCo3). It gives a super set of the functionality of the original CoCo3.

CoCo3FPGA version 5.x is implemented into MiSTer. The external 128MB ram board is required for functionality.

This full release of CoCo3FPGA includes the programming files, .RBF, for MiSTer. It also includes the Verilog/VHDL source code.

## Processor

The CoCo3 came with a Motorola 6809 processor. The default speed was 0.89 MHz and it had software programmable setting to double the speed to 1.78 MHz. CoCo3FPGA also runs at the default 0.89 MHz. and has the same double speed.

Through the MiSTer's 'On Screen Display' (OSD), there are additional "Turbo" speeds of 3.58 MHz and 7.16 Mhz selectable. Other versions of CoCO3FPGA have run at 25 MHz, but the MiSTer version uses SDRAM and acheiving that is not possible in the present hardware. In the 7.16 Mhz Turbo mode it is possible a few CPU cycles will be delayed for video read cycles. Additionally a 'Force Turbo' mode has been added to the MiSTer's OSD. When 'ON' the high speed mode will ALWAYS be enabled without consideration of the status of the high speed poke.

Note: To use these higher speed "Turbo" modes, they must be selected in the MiSTer OSD and the high-speed poke must be executed. Normal speed operation is no affected and remains at 0.89 MHz.

The 6809 logic core, CPU09, has been borrowed from the System09 on opencores.com. The author is John Kent. It is not cycle accurate to the Motorola 6809. The crude measurements taken from me and helpers on the web show this CPU to be approximately 15% faster than the original 6809. But not all instructions are completed faster than the 6809. This is a benefit for most applications, but a hindrance for others. If the application was written with tight timing requirements, then it will probably fail under CoCo3FPGA.

## ROM / RAM

The ROMs for CoCo3FPGA are the original CoCo3 ROMs with no modification. The CoCo3 ROMs should be loaded into the games/coco3 area of MiSTer as boot roms as follows:

| ROM CONTENTS | MiSTer Filename | SIZE |
|---|---|---|
| CoCo3 | romboot0.rom | 32KB |
| Extended Disk Basic | romboot1.rom | 8KB |

These files MUST be present at the time of selection of the 'core' inside MiSTer's OSD. The first activity is loading these files into the core. Without them present - you will be presented with a blank screen.

The original CoCo3 came equipped with 128K of Dynamic RAM. There was an upgrade available for 512K. Several third-party vendors sold upgrades to 1Meg or even greater.

The CoCo3FPGA on MiSTer comes set with memory size of one of the following: 512K, 1M, 2M, 16M. This size can be selected from the MiSTer OSD.

Along with the addition of larger memory capacities, another feature has been added to the GIME-X to accelerate access to the full memory space. The only way for the CoCo3 to access all its memory is with a memory paging scheme. This paging scheme is well documented. GIME-X adds a different method. The new method uses 9 previously unused IO bytes in the $FFEX hardware area to read and write to memory. By writing the destination address into three IO addresses and the data into a fourth IO address, this data will be written into the memory location specified in the three destination address bytes. Also, by writing a source address into three different IO addresses and reading from the same data IO address used by the writes, the data returned will come from the memory specified by the three origin bytes. By setting the origin address to a different location than the destination address, memory can be moved from one area to another. The GIME-X

supports 8 or 16 bit reads and writes in this new memory access method. One last IO address contains the control register. When bit 0 is set to 1, this enables auto increment the destination addresses on writes. When a data byte is written to a destination specified in the address bytes, the destination address bytes are incremented by 1. This saves the CPU from having to reprogram the bytes each time sequential bytes needs to be written. Bit 1 works exactly the same way except for the origin addresses during reads. When a byte is read from an origin address, the origin address will be incremented by 1. No other bits are used in the control register, but these need to be set to 0 for forward compatibility. To copy memory from one location to another is as simple as programming the source and destination address then looping reads and write to the data register the number of bytes you want to copy. Here is a list of the nine IO bytes and the description of their functions.

| IO | Function |
|---|---|
| $FFE1 | High 7 bits of Destination Address |
| $FFE2 | Middle 8 bits of Destination Address |
| $FFE3 | Lower 8 bits of Destination Address |
| $FFE4 | High 7 bits of Source Address |
| $FFE5 | Middle 8 bits of Source Address |
| $FFE6 | Lower 8 bits of Source Address |
| $FFE7 | Auto Increment Control Register*<br>Bit 1 is Reads Auto Increment<br>Bit 0 is Writes Auto Increment |
| $FFE8 | Write / Read Data |
| $FFE9 | Write / Read Data |

## MPI Slots

The original CoCo3 included a single slot to plug in additional Program PAKs. An optional Multi-PAK Interface (MPI) upgraded the system to a total of 4 PAKs. The MPI has been implemented into the CoCo3FPGA. The MPI is controlled functionally through the MiSTer OSD. The menu item is called 'Multi-Pak Slot:' and it contains options as follows:

Slot 1 is presently unused.

Slot 2 is used by the CoCoSDC controller. This slot has been mapped to include the Disk Extended Color Basic ROM – the same as slot 4. [see Slot 4 below for attaching the rom] The CoCoSDC has two internal floppy drives which function the same as the floppy drives in slot 4. Besides the floppy disk hardware attached to this slot, another register level interface for the CoCoSDC is installed. This allows slot 2 to look like a cocosdc in a super-floppy environment for NitrOS9. Specifically, the .vhd file for the 6809 NitrOS9 Ease of Use Project may be attached and booted.

Slot 3 contains the 'blank' cartridge slot which can be loaded through MiSTer's GUI. Note: upon first boot if no cartridge is loaded then Extended Color Basic is loaded.

Slot 4 is used for the Disk BASIC ROM and disk interface. The CoCo3 Disk BASIC ROM should be loaded upon boot via 'boot1.rom'. Up to 4 floppy drives are supported with 0-3 also supporting double sided operation. Double sided operation is identified by looking at the disk size supplied. If the .dsk file >160K then double sided is enabled.

Note - each time a different option is selected for the 'Multi-Pak Select' the core will perform a programmed reboot of the coco by displaying the Easter Egg for a fraction of a second, then a reset.

A feature added to CoCo3FPGA is the ability to disable the interrupt signal that causes a PAK ROM to auto-start in slots 1 and 3. Slots 4 is disk controller slots, so no auto-start interrupts are implemented for these slots. By turning 'Disable Interrupts' to the On position in the OSD Debug menu, the auto-start interrupt is disabled.

## Multiple Cartridge ROM System

Not used in this implementation.

## Flash Programming

Not used in this implementation.

## Memory Map

The memory map for the lower 512K RAM memory is exactly the same as the original CoCo3. Most of the hardware IO page is identical to the original CoCo3.

## Keyboard, CPU RESET, and CoCo3 Easter Egg

The CoCo3 includes a 57 key keyboard. CoCo3FPGA uses a PS/2 keyboard that emulates the original 57 key keyboard. There is no additional software needed and all the original CoCo3 programs will work without modification. The keyboard layout is not the same between the original keyboard and the PS/2 keyboard. CoCo3FPGA translates the PS/2 key layout to the CoCo3 layout. So when you push a [shift] 8 on the PS/2 keyboard, CoCo3FPGA will display a "*" on the display. Pushing the [shift] 8 key on a CoCo3 will display a "(" on the screen. Whatever key you push on the keyboard comes up as what is labeled on the keyboard. No need to memorize the CoCo3 layout. One the CoCo3 keyboard, a Shift – 0 is used to toggle shift lock. A Shift – 0 on a PS2 keyboard is the ) key. The PS2 does contain a caps lock key. This key has been programmed to output a Shift – 0 to toggle the shift lock.

Because some PS2 keyboards put out initialization characters after first power up, the keyboard is ignored for less than a seconds after any type of RESET. CoCo3FPGA will not accept any keyboard input for this short time after any system RESET. An additional feature of the keyboard interface is a CPU RESET. Inspired from the PC, hitting the keyboard combination Ctrl-Alt-Del will send a RESET signal to the CPU. The DE-1's push button 3 will also assert the CPU RESET.

The CPU RESET can also be asserted by using the keyboard combination Ctrl-Alt-Ins. This combination also triggers the CoCo3 Easter Egg. An additional way to trigger the Easter Egg is to push button 0 while asserting RESET. An additional method through the OSD can display the Easter Egg. Simply select 'Easter Egg' and it will be displayed until you select 'Reset' from the OSD or type <ctrl> <alt> <del>. Displaying the Easter Egg is a good way to do a cold RESET on the CoCo3 system. When the RESET button is pushed while the system is displaying the Easter Egg, the CoCo3 is booted as if the system is first powered up. A automated 'Cold Boot' is also available on the OSD which will display the 'Easter Egg' for a fraction of a second, then execute another reset.

NitrOS-9 does a few things differently with the keyboard. This will be discussed later in this document.

## Video

The MiSTer system includes a DB-15 Video connector for use with a VGA monitor. Natively the CoCo3FPGA will put out 15Khz video. In this way it is the same as the original COCO3. MiSTer's scan doubler is attached to its output. Through the ini file VGA may be enabled for the DB-15 and the HDMI automatically upscales as required.

The palette registers of the CoCo3 are implemented. The CoCo3 6 bit, 2 bits per color, palette registers have been extended to 12 bits, 4 bits per color, on CoCo3FPGA. With the extra 2 bits per color, a total of 12 bits, allows up to 4096 different color combinations. When writing to the original palette registers, the additional 6 bit registers are written with the same data. This ensures total compatibility with the original CoCo3 palette registers. The additional lower order 2 bits per color can be written separately by writing to the original palette location with bit 7 of the data set to 1. Even though, the palette registers have been increased in depth, there is still a limitation of 16 palette registers so the maximum number of colors that can be displayed at the same time using the palette registers is 16.

To get around the 16 color limitation, a 256 color mode has been added to the CoCo3FPGA. In a CoCo3, address $FF99 bits 0 and 1 sets the maximum number of colors for graphics modes. Only three of the four settings are used, with the other left as undefined. By setting this undefined combination (both bits 1), the CoCo3FPGA turns on the 256 color mode. The CoCo3 has a maximum of 160 bytes / scan line. With this limitation, the maximum horizontal resolution with 256 colors is 160 pixels. The CoCo3FPGA's maximum has been extended to 640 bytes / scan line allowing a 640 pixel 256 color mode. The default color definitions for the 256 color mode in the CoCo3FPGA are different than the ones defined in the rumored 256 color mode in the CoCo3. CoCo3FPGA uses six bits of color (two bits for each primary color) and two bits of intensity. The lower six bits are defined just like a palette register. But the upper two bits are used as a multiplier for these colors. The multiplier works on all three primary colors.

| Bit 7 | Bit 6 | Multiplier |
|---|---|---|
| 0 | 0 | 2 |
| 0 | 1 | 3 |
| 1 | 0 | 4 |
| 1 | 1 | 5 |

These values were chosen to allow the maximum flexibility in displayed colors. The multipliers 0 and 1 were not used. A multiplier of 0 means Black. Black can be obtained by setting all the color bits to 0. If the colors are all set to 0. the Multiplier does not make any difference. This means there are four versions of Black. This limits the actual number of colors that can be displayed to 252.

The default 256 colors are contained in an internal RAM block in the FPGA. Because they are in RAM, they can be changed. This means the 256 color palette can be changed. To change one of the 256 colors, write the desired 12 bit color data into Palette 0. Then write the color number that is to be changed into IO address $FF7E. The color contain in Palette 0 will be written into the 256 color RAM at the address specified in IO location $FF7E.

| $FF99<br>Bits 4-2 | Bytes / Scan Line |
|---|---|
| 000 | 16 |
| 001 | 20 |
| 010 | 32 |
| 011 | 40 |
| 100 | 64 |
| 101 | 80 |
| 110 | 128 |
| 111 | 160 |

The only Semi-graphics (SG) mode supported with the CoCo3 was SG4. CoCo3FPGA supports all the original Semi-graphics modes supported by the Color Computer 1 and 2. In most cases, software written using the Semi-Graphics will run with out any modification. The one caveat is SG6. On the CoCo3, the settings bit used to enable SG6 was re-tasked to enable lower case text. To get around this limitation, a switch on the DE-1 is used, SW5. If a program is run that uses the SG6 mode, turn on SW5. Because the original setting bit enables lower case text, any program written to run with SG6 can also display lower case text. Most Semi-graphics modes disallowed using text on the same screens. This limitation is gone with CoCo3FPGA. All Semi-graphics modes will now display text. Modes SG8, SG12, and SG24 modes use multiple lines of memory to display 12 line of graphics. To display a full text characters in these modes, the text will need to be duplicated on each line. As an example, SG8 uses 4 lines of memory to display the 12 lines of a text character. To display text, the characters to be displayed will need to be duplicated on all 4 lines. One thing to remember, SG modes require the high bit of memory to be set. The text on the same line will have the high bit cleared.

## Programmable Character Generator

A Character Generator ROM is used to draw characters on the text screens. In the original CoCo3, this was Read Only. But the GIME-X allows this to be modified. The Storage used to hold this data is 2 KBytes in size. Each character takes 16 bytes, so 2 KBytes will hold 128 characters. The number of the character, in HEX, and the character is shown in this table.

```
00 Ç 10 ó 20  30 0 40 @ 50 P 60 ^ 70 p
01 ü 11 æ 21 ! 31 1 41 A 51 Q 61 a 71 q
02 é 12 Æ 22 " 32 2 42 B 52 R 62 b 72 r
03 â 13 ô 23 # 33 3 43 C 53 S 63 c 73 s
04 ä 14 ö 24 $ 34 4 44 D 54 T 64 d 74 t
05 à 15 ø 25 % 35 5 45 E 55 U 65 e 75 u
06 å 16 û 26 & 36 6 46 F 56 V 66 f 76 v
07 ç 17 ù 27 ' 37 7 47 G 57 W 67 g 77 w
```

```
08 ê 18 Ø 28 ( 38 8 48 H 58 X 68 h 78 x
09 ë 19 Ö 29 ) 39 9 49 I 59 Y 69 I 79 y
0A è 1A Ü 2A * 3A : 4A J 5A Z 6A j 7A z
0B ï 1B § 2B + 3B ; 4B K 5B [ 6B k 7B {
0C î 1C £ 2C , 3C < 4C L 5C \ 6C l 7C |
0D ß 1D ± 2D - 3D = 4D M 5D ] 6D m 7D }
0E Ä 1E ° 2E . 3D > 4E N 5E é 6E n 7E ~
0F Å 1F ƒ 2F / 3F ? 4F O 5F ç 6F o 7F _
```

Look in appendix A for the hex dump of the default Storage contents. As an example, character 40 hex is A. The contents for this character, in HEX, is:

08, 14, 22, 22, 3E, 22, 22, 00, 00, 00, 00, 00, 00, 00, 00, 00

If you create a Bit Map of this data it looks like this:

```
Byte 0      08    00001000     *
Byte 1      14    00010100    * *
Byte 2      22    00100010   *   *
Byte 3      22    00100010   *   *
Byte 4      3E    00111110   ******
Byte 5      22    00100010   *   *
Byte 6      22    00100010   *   *
Byte 7      00    00000000
Byte 8      00    00000000
Byte 9      00    00000000
Byte 10     00    00000000
Byte 11     00    00000000
Byte 12     00    00000000
Byte 13     00    00000000
Byte 14     00    00000000
Byte 15     00    00000000
```

Both CoCo 1/2 mode and the CoCo3 mode uses this same characters. While in CoCo 1/2 mode, the scan lines start with line 15 then loops around to 0 through 10. This is a total of 12 lines. Because it starts with line 15, this creates a blank line above the character. This is a standard feature of the CoCo3 in this mode. You can see the byte in the CoCo3 ROM code at $C232. This shows the default setting for 32 column mode is 15 in the VERTICAL SCROLL REGISTER. In CoCo3 mode, the VERTICAL SCROLL REGISTER is set to 0, so there is no scan line above the character in this mode. This allows the same Character Generator ROM to be used for both modes.

But the MiSTer port has two character ROMs. The HI-RES text screens use one of the character generator ROMs and the CoCo1/2 compatible 32 character text screens use the other. In the CoCo1/2 character ROM, the characters have been moved down two lines to make it more compatible with some of the CoCo1/2 programs. In the rare case where the CoCo1/2 text screens need to use the original CoCo3 characters, there is a method to swap the two ROM banks. This feature is explained below.

The process of modifying a character, has several steps:

1. Unlock the desired bank for writing:
   - (a) Write $A5 into $FFF0 to unlock the CoCo1/2 Character generator ROM Bank
   - (b) Write $5A into $FFF0 to unlock the CoCo3 HI-RES Character generator ROM Bank
2. Choose the character you want to replace:
   - (a) Write $00 to $7F into $FFF2 to choose which character to replace.
   - (b) This is optional, when unlocking the Bank, the character is automatically set to 0.
3. Write 16 values into the character.
   - (a) The address of the Character ROM is automatically incremented with each write.
   - (b) To write two consecutive characters, write 32 values.
   - (c) All 127 characters can be written with out ever changing the character number.
4. Write 0 into $FFF0 to re-lock the Character ROM Banks.

Any changes made to the Character Generator Storage will be reflected to the screen immediately. And changes made to the Character Generator Storage will persist until the original contents are written back into the Storage or the CoCo3 is powered off then back on.

By writing a $C3 into $FFF0, the two Character ROM Banks are swapped. Meaning the original CoCo3 Character ROM Bank is used for the CoCo1/2 text screens.

In addition to the documented programmable interface described previously, the MiSTer port has the ability to load a font file. This is a binary representation of both font banks must have a .bin extension and must be stored in the /games/CoCo file area. The font load in in the 'Debug' sub-menu of the MiSTer OSD.

## Floppy Interface

The floppy disk controller in other implementations converts the floppy to a DriveWire serial data stream. This is not supported in the MiSTer implementation. Instead, a hardware implementation connected to the MiSTer SD file interface. To use the floppy interface, you must select Disk from the MPI Select. [Choose ether slot 2 or 4] Then you must select a .dsk file for the associated drive. At this point Disk Extended Color Basic can read and write to the drives as normal. The .dsk files must be placed in the games/coco3 directory of the MiSTer fle system.

Disk Extended Color Basic supports 4, single sided 35 track, 18 sector disks. The hardware supports double sided floppy which can be used in OS9. The OSD examines the .dsk file size and mounts the disk as double sided if its size is larger that a 35 track single sided disk.

The track write function of the WD1773 chip is not supported, therefore the dskini dos command, while it functions and returns with a 'OK' – it does not initialize a disk. Instead, mount a blank disk or use the appropriate kill commands to clean a disk.

Additionally, if you select MPI slot 2, a CoCoSDC register compatible interface has been added to allow super-floppy's to be mounted and ran. The super-floppy is mounted as a .dsk or .vhd file. The intension of this interface is to enable the use of the NitrOS9 Ease of Use Project distributions only. Thus, CoCoSDC does not support the full functionality of the actual hardware CoCoSDC. It is limited to only the requirements of the NitrOS9 llcocosdc driver. Because of this, no specific software written for the CoCoSDC will function.

Note that the NitrOS9 Ease of Use superfloppies support a second superfloppy 'blank' hard drive as drive 1 on the CoCoSDC. This second HD is referenced as /H1. Also while the NitrOS9 CoCoSDC driver shuts down normal floppy access on the CoCoSDC, NitrOS9 does see the floppy controller in slot 4. The On Screen Display for control of MiSTer functions has been updated to allow slot 4 floppies to be mounted for function under NitrOS9. Also note, that at the time of writing the Ease Of Use only has supplied floppy descriptors /d0 and /d1. Use of Drive 2 and 3 from slot 4 will require a updated boot file.

### RS232 PAK

The RS232 PAK is implemented in CoCo3FPGA. The Terminal program form the PAK is not implemented, but can be loaded if needed. But there are much better terminal programs available for both Disk BASIC and NitrOS-9.

The MiSTer OSD allows the serial port to be connected to one of the following: None, PPP, Console, MIDI, or Modem. The Console will connect you with the Linux system underneath MiSTer. The Modem connection allows you to use standard modem 'AT' commands to connect to the internet via IP address…. i.e. ATDT 192.168.1.1…

The internal RS-232 port is not used in the MiSTer port.

## Sound

The original CoCo3 sound is implemented using the MiSTer system sound hardware. Plug in a set of headphones or speakers to hear the sound. The volume is loud, so a volume control is mandatory when listening. Along with the original CoCo3 sound, the Orchestra-90 sound is also implemented using the same hardware. No additional setup is needed to listen to the sound from the Orchestra-90CC. The Orchestra-90CC sound hardware has been extended to give 16 bit sound. The normal 8 bit sound interface uses two addresses to program, $FF7A for left channel and $FF7B for the right. CoCo3FPGA uses two additional addresses to extend the two 8 bit registers to 16 bits. Addresses $FF7C is the lower 8 bits for the left channel and $FF7D for the right channel.

Writing into $FF7C and $FF7D only buffer the data. It is actually written into the sound hardware registers when the accompanying most significant 8 bit address is written. This means the data written into $FF7C does not take effect until data is written into $FF7A. The same is true for the data written into $FF7D only takes affect when $FF7B is written.

### Joysticks

The MiSTer supported controllers are mapped to the CoCo3_FPGA port.

### Mouse

A MiSTer supported Mouse nay be mapped to the right joystick. This is primarily of benefit to the NitrOS9 Easy of Use distribution upon invoking the 'gshell' command.

### Revision Bytes

To determine which revision of CoCo3FPGA is running, two previously unused bytes have been programmed with a revision number. The 6809 CPU uses $FFF2 - $FFFF for RESET and Interrupt vectors. The two bytes $FFF0 and $FFF1 now hold the revision in a hex format.

| | |
|---|---|
| $FFF0, bits 7-4 | Major Revision |
| $FFF0, bits 3-0 | Minor Revision |
| $FFF1, bits 7-4 | Implementation, 0000 = DE1, 0001 = DE2-115, 0010 = MiSTer |
| $FFF1, bit 3 | Analog board, 0=Gary's (or none [MiSTer]), 1=Ed's |
| $FFF1, bits 2-0 | Max Memory Size, 000 = 128K, 001 = 512K, 1M or 2M, 010 = 5 Meg |

Previous to CoCo3FPGA revision 3.0.0.1, these two bytes hold all 0.

## NitrOS-9 Special Features

### Floppy Interface

The standard FDD controller has some additional features that will not be supported by normal software. Some support is available using the standard FDD driver for NitrOS-9. One feature is the number of tracks that can be set. There are several types of 5.25" floppy disks most supporting 35, 40, up to 80 tracks. Using the CoCo3FPGA, a total of 256 tracks can be implemented using the standard register to program the tracks. This "double sided floppy image" can contain over 2 Mbytes of storage.

In addition to the extended number of tracks, the drive selects can be used to address a total of 8 double sided disks or 16 single sided disks. NitrOS9 Super-floppy's in the form of .dsk or .vhd files are supported through a register compatible cocosdc interface.

### SD Card Interface

The SD Card Interface implemented in other implementations of the CoCoFPGA is not supported.

### SDRAM

The SDRAM ram-disk implemented in other implementations of the CoCoFPGA is not supported.

### Real Time Clocks

The CoCo3FPGA has a real time clock. This clock uses the CLOCK2_JVEMU module in NitrOS-9.

The IO addresses are:

| Port Address | Value |
|---|---|
| $FFC0 | Century |
| $FFC1 | Year |
| $FFC2 | Month |
| $FFC3 | Day |
| $FFC4 | Day of Week |
| $FFC5 | Hour |
| $FFC6 | Minute |
| $FFC7 | Second |

The hardware implementation for this clock is not complete. The ability to keep track of all changes up to the Century takes an excessive amount of logic. Instead, only the logic to update seconds, minutes, hours, days, day of week, month, and year is implemented. Century is set as a constant of 20. At the start of the FPGA module, Linux initializes the RTC one time. If the MiSTer system is connected to the internet the Linux supplied time information will be accurate.

## Keyboard

There are several keyboard codes that take on special meaning in NitrOS-9.

| Key | Result |
|---|---|
| Control / | \ |
| ~ | Control 3 |
| ^ | Control 7 |
| _ | Control = |
| [ | Control 8 |
| ] | Control 9 |
| { | Control , |
| } | Control . |

Read the Special Key section of "Getting Started With NitrOS-9" for more information and other key combinations that cause special actions to be performed.

## WiFi Module

The WiFi module implemented in other implementations of the CoCoFPGA is not supported.

## Support

The main support for CoCo3FPGA is the CoCo3FPGA groups.io Site. Questions, ideas, bugs, and problems can be raised in the messages. Also, pictures and files are uploaded for discussion and support. The URL for the yahoo Group is

https://groups.io/g/CoCo3FPGA/topics

Questions related to the MiSTer port can be posted to the discord MiSTer group:

https://discord.gg/misterfpga

## Revisions

1.0 Initial Revision

- 1.1 Updated MPI slot 4 and floppy sections to documenting the register level compatible cocosdc interface to be able to run the NitrOS9 Ease of Use Project super-floppy's.
- 1.2 Updated to reflect the movement of the cocosdc to slot #2 and the two additional floppy drives associated. The fact that NitrOS9 finds the real floppy drive controller in slot 4 is a bonus and functions ONLY on OS9. The additional MiSTer mouse to joystick is also documented.