CoCo3FPGA

Users Guide for

Version 5.X

For MiSTer

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, the disk controller cannot be accessed. (This may be addressed in hardware in a future relase)

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. An example of this failure is Sock Master's Boink bouncing ball demo. The Boink demo uses the "mul" instruction to waste CPU clock cycles for timing purposes. The "mul" instruction is one of the examples where the 6809 completes the instruction faster than CPU09.

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 rom	boot0.rom	32KB
Extended Disk Basic rom	boot1.rom	8KB
Orch 90 rom	boot2.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 2MB of SDRAM. (Future releases may allow selection of more or less ram)

The Block Copy feature of the CoCo3FPGA in other implementations is not presently supported.

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 holds the Orchestra 90CC ROM. The sound interface hardware is also implemented. The 8K ROM should be loaded on boot as 'boot2.rom'. If the orchestra 90CC ROM is not wanted, it can be replaced with any 8K ROM.

Slot 2 is not used but implemented and may be selected via programming. No options appear in the OSD for this selection.

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. If Disk Basic has ever been selected it will persestantly show up until the core is rebooted.

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'.

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 2 and 4 are disk controller slots, so no auto-start interrupts are implemented for these slots. By turning Switch 4 to the On position, the auto-start interrupt is disabled. The Disk, RS232 Port, SD card, and WiFi interrupts are not switched off.

The Multiple Cartridge ROM System, MCRS, is a method of using the DE-1 Flash ROM to implement multiple cartridge ROMs of different types into slot 3 of the MPI. The MCRS uses the IO space of slot 3 to program the location in Flash and the type of ROM that is to be used. This is an explanation of the registers used to implement the MCRS.

$FF40 ROM Bank Select

The lower three bits sets the ROM bank. This is the same register location used by the Super Program PAKs to bank select up to 128K of ROM. The Super Program PAKs used a bank select method of swapping out different banks of 16K ROMs into the address space. A total of 8 banks could be selected for a total ROM PAK size of 128K. The normal PAK ROMs used a single ROM that was 16K or less, but some PAKs used the special features of the CoCo3 to allow 32K PAK ROMs. The ROM Bank Select is set to 0 by a system RESET.

$FF41

Not used in this implementation.

$FF42 Flash Block Address for Bank 0

The 2 Meg Flash ROM allows for up to sixty four 32K ROMs. By setting this location to a value between 1-63, the upper six address bits for the Flash ROM (block number) are set when location $FF40 is set to 0. This location is not cleared with a RESET. Because the CoCo3 ROM, Disk BASIC ROM, and the Orchestra 90CC ROM are stored in the lower 48K of the 2 MEG Flash, not all blocks can be used. A single 16K PAK ROM can occupy the upper half of block 1 and all other ROMs must be loaded at block 2 or higher.

$FF43 Bank Size / Sub-Bank Select

Because the maximum size of a ROM can be 32K, each Flash ROM block has to be 32K in size. But since most ROMs are 16K or less, a lot of space can be wasted. To minimize this waste, the 32K block can be split up into two 16K blocks. Bit 0 of this address specifies the size of the Bank.

Bit 0
0 = 32K Bank Size
1 = 16K Bank Size

This Bank size is important. If the 16K Bank Size is set and the CoCo3 is setup for a 32K ROM, the 16K bank will be mirrored across the 32K address space.

To minimize the wasted space, each 32K Flash ROM block can be split into two 16K blocks. Bit 1 of this address specifies which 16K block is to be selected.

Bit 1
0 = Use lower 16K of each 32K block
1 = Use upper 16K of each 32K block

Bit 1 is only effective if bank size is set to 16K by bit 0. This bit determines which 16K block is used for all 8 Super Program PAK banks.

$FF44-$FF4A Flash Block Address for Banks 1 – 7

These registers have the same functionality as $FF42, except they work on banks 1 through 7. The eight 16K banks allow ROMs up to 128K to be used.

Included on the DOS disk image with CoCo3FPGA is a simple BASIC / Assembly program that allows the setup of the MCRS. The program's name is "SETUP3.BAS".

The Flash ROM is slower than SRAM. The Flash access time limits the max frequency to less than 25 MHz. The "Turbo" setting needs to be disabled when using the MCRS.

By default the Flash is write protected. But a special feature has been added to enable the Flash to be programmed. This uses the Multiple Cartridge ROM System and the ability to unlock the write protect on the Flash. To unlock the write protect and map a section of the Flash into slot 3 address space:

1 Turn off Interrupts
(Turns off all RAM mode)

2 Write to $FFDE data does not matter
(Enable 32K external ROM)

3 Set bits 0 and 1 of $FF90
(These next two steps change to slot 3 and removes the write protect for the Flash)

4 Write $2A into $FF7F
5 Write $26 into $FF7F

(Enable MCRS bank 0)
6 Write 0 into $FF40

(Bank 0 address, the 7 address bits should be located in the low 7 bits of the data)
7 Write the Flash Address 15 to 21 bits of the Flash Address into $FF42
8 Write Flash Address bit 14 * 2 + 1 into into $FF43

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.

! " "# $ " "

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. 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.

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

%

The DE-1 board includes a DB-15 Video connector for use with a VGA monitor. The standard video modes of the CoCo3 are displayed as a 640 x 480 60 Hz VGA video mode. Each horizontal video mode is upscale to fit into the VGA video resolution. Every CoCo3 video mode scan line is represented by two scan lines on the VGA monitor. The VGA 640x480 60 Hz video mode timing is similar to but not exactly the same as the original CoCo3 timing. The pixel clock rate is 25 MHz (40 nS). And each scan line is 800 pixels across including the times for horizontal blanking and sync. This means that the horizontal sync pulse is 32 uS apart. The standard CoCo3 sync is 63.5 uS apart. But because each CoCo3 scan line is two lines on the VGA screen, every other sync pulse is invisible to the interrupt and timing circuitry. This gives an effective sync rate

of 64 uS. The vertical time is also similar but not identical. Each frame consists of 521 scan lines including the vertical blanking and sync times. This gives a vertical sync pulse every 16.672 mS. The standard CoCo3 has a vertical sync time of 16.667 mS.

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.

As previously mentioned, the number of bytes / scan line has been extended to 640 bytes. The register that contains the settings is fully occupied. The additional bit has been added to a register that is previously unused, register $FF98 bit 6.

$FF98 Bit 6	$FF99 Bits 4-2	Bytes / Scan Line
0	000	16
0	001	20
0	010	32
0	011	40
0	100	64
0	101	80
0	110	128
0	111	160
1	000	256
1	001	320
1	010	512
1	011	640

All the standard modes of the CoCo3 are supported with the CoCo3FPGA. The CoCo3FPGA has several features not available with the CoCo3. As previously documented, each CoCo3 is really 2 scan lines on the CoCo3FPGA. Because of this duplicate line, a new feature has been added, a double vertical resolution mode. Register $FF99 bit 7 is undefined in the CoCo3. CoCo3FPGA defines this bit as double vertical mode. When this bit is set to 1, CoCo3FPGA no longer draws each CoCo3 line two times. It draws it only once. This allows twice the number of lines to be displayed on the screen. Every text and graphics mode is changed by this bit. The maximum number of lines goes from the CoCo3's 225 to 450. Of course, the memory used in this mode is doubled. Careful selection of video buffer location and size are needed to utilize this new mode. The amount of memory need to display the maximum resolution of 640x450 with 256 colors is 288,000 bytes. This is more than half the normal available memory.

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.

A new feature has been added to this revision of CoCo3FPGA. Since most video modes have two identical scan lines, there is a switch that can turn the second scan line to half intensity. The appearance of the screen looks similar to the original CoCo3. This feature will disable the double vertical resolution mode. Toggle SW3 to enable this feature.

&

The original floppy hardware is emulated on the CoCo3FPGA. The physical floppy is replaced with a co-processor that speaks DriveWire over the serial port that communicates with a file server. The 6502 co-processor was borrowed from the T65 project on opencores.org. The author is Daniel Wallner. The server program is DriveWire written by Aaron Wolfe. The floppy disk emulator does not need any special software to work. All software tested that works with the CoCo3 FDD controller has worked with this floppy interface.

The speed of the serial port is controlled by slide switches SW7 and SW8.

' ( )

SW8	SW7	Speed
Off	Off	115,200
Off	On	230,400
On	Off	460,800
On	On	921,600

The specification for the RS232 Level converters on the DE-1 board states they will work up to 250,000 baud. But these have been used successfully at the 460800 baud speeds with no problems seen. It will not work at the 921600 baud speeds. The UART output signals for the DriveWire interface can be switched with the signals of the UART for the RS232 PAK. See the section on the RS232 PAK for more specifics. By turning on SW9, and using the RS232 PAK hardware, the 921,600 baud speed can be utilized.

$

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 RS232 PAK signals are routed to expansion Connector 1. The TX signal is on pin 33 and the RX signal is on pin 31. See the DE-1 User Manual for the location of these pins. These signals are not standard RS232 level signals. They are 3.3V TTL signals. An analog board has been developed that can be used to interface to the RS232 PAK and give true RS232 level signals. This analog board has been previously published on the CoCo3FPGA Yahoo group. In lieu of the analog board, the RS232 PAK can be accessed using a special USB serial port cable. A suitable cable can be found on Amazon.com and other sites for under $10. Search for “USB TTL serial.” This cable allows serial communication between a computer system and the CoCo3FPGA. A picture of this connection is available at the Yahoo CoCo3FPGA group in the Photos section, see the link below. The VCC signal should not be connected.

https://groups.yahoo.com/neo/groups/CoCo3FPGA/photos/photostream/lightbox/469950296?orderBy=mtime&sortOrder=desc&photoFilter=ALL#zax/469950296

As mentioned in the Floppy Interface section, the RS232 PAK signals can be swapped with the signals for the DriveWire. Slide switch SW9 is used for this functionality.

! #

SW9	DE-1 DB9	Expansion connector
Off	DriveWire	RS232PAK
On	RS232	DriveWire

The original CoCo3 sound is implemented using the DE-1's 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 affect until data is written into $FF7A. The same is true for the data written into $FF7D only takes affect when $FF7B is written. The DAC used to implement the sound is feed using an I2S bus. This bus takes left and right audio signals and transfers them to the Audio DAC using a serial stream. The sound data for the left and the right channels are loaded into the buffers for the parallel to serial converters simultaneously to prevent any phase distortion.

+ ,

Joystick ports have been developed for CoCo3FPGA. The analog board previously mentioned includes two CoCo3 analog joysticks. The standard Joystick interface supports 6 bit resolution. A higher resolution version of the joystick interface is available. Using 8 bytes of the IO space, each Joystick can have 12 bits of resolution. Unlike the Tandy Hi-Res joystick interface, heavy CPU utilization is not needed for the A/D conversion. The Joystick can be read directly from these IO locations:

Address	Data
$FF60	Right Y High 8 bits
$FF61	Right Y Low 4 bits + 4 bits of 0
$FF62	Right X High 8 bits
$FF63	Right X Low 4 bits + 4 bits of 0
$FF64	Left Y High 8 bits
$FF65	Left Y Low 4 bits + 4 bits of 0
$FF66	Left X High 8 bits
$FF67	Left X Low 4 bits + 4 bits of 0

(-

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	Super Major Revision
$FFF1, bit 3	Analog board, 0=Ed's, 1=Gary's
$FFF1, bits 2-0	Max Memory Size, either 1 or 5 Meg

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

-

The DE-1 board includes 10 slide switches and 4 buttons. Most of the slide switches are explained elsewhere in this document, but are included here for completeness.

Slide Switches	Function
SW0	CPU Turbo Speed
SW1-SW2	MPI Switch
SW3	Half Intensity on Duplicate Lines
SW4	Disable Autostart Interrupt in slots 1 and 3
SW5	SG6 Enable
SW6	SD Card Presence / Write Protect Override
SW7-SW8	DriveWire Speed
SW9	UART Switch

Buttons	Function
KEY0	Easter Egg
KEY1	SD Card Inserted
KEY2	SD Card Write Protect
KEY3	System Reset

.'''

The DE-1 board comes with four 7-Segment LEDs, ten Red LEDs, and eight Green LEDs. The four 7-Segment LEDs are programmed to spell out CoCo. This cannot be changed.

The Red LEDs are used as status

LED	Function
LEDR0	FDD Select 0
LEDR1	FDD Select 1
LEDR2	FDD Select 2

LEDR3	FDD Select 3
LEDR4	SD Card Write Protected
LEDR5	SD Card Inserted
LEDR6	CoCo3FPGA in RESET
LEDR7	Keyboard Shift
LEDR8	Address 19
LEDR9	Address 18

The Green LEDs also give status

LED	Function
LEDG0	DE1 SRAM Accesses
LEDG1	Analog board SRAM0 Accesses
LEDG2	Analog board SRAM1 Accesses
LEDG3	Flash ROM Accesses
LEDG4	SDRAM Accesses
LEDG5	WiFi Accesses
LEDG6	SD Card Activity
LEDG7	RS232 PAK Activity

/ 01

&

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. Using special software, a total of 65536 tracks can be implemented.

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. Special software could be written to implement a total of 128 double sided drives or 256 single sided drives.

- - , &

Along with the CoCo3 FDD compatibility, there are other communication modes available on the CoCo3FPGA. The Boisy / Becker Interface is a DriveWire compatible interface Boisy Pitre conceptualized. This simple interface has NitrOS-9 support and allows low overhead DriveWire communications from the CoCo3FPGA to the DriveWire server. Use the NitrOS-9 disk image with “Becker” in the name to make use of this interface. Since the Boisy / Becker Interface is part of the disk controller, its addresses are only available when one of the disk controller slots is selected. The Boisy / Becker Interface uses FIFO buffer memories. This buffer memory needs

very little handshaking to operate. The only status bit is to indicate when read data is available. There is no handshaking needed for writes. If the buffer becomes full during writing, then the CPU09 is sent a halt signal until the buffer can again accept data. The buffer memory is sufficiently deep to avoid the need of the halt signal for most transfers.

The IO address used for the Boisy / Becker Interface:

Address	Function
$FF41	Read Status
	Bit 1 Read Data Available
$FF42	Read / Write Data

' &

This version of CoCo3FPGA has implemented the SD Card slot on the DE-1 board. Presently, the SD Card can only be used under NitrOS-9. The DE-1 includes the connections to the SD Card to communicate with the SD Card using SPI. The SPI bus is running at 12.5 MHz and uses CPU09's halt signal to keep the SPI bus synchronized. The driver for this interface is a low level driver that requires Superdriver from Cloud9 to operate. Superdriver is available for free in the modules directory on NitrOS-9's SourceForge site. The user manual is available at Cloud9's web site. Please read the user manual for specifics of Superdrivers features, see the link below.

A disk image, Master330.dsk, is included in this distribution that has the low level driver file. The disk also includes a pair of device descriptors, /sd0 and /sd1. These device descriptors create disks with just under 128 Mbyte of storage each. They use the Partitioning feature of Superdriver to use the same SD Card to hold more than 1 disk drive. Both partitions can be held on a single 256 Mbyte SD Card.

The driver has been tested with a limited number of SD Cards, but should work with most readily available Cards. It has been tested with 256 Mbyte, 512 Mbyte, 1 Gbyte, 4 Gbyte, 8 Gbyte, and 16 Gbyte cards. It is probably best to use SD Cards that are no larger than 4 Gbyte. A larger SD Card will waste all the storage above the 4 Gbyte. NitrOS-9 uses three byte Logical Sector Number (LSN). The total number of sector that can be held with three bytes times the 256 byte sector length equals 4Gbyte.

The driver has a feature that checks if the card is installed and if the card is locked (write protected). To fully implement these features, two additional wires need to be added to the DE-1 board as these signals are not included for the DE-1 implementation of the SD Card slot. A picture of the modification is available at the Yahoo CoCo3FPGA group in the Photos section.

https://groups.yahoo.com/neo/groups/CoCo3FPGA/photos/photostream/lightbox/1017358464?orderBy=mtime&sortOrder=desc&photoFilter=ALL#zax/1017358464

If you do not want to add the wires and do not care about write protecting the SD Card, SW6 can be set to the on position and these two signals will be ignored. But beware, any writes to the card will be successful even if you use the write protect slide switch on the card. When the SD Card

driver initializes the card, the card is left in a state where reads and writes can be processed. If the card is removed from the slot, the driver will not reinitialize. The easiest way to reinitialize the card is to reboot NitrOS-9.

The interface for the SD Card is Serial Peripheral Interface, SPI. SPI is a three wire interface and runs at 12.5 MHz. When ever a byte of data is sent from the SPI to the SD Card, a byte of data is sent from the SD Card to the SPI interface. There is no one direction transfers. Only the SPI interface can initiate a transfer. If there is no data for the SD Card to send, it will return $FF. If there is no data to send to the SD Card, but the SPI interface needs to read data from the SD Card, it will also send $FF. To facilitate this mechanism, whenever a byte is read by the CPU from the SPI interface, that will initiate another transaction sending the SD Card a $FF. The alleviates the CPU from having to write a byte to read a byte. At "Turbo" speed, the CPU could actually read data faster than it is available. For this reason, the CPU is halted for a very short time whenever a transaction is in progress. After every read operation, one more instruction needs to be executed, even if it is just a NOP. If the SPI is not finished at the end of that instruction, the CPU will be halted until the transaction is finished. The SPI interface has the capability to generate an interrupt whenever a SD Card in inserted. This interrupt can be used to setup the SD Card to be read and written. This function is not currently used in the driver.

/#

The driver for the SD Card has a couple of "features" that do not work exactly as designed. The SD Card Lock will prevent the driver from writing to the card, but there is no indication that the write was not successful. Meaning you can copy bunches of files to a locked SD Card and every thing will seem to go OK, but no changes will be made to the SD Card. The 6809 assembly source for the drivers and the device descriptors are located in the Sources directory in the release.

The IO addresses and functions for the SPI controller are:

Address	Function
$FF64 (Writes)	SPI Control Register
	Bit 0 Slot Select
	Bit 6 SPI IRQ Enable
	Bit 7 SPI Enable
$FF64 (Reads)	SPI Status Register
	Bit 0 Card Inserted
	Bit 1 Card Write Protected
	Bit 7 IRQ Active
$FF65	SPI Data Register

'

The DE-1 FPGA board includes 8MB of SDRAM. This SDRAM is not suitable for use directly by the 6809. But with the controller built into the CoCo3FPGA, will make an acceptable RAM

disk drive. At the present time, a NitrOS-9 driver for this RAM disk drive is not written. But it should become available soon.

The controller for the SDRAM is copyrighted by Xess Corporation. The code can be freely distributed and modified as long as the source header is retained. See the source listing for this message.

The controller is setup to access SDRAM data a sector a 256 byte sector at a time. There is a total of 32768 sectors. The controller is accessed using 5 previously unused IO addresses:

Address	Function
$FF84 (Write)	SDRAM Control Register Bit 0 Read / Write
$FF84 (Reads)	SDRAM Status Register Bit 7 SDRAM Controller Finished Bit 3-1 SDRAM Controller State Bit 0 SDRAM R/W
$FF85-$FF86 (R/W)	SDRAM Data (16 bits)
$FF88-$FF87 (Write)	SDRAM Sector Number (15 bits)

The data to and from the SDRAM is accessed 16 bits at a time. To use, set the $FF84 bit 0 to 1 for a read or 0 for a write. Then set the sector number in $FF85-$FF86. The controller is triggered by reading or writing $FF86. A dummy read to $FF86 is needed to start the controller for the first read operation.

,

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, and days is implemented. The days do not have limit correction, so you

could have a day 32 in a month or an eighth day of the week. When the DE-1 board is connected to a DriveWire server, the RTC is updated automatically approximately every minute.

A real battery backed RTC module is included on the analog board. The RCT is a module mounted on the analog board and uses a Maxim Integrated DS3231 RTC chip. The RTC interface uses an I2C 2 wire bus. CoCo3FPGA uses an I2C controller to talk with the RTC chip. The I2C controller uses 4 IO addresses.

Address	Function
$FF80 (Writes)	I2C Control Register
	Bit 0 I2C Transaction Start
$FF80 (Reads)	I2C Status Register
	Bit 6 Transaction Failed
	Bit 7 Transaction Finished
$FF81(Read)	Read Data
$FF81(Write)	Write Data
$FF82	Slave Address
	$D0 for Reads
	$D1 for Writes
$FF83	Register Addresses

Check the DS3231 data sheet for detailed instructions on reading and writing the registers. The RTC module also includes an AT24C32 EEPROM. The I2C interface can also read and write to this EEPROM. Check the data sheet for the AT24C32 for the Slave Address and register addresses needed.

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.

)

The analog board also contains a WiFi module. The module contains an ESP8266 chip. The interface to this module is similar to the Becker Interface used for DriveWire. It also includes the FIFOs for reading and writing to the WiFi module. This interface also includes interrupt capabilities. By enabling the interrupt, the interface with generate an interrupt when ever there is data available in the read FIFO.

Address	Function
$FF6C (Write)	Control Register
	Bit 7 Interrupt Enable
	Bit 1-0 Baud Rate
$FF6C (Read)	Status Register
	Bit 7 IRQ
	Bit 1 Read Data Available
	Bit 0 Write Data Full
$FF6D (Write)	Write Data
$FF6D (Read)	Read Data

2 3

The main support for CoCo3FPGA is the Yahoo Group 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.yahoo.com/neo/groups/CoCo3FPGA