MiSTer Wiki

Table of contents

MiSTer Wiki

User Manual

Add-Ons

- Assembly (DIY)

)

- Secondary SD card
- User Port (Serial I/O)

)

- Direct Video
- Analog video output compatibility
- RTC board
  - Assembly (DIY)

)

  - Core support
- USB Hub
  - USB Hub Assembly (DIY)

)

- ADC-in (Audio/Tape input)

)

  - Core support
- Case
  - 3D-printed (DIY)
  - Pi-Top (v1)

FPGA Cores

Computers - Classic

- Acorn Archimedes
- Acorn Atom
- Acorn Electron
- Alice MC10
- Altair 8800
- Amiga
- Amstrad CPC 6128
- Amstrad PCW
- ao486 (PC 486)
- Apogee
- Apple I
- Apple II+
- Apple Macintosh Plus
- Aquarius
- Atari 800XL
- Atari ST/STe
- BBC Micro B,Master
- BK0011M
- Camputers Lynx48,96
- Color Computer 2, Dragon 32
- Color Computer 3
- Commodore 16, Plus/4
- Commodore 64, Ultimax
- Commodore PET
- Commodore VIC-20
- Compukit UK101

DEC PDP-1

EDSAC

Einstein TC01

Galaksija

Interact Home Computer

Jupiter Ace

Laser 310

MSX

MultiComp

NEC PC8801

PMD 85

Ondra SPO 186

Orao

Oric 1 & Atmos

RX-78

SAM Coupe

Sharp MZ Series

Sinclair QL

Sord M5

Specialist/MX

SV-328

TI-99/4A

TRS-80 Model 1

TSConf

Vector 06C

X68000

ZX Spectrum

ZX Spectrum Next

ZX81

Consoles - Classic

Adventure Vision

Astrocade

Atari 2600

Atari 5200

Atari 7800

Atari Lynx

AY-3-8500

ColecoVision, SG-1000

Emerson Arcadia 2001

Fairchild Channel F

Gameboy, Gameboy Color

Gameboy, Gameboy Color 2P

Gameboy Advance

Gameboy Advance 2P

Genesis/Megadrive

Intellivision

Other Systems

Arcade Cores

Service cores

Development

)

Donate

MiSTer is fully open source project and will remain open. This is my hobby and i'm getting fun to make hardware and software for this project. While software part just requires time, hardware requires money as well. To release a working add-on i have to make some test samples. Sometimes disaster happens and some equipment/device dies during experiments. Some bought devices/components intended to be MiSTer companions turned to be useless. So i have to try many options before give advice to users.

I'm not a big fan of asking donation for hobby projects, but this project sometimes requires investment to develop. It's absolutely voluntary, you don't have to. But if you satisfied my work and would be glad to donate, then here is my Paypal:

- Paypal
- Patreon

FAQ

This page contains commonly asked questions and their answers.

Table of Contents

- Why doesn't this core from another repo work? Why is MiSTer so hard to use?
- When is an N64 core coming?
- When is PlayStation core coming?
- Other cores work but I get a black screen for this one. What do I do?
- My CD games don't work! Help!
- What version of the CD card bios do I need for TG16 CD?
- How do I make MiSTer boot into the NES core and a specific game?

Other

- I heard the DE10-Nano board uses subsidized components. Is MiSTer doomed if that stops?
- I see a defect that some other people aren't seeing. Should I get a different de-10 nano?
- I found some other Cyclone based dev board on sale and it has built in WiFi. Can I use it for MiSTer?
- The DE-10 is rated for up to 100°C operation and it doesn't get nearly that hot. Do I really need a fan or heatsink?

When will MiSTer support cartridges?

MiSTer will never use physical cartridges.

The project aims to replace the need for having original hardware for the same experience. It is also physically impractical/impossible given the number of GPIO pins available from the FPGA.

Does MiSTer have lag?

It depends on your setup, but it ranges from imperceptibly low to around one frame of lag.

Zero latency is possible with certain equipment and tweaks. See here for a more detailed explanation.

I've seen in the news, "Update Framework". What does that mean?

The 'framework' is all the common elements between cores that handle things like IO, video scaling, etc.

Can I use original console controllers?

Yes, there are many USB adapters for original console controllers. They can also be connected directly via the IO board as detailed below.

What about light guns?

Light guns such as the NES Zapper are too timing sensitive to work over USB, but will work fine on supported cores via the IO board as detailed below.

Any USB controller recommendations?

Yes, please refer to this page.

What are the USB controller options?

Almost any controller that uses USB will work with MiSTer. You can also use a Bluetooth or 2.4ghz USB adapter for wireless. To reduce input latency USB may be overclocked which works with some controllers. For more information, check the links below:

- USB overclock instructions
- USB controller performance data
- USB DaemonBite (known fast controller adapters):

Can I increase the polling rate of my USB controller to improve input latency?

Yes, you can. See here.

Do I need the official USB Hub Add-On Board?

No, but you will probably want some sort of inexpensive OTG hub at least (or a regular hub with an adapter). Use a powered hub if you have many devices.

Do I need to have a USB keyboard with me to use MiSTer?

Yes, it's a good idea to always have a USB keyboard available. Cheap wireless keyboards exist that you can use to reduce clutter.

How do I add a second controller to MiSTer?

First setup the controller in the main menu after plugging it in with no core loaded. Then you can define it in the various cores' OSD menus.

Does MiSTer need an IO board?

No. The IO board is optional, but offers features that could be important for some users (3.5mm and optical audio, CRT output, tape audio input, etc).

What are the methods for connecting controllers to the serial port of the IO add-on board?

SNAC (Serial Native Accessory Converter) is used for direct wiring. Supporting cores (SNES, Genesis, NES, and TG16) allow you to directly connect original controllers. See this page for details on SNAC.

Do I need an IO board to get analog video output to my CRT?

No. You can use an HDMI to VGA adapter to do it instead. See Direct Video.

Is MiSTer hard to set up? Is it really only for technical people who like to tinker?

No and no, but those who enjoy tinkering can get a lot extra out of their MiSTer if they wish. The minimum setup only requires attaching a memory board to a DE-10 Nano and making an SD card. See this Setup Guide.

Which cores require SDRAM?

You can find the updated list at the Cores-that-use-SDRAM section.

Why does the NES core require an SDRAM add-on board when the Genesis core does not?

The Genesis has graphics RAM and stores data that is copied from the cartridge ROM. The NES does not do this, and typically accesses the ROM directly for the data which requires much tighter timings. Using the SDRAM board on the NES core allows meeting these timing requirements.

Does my MiSTer need cooling?

Yes, you will want at least a heatsink (passive cooling). 22mm x 22mm is the ideal size. Active cooling (a fan) helps but is not strictly required. Some faster cores like ao486 may generate a lot of heat.

What power supply is compatible with MiSTer?

The DE10-Nano board needs a 5V power supply with at least 2A. The connector is a coaxial "barrel" plug of 5.5 mm outer diameter and 2.1 mm inner diameter, center positive. If you are using a USB Hub addon board and have a lot of power hungry peripherals attached to it (wifi, bluetooth, multiple controllers), it might be a good idea to upgrade to a similar 5v power supply, but with more amps (4A-5A).

My MiSTer needs a case. What should I do?

There are two routes you can take; either make it yourself or purchase a case from an online vendor. For DiY, you can find the necessary 3D print

files online (e.g. thingiverse).

What kind of screws do I use with the DE-10 Nano's 14mm long brass standoffs?

M3 screws, 8mm is a good length.

How do I set up MiSTer for 1080p, shimmer free gameplay?

Edit mister.ini on your SD card for 1080p by setting `video_mode=8`. You will also need to use a filter in each core, which you can select in the system core menu (press OSD then 'right'). If you have no filters files, you can download Interpolation sharp here.txt) and put it in your `/media/fat/filters/` folder.

How do I get the scanlines filters on my MiSTer to look good?

You will need to set the device to integer scaling for best results. Set vscale_mode=1 in mister.ini.

"Integer scaling" is when your source resolution (240p) is divided into your target resolution (1080p) using only whole numbers (e.g. 1, 2, 3, 4...). This means if you use integer scaling with the MiSTer set to a 1080p resolution in the MiSTer.ini, and the core you are using natively outputs 240p, then the image will be scaled 4x which 4 times 240 equals 960. This leaves a black border for the remaining 80 pixels distributed across the top and bottom the screen.

You can play with other settings if you want the scaler to fill the screen, but scanlines won't be "perfect" unless the source resolution (e.g. 240p) divides evenly into the target resolution (e.g. 1200p at 5x scaling).

I have a 4k or other higher than 1080p display, does MiSTer support that resolution? I'd like to enjoy better video scaling.

MiSTer can't quite reach 4k. Edit mister.ini on your SD card to change resolutions, e.g. video_mode=12 for 1440p or video_mode=13 for 1536p (hardware maximum). Compatibility with these higher resolutions may depend upon your display and signal chain.

How does the accuracy level of various MiSTer cores compare to other FPGA options?

Most MiSTer cores are just as, if not more accurate, as any of the other major FPGA offerings available today. Most people would not able to tell the difference between these cores and the original hardware.

Will any MiSTer core ever get save states?

The Gameboy, GBA, Lynx, and NES core supports save states.

Cores need to be written from scratch to support them (as was done with the GBA and Lynx cores) or they need to be reworked to support pause before save states can be added (as was done with the Gameboy and NES cores). At this time there are no specific plans to apply this to other cores, but it may happen one day.

Why doesn’t this core from another repo work? Why is MiSTer so hard to use?

MiSTer repositories are self contained and the official updater script only fetches cores from the active official MiSTer repositories. Cores from other repos are not fully integrated so your results may vary. Some developers have their own discord servers or forums and you can seek support there.

When is an N64 core coming?

Probably never. There isn’t really sufficient bandwidth.

When is PlayStation core coming?

Probably next year. Serious progress has been made by a few developers but there’s no official ETA.

Other cores work but I get a black screen for this one. What do I do?

Try setting vsync to 0 as your display may not support all refresh rates. (Note that this was the default setting.)

My CD games don’t work! Help!

Unzip them. Do not zip CD games. Also put them in iso/wav/bin/img + cue format.

What version of the CD card bios do I need for TG16 CD?

Use Japanese version 3.0. You will commonly find it with filename: `Super CD-ROM System (Japan) (v3.0).pce`

How do I make MisTer boot into the NES core and a specific game?

You need to use two different options: autobooting a core, and starting the core on a given ROM. Here is how: In the .INI file, set `bootcore=NES_20201102.rbf` (or the specific core version you have), and comment out `;bootcore_timeout`. Then on your NES games folder (e.g. `/media/fat/games/NES/`, copy the FDS bios as `boot0.rom`, and your .NES rom to boot as `boot1.rom`. For more options, please refer to the NES core documentation

I heard the DE10-Nano board uses subsidized components. Is MiSTer doomed if that stops?

De10-Nano is manufactured in very large scale for use by students and is widely available. When it reaches end of life, the open source cores and infrastructure will be ported to another widely available board.

I see a defect that some other people aren’t seeing. Should I get a different DE-10 Nano?

No, you should report the defect and wait for a fix. There is no magical best DE-10 Nano.

I found some other Cyclone based dev board on sale and it has built in WiFi. Can I use it for MiSTer?

Generally, no. While it’s always possible that someone will take time to port things to other boards, the different pins and memory will mean it won’t be a straight use and unless it’s a significant upgrade, it would never gain official support.

The DE-10 is rated for up to 100°C operation and it doesn’t get nearly that hot. Do I really need a fan or heatsink?

A number of complex cores (like ao486) benefit from having the chip at cooler temperatures, since heat can affect the tight timings they require. A fan is recommended to avoid any possible glitches, but you won't damage your DE10-Nano if you choose not to use one.

Videos

This page aims to centralize good tutorials and introduction videos.

MiSTer FPGA Overview

MiSTer is an evolving platform, so these videos can only represent what was available at the time:

- March 2022 - My Life in Gaming - MiSTer FPGA in 2022: A Primer Guide to Retro Gaming's Hardware Emulator **Very good in-depth full guide and review! Highly recommended!**
- Jan 2021 - Digital Foundry - DF Retro Hardware: MiSTer FPGA - A Brilliant Mini Emulation System Explored!
- Jan 2021 - Scarlet Sprites - MiSTer FPGA Review 2021: Arcade & Console Accuracy!
- Dec 2020 - Retro Bits - MiSTer FPGA - how to build, demo, costs, and pro/cons
- Mar 2020 - Briar Rabbit - What is a MiSTer FPGA?
- Nov 2019 - SmokeMonster - MiSTer Cores without add-ons (no SDRAM)
- Jul 2019 - RetroManCave - Exploring MiSTer
- May 2019 - GameSack - MiSTer Review
- Oct 2018 - ETA Prime - FPGA Emulation MisTer Project on the Terasic DE10-Nano
- Oct 2018 - SmokeMonster - Introducing the MiSTer FPGA

Tutorials

- MiSTer assembly and config (Ownlonymous)
- Creating a MiSTer SD card (NML32)
- Mounting a VHD from Windows over the network (NML32)
- MiSTer FPGA How To Setup Tutorial (MadLittlePixel)
- MiSTer FPGA Input Mapping (rsn8887)
- MiSTer FPGA Getting Started and Setting up with Mr. Fusion (Master Hacks)
- AO486 Core - Setting up DOS Games with pre-built Total DOS Launcher (FlynssBit)

FPGA Discussion

(mentioning MiSTer)

- Oct 2019 - Lon.TV - Lon & RetroRGB discuss FPGA
- Sep 2019 - SmokeMonster - FPGA Revolution

Other / Demonstrations

- Jul 2020 - RetroManCave - My Ultimate FPGA Desktop

Articles

This wiki page contains some links to articles that were written about the MiSTer FPGA project.

MiSTer FPGA Overview/Reviews

- Polygon - MiSTer 101: A classic gaming device to rule them all by Christopher Grant - August 8, 2021
- Kotaku - And Now, The Ultimate Retro Gaming Device by Mike Fahey - September 02, 2021
- Edge Magazine Australia - Hard Core: How MiSTer emulation is redefining the art of resurrecting gaming's past by Alex Wiltshire - April 22, 2021
- The Verge - The DIY Issue: Building The Ultimate Retro Computer by Sam Byford - March 11, 2021
- Nintendo Life - Hardware Review: MiSTer FPGA - A Tantalizing Glimpse Into The Future of Retro Gaming by Damien McFerran - February 11, 2021
- Dream Machine: MiSTer FPGA by Félix Léger (@felleg) - October 18, 2020

Tutorials/Guides

- MiSTer Manual by adreeve

What you will need

Requirements

In order to get started with the MiSTer platform, there are a few things that will be required. Optional addons are also available but for the bare minimum setup, you will need the following items:

- DE10-Nano Board with supplied power supply and SD card. (required)
- USB OTG connector or OTG USB hub(required)
- USB Keyboard (required)
- HDMI Monitor & HDMI Cable (required)
- Network Connection (recommended for initial setup and updates)
- Micro SD card reader (required for initial setup)
- Cooling (recommended)
- Upgraded power supply (recommended for users using the USB hub and lots of power hungry devices)

1. DE10-Nano board

The heart and engine of the whole platform is the **Terasic DE10-Nano** development board, made in Taiwan.

You can buy it:

- Directly from Terasic Inc.

Or from major electronics suppliers such as:

- Mouser
- Digikey

A power supply unit (PSU) and 8GB MicroSD card is included with the kit. The SD card can be reformatted to use with your MiSTer. Any MicroSD card 2GB and larger should work fine as well. Speed class doesn't affect it.

2. USB connection

Most unpowered basic USB hubs will work. Although it is recommended that you use a powered USB OTG hub with your DE10-Nano board as it is not able to handle high current consumption, so depending on the peripherals you use, you may need an external powered hub.

If you're not connecting too many USB peripherals, it does handle a keyboard, mouse, and gamepad just fine. Do note that the USB socket isn't very robust so it is best to avoid connecting and disconnecting too often.

USB option 1:

Micro USB OTG cable + basic USB 2.0 hub.

A basic USB 2.0 hub, or one with external power would be a good idea both to eliminate OTG socket reliability issues and provide power to external devices. Some cheap ones on eBay/Aliexpress may be declared as full speed USB 2.0 but in fact work in USB low speed mode. They may be acceptable for keyboard, but better to avoid them.

USB option 2:

USB OTG Hub.

These hubs are designed to connect directly to the micro-USB OTG port and require less inter-connection cables. Such hubs are also available on eBay/Aliexpress.

USB option 3:

USB hub daughter board.

You can assemble or purchase this board that provides 7 USB ports available to the MiSTer system.

3. Cooling

The hybrid ARM+FPGA chip has been found to get hot even when idling in the core menu, therefore some passive cooling is recommended. The main heat producer in the chip is the integrated dual-core ARM processor producing a constant heat regardless of the FPGA core in use.

The Cyclone V FPGA chip on the DE10-Nano board is industrial grade and supports up to 100°C, but for guaranteed long term usage without degrading its characteristics, it's highly advisable to add at least a heatsink.

This chip is approximately 21.5mm x 21.5mm. Ideal dimensions of a heatsink is 22mm x 22mm, and it will cover all the FPGA. Commonly found heatsinks with dimensions ranging from 20mm x 20mm to 25mm x 25mm can be used, but for larger heatsinks pay attention to nearby components, they must not touch the heatsink.

The height of the heatsink should be no more than 10mm if an I/O board is used because it could touch parts in the I/O board and create short circuits.

Active cooling

Some large cores such as ao486 and Minimig are sensitive to FPGA chip temperature and become unstable if it becomes too hot. So active cooling, in addition to passive cooling, is recommended for stability.

If you're building or purchasing an I/O board, they are designed for a 40mm x 40mm fan. Assembled I/O boards should already have a fan installed.

If you do not use any I/O boards then you are free to choose any fan, but bear in mind that you should only use the 5V line from the Terasic DE10-Nano board to drive a 5V fan. (The DE10-Nano's 9V line is deliberately hidden by the MiSTer I/O board to prevent any possibility of accidentally shorting it to any other signals.) You may also consider a larger 12V fan, they should work but spin slower and still provide good airflow. (Do note that popular Noctua fans may not spin up when undervolted).

A large selection of fans can be found on most electronic components sites, such as Digikey, Mouser and many others.

4. Upgraded Power Supply (recommended)

If you are going to use the official USB Hub add-on board with many peripherals attached, then there will be a much larger current draw required of the stock PSU than it was originally designed for. This could lead to instability and should be avoided. If you are using a usb hub add-on board, you should upgrade to a 4amp or greater power supply to provide sufficient current to both. When replacing the power supply make sure to get one that is relatively high quality, as the DE10-Nano could become damaged if you switch to a cheap badly-made power supply. Here's a Mean Well Power Supply Unit that is highly recommended:

Amazon: https://www.amazon.com/gp/product/B01D0Z8PLW/
Digikey: https://www.digikey.com/short/43cbh4
Mouser: https://www.mouser.com/ProductDetail/MEAN-WELL/GST25A05-P1J/?qs=drgMNd%252BkGPOX8brSXUPVtQ%3D%3D

This power supply would also require the appropriate female IEC 320 C13 power cable for your outlet that is appropriate for your region.

Setup Guide

MiSTer Setup Guide

This is an essential guide for your first time setup of the MiSTer system. It will guide you through the SD card installation, help you update the MiSTer system files, and shows you how to install an example console core and run a game.

There are two option for getting started, the recommended method is to use the Mr Fusion setup script. This is the best option for beginners and works on all platforms.

The second option is a manual install which is only recommended for users with specific use cases.

Mr Fusion Installation Method (Recommended)

Mr Fusion provides a compact image that you can download and flash onto an SD card of any size with a tool like Apple Pi Baker, balenaEtcher, Win32 Disk Imager, or even dd.

When you put this SD card into your DE10-nano and start it up, it will expand the card to its full capacity and install a basic MiSTer setup. This will be familiar to anyone who's worked with a Raspberry Pi before.

From there, using the built-in scripts, you can configure WiFi (or use ethernet out of the box) and run the standard MiSTer Updater script to get an up to date MiSTer installation.

You can optionally provide custom WiFi and Samba configuration which Mr. Fusion will install alongside the basic MiSTer setup.

Requirements

- A Micro SD card of minimum 2 GB, for example the one that came with your DE10-nano kit.
- Windows, Mac or Linux based computer with a (micro)SD card reader.
- An SD card flash utility.

Instructions

Step 1

Download the latest version from the releases page.

Step 2

Download and install an SD card flash utility for your system. Here are a few example in no particular order:

- Apple Pi Baker
- balenaEtcher
- Win32 Disk Imager

Refer to the documentation of the SD card flash utility for more information.

Step 3

Follow your SD card flash utility's instructions to flash the downloaded image onto your SD card.

Note: Extract the downloaded SD card image zip file if your SD card flash utility does not support flashing zip files!

Step 4

Put the SD card into the DE10-nano and power it on. (Be careful to pick the right SD card slot! Some pre-configured bundles have a secondary SD card slot on, e.g., the I/O board. Using the secondary slot will not work in this step. The SD slot on the DE10-nano is on the underside of that board.)

After a few seconds the orange LED on the board should light up. If you have a TV or monitor connected to the HDMI port, the screen will turn blue and then show an installation notice splash screen:

Mr. Fusion will automatically re-partition and resize your SD card and copy all the necessary MiSTer files onto it. When it's done it will reboot your DE10-nano and you will be greeted by the MiSTer menu.

Connect a keyboard to your DE10-nano and hit F12 to open the menu. Through the Scripts section you can configure WiFi and update your MiSTer.

Note: From powering on the DE10-nano and getting to the MiSTer menu should not take more than 90 seconds. If you don't see the MiSTer menu appear after two minutes, power off the DE10-nano, remove the SD card and start over.

MiSTer scripts support

The MiSTer Downloader tool is included by default in every MiSTer installation. It will install every new update for the system, including core updates. This image also includes the WiFi setup script to allow you to quickly setup a wireless internet connection after installation.

Adding more scripts

You can add more scripts if necessary: After you have flashed your SD card and before you move it over to the DE10-nano, re-insert it into your computer. A new drive called `MRFUSION` will appear. In it is a `Scripts` folder. Put any script you want to have available in your MiSTer in this folder. It will be copied to your MiSTer's Scripts folder automatically during the installation.

Custom WiFi configuration (optional)

You can copy a custom `wpa_supplicant.conf` file in the root of the SD card after flashing the Mr. Fusion image. It will automatically be copied to the correct place during the installation of MiSTer. This allows you to configure your WiFi credentials before you install MiSTer and thus removes the need to connect a keyboard after installation.

Custom Samba configuration (optional)

You can copy a custom `samba.sh` file in the root of the SD card after flashing the Mr. Fusion image. It will automatically be copied to the correct place during the installation of MiSTer. This allows you to enable Samba before you install and thus removes the need to connect a keyboard to your MiSTer or having to ssh into it.

Manual Installation Method

Requirements

You will need the following things to get everything started.

For the SD card setup:

- Windows 10 (recommended, older versions may work). For SD card creation under macOS and Linux, see this script.
- Internet connection.
- SD card reader.
- SD card with at least 2GB capacity.

And to run it:

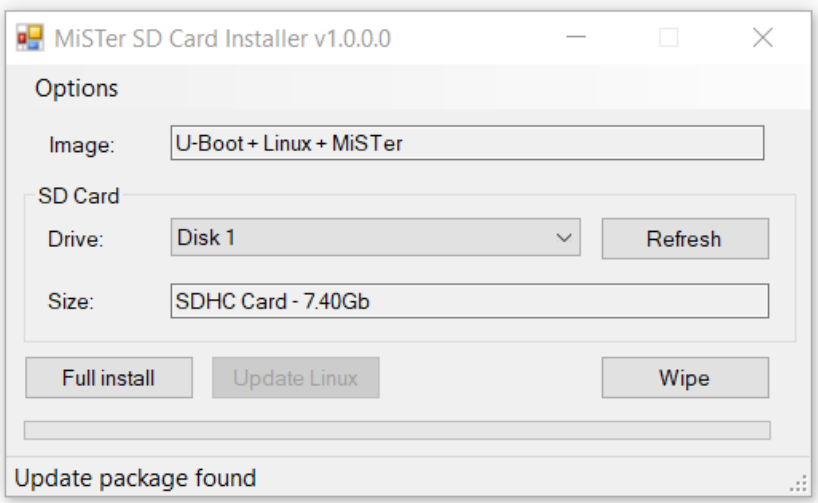
- DE10-Nano board + 5V power supply (supplied with the board).

- HDMI monitor + HDMI cable.
- USB-OTG (Micro USB) adapter + USB keyboard.
- SDRAM Board (Optional, but is required for a majority of the cores, see wiki page for instructions.)

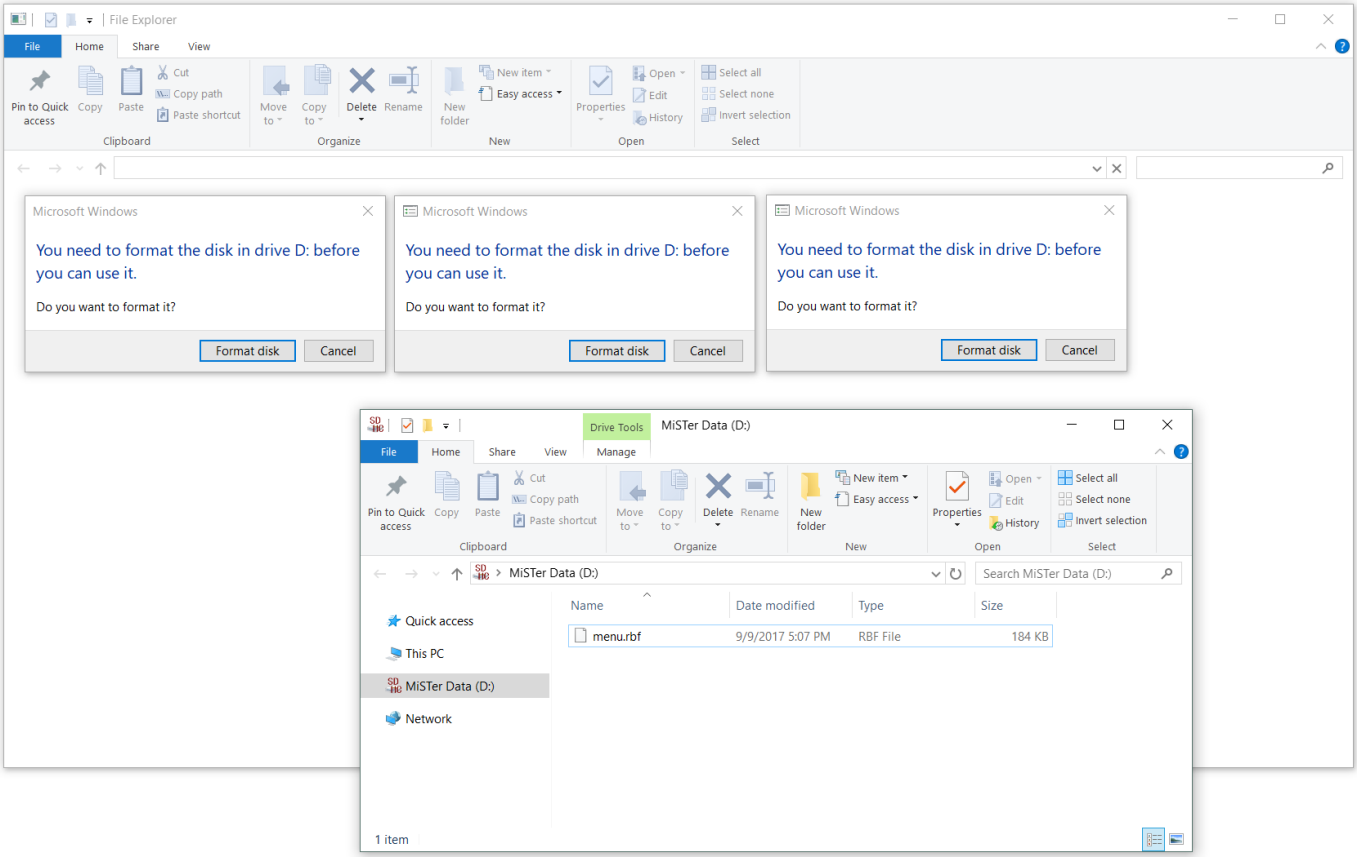
Do check the How to start and Input devices wiki pages for further information.

Prepare the SD Card

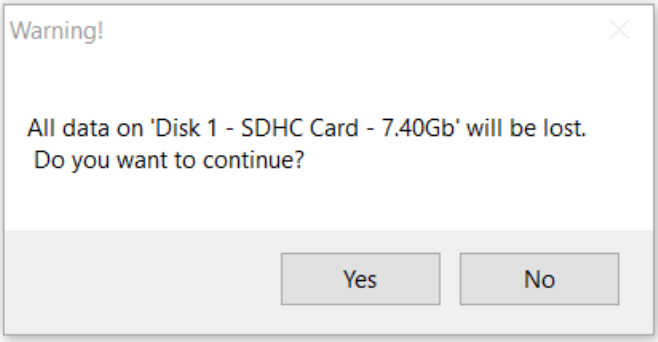
1. Download the latest SD card installer.
2. Insert your SD card into your card reader. All data on the SD card will be deleted! Make sure that the correct drive is selected, and if needed, backup the SD card.
3. Extract the `release_201#####.rar` file.
4. Start `MiSTer SD Card Utility.exe`



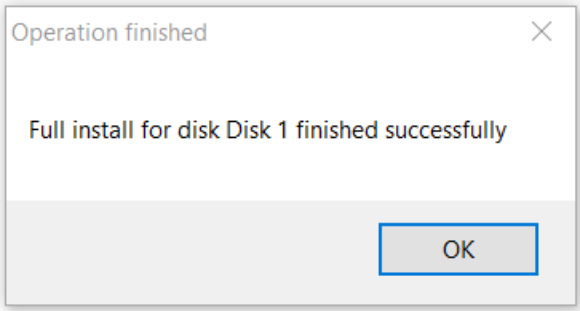
5. Make sure it says **"Boot + Files"** in the **"Image"** field.
    - Older versions of Mister SD card Utility (as pictured above) will say `U-Boot + Linux + MiSTer` in the `Image` field.
6. Select your SD card in the `Drive` field. If you have inserted the SD card after starting the Installer, hit the `Refresh` button and your SD card should appear.
7. The Installer may open multiple windows which will ask you to format the drive. If this happens, **don't format the drive!** Press `Cancel` in all windows.



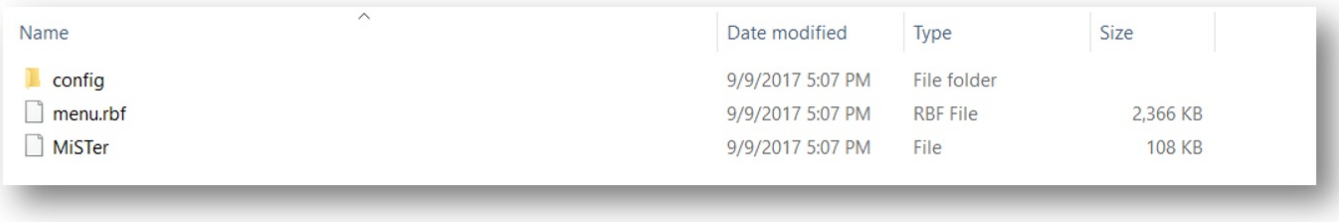
8. Press Full Install and confirm the following Warning with Yes. All data on the SD card will be deleted! If needed, make sure to backup the SD card before you execute this.



9. Confirm the successful installation with OK



10. The SD card file explorer window may be opened twice, if so, close one of them. The SD-card should contain the following files and folders:



If you see only the `menu.rbf` file, hit F5 on your keyboard or `right click > Refresh` to refresh the window. You should see all of them now. There may be other files and folders, but these are the essentials.

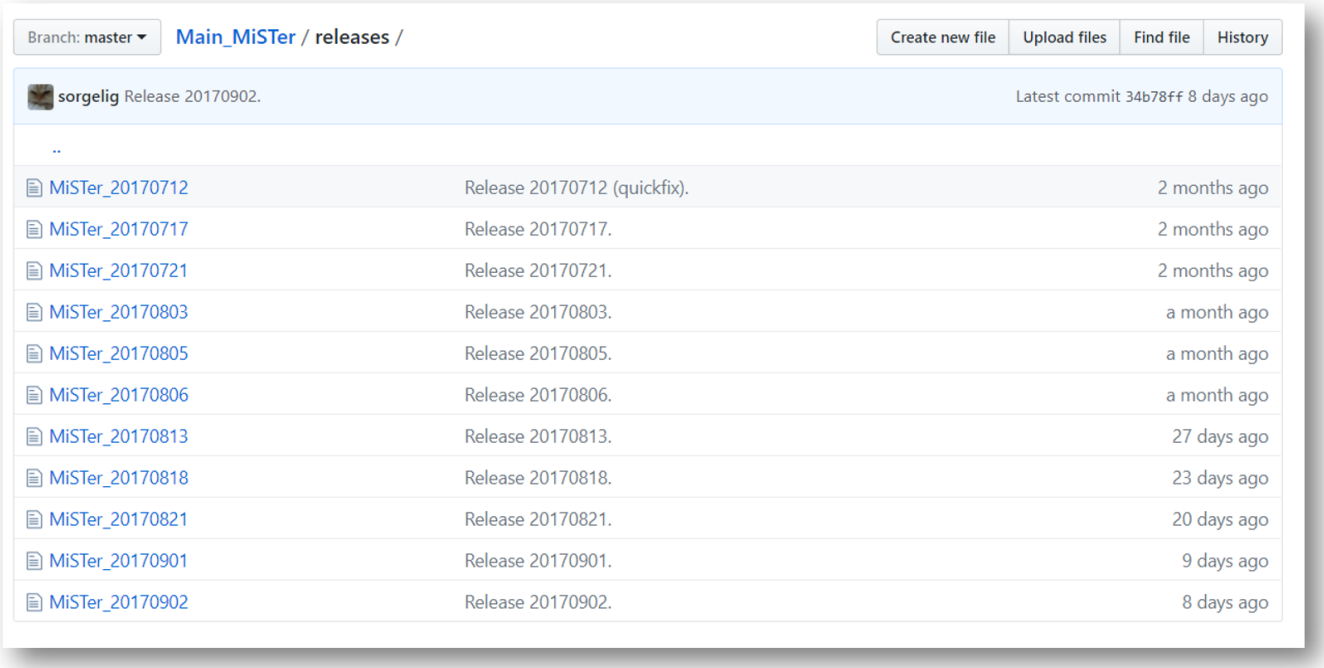
The files and folders you should see are:

- `linux` - Folder containing linux files
- `config` - The configuration folder where various config files are placed automatically. Those files usually don't need any manual modifications.
- This folder is no more created by newer version of SD Card Utility, but it will be created automatically by the MiSTer hardware at first run (you can manually create and populate it if you want)
- `menu.rbf` - This is the actual MiSTer menu core, which you will see when you boot up the DE10-Nano board (GitHub).
- `MiSTer` - MiSTer main firmware (GitHub)

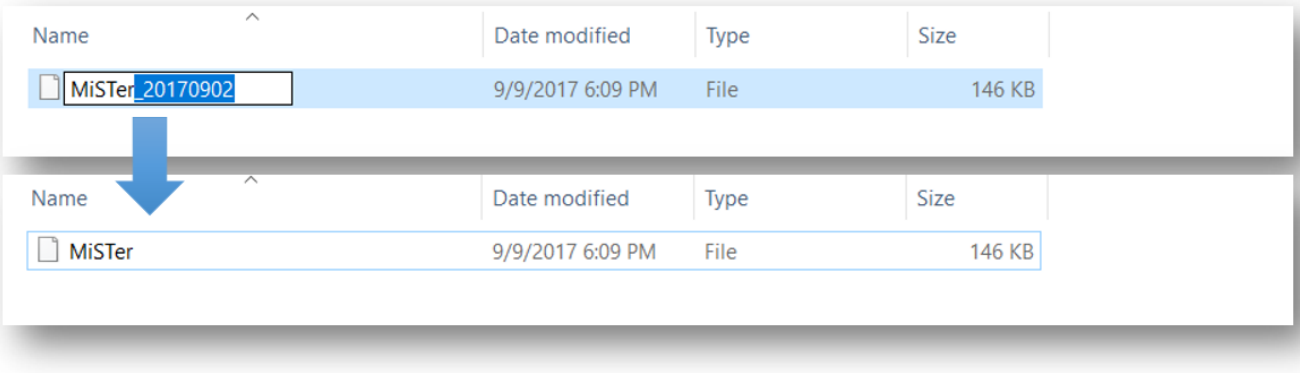
Update MiSTer files

The SD card installer will be older then the actual binary releases of the MiSTer firmware and the menu core. Therefore, we want to bring those files up to date.

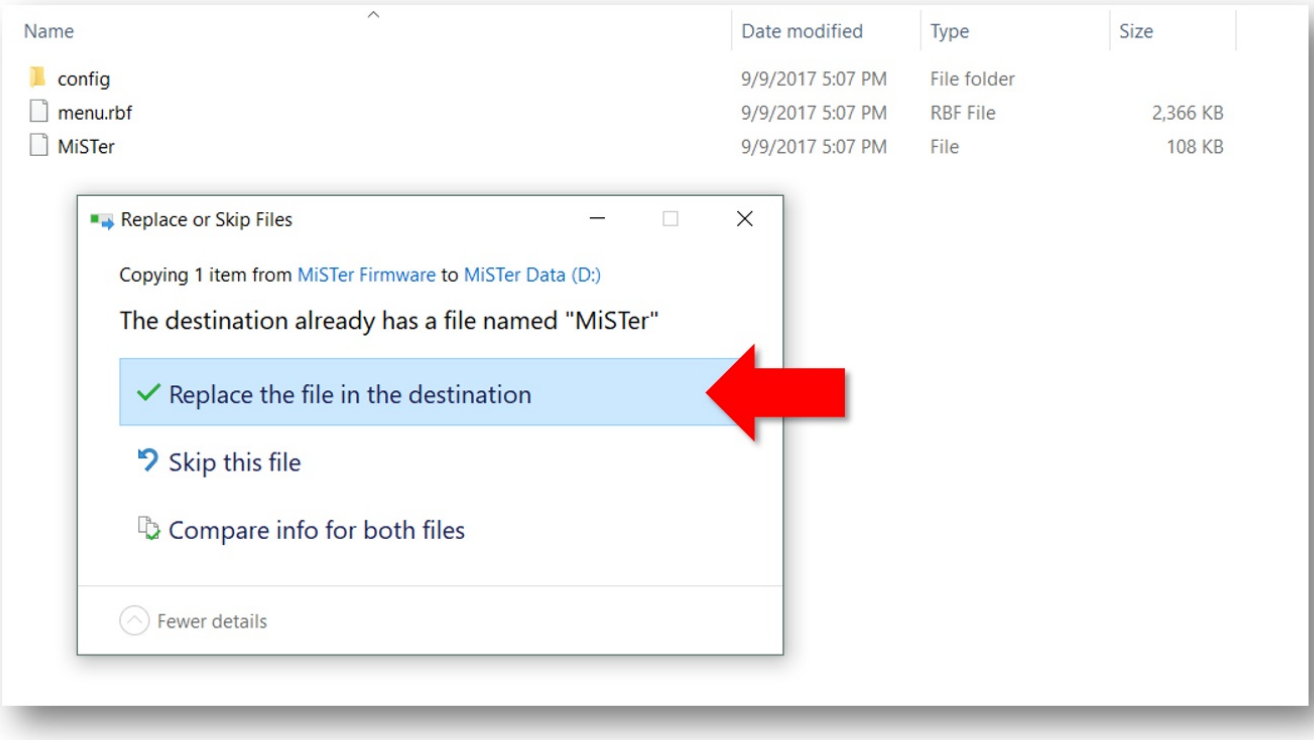
1. Go to the MiSTer-devel/Main_MiSTer Repository and download the most recent `MiSTer_202#####` firmware file on the bottom of the page.



2. Rename the `MiSTer_202#####` file to `MiSTer`



3. Copy the file over to your SD-card and override the old `MiSTer` file.



4. Repeat this for the menu core file. Go to the MiSTer-devel/Menu_MiSTer repository and download the most recent `menu_202#####.rbf` core file on the bottom of the page. Rename `menu_202#####.rbf` to `menu.rbf` and override the old file on the SD card.

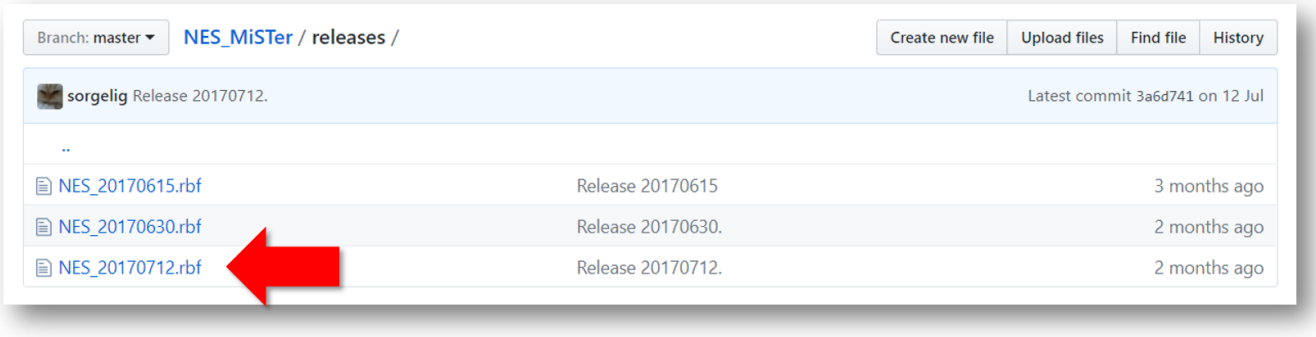
Get a core

We want to actually run a core like the NES or Genesis (Megadrive) console or Amiga computer on our DE10-Nano FPGA board. Therefore, we have to copy a core `.rbf` file to the root of the SD-card. The sidebar on the right contains a list of MiSTer compatible cores. Check out the GitHub repository page of each core for specific information.

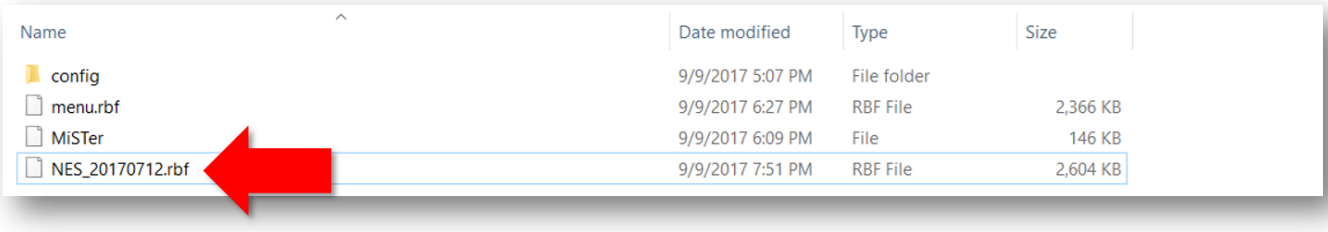
The following description is a generic example based on the NES core, but it is applicable to most other cores.

Do note that the NES core requires additional SDRAM. If you do not have this add-on, you can still continue following the example by using the Genesis core (which does not require additional SDRAM) instead of NES.

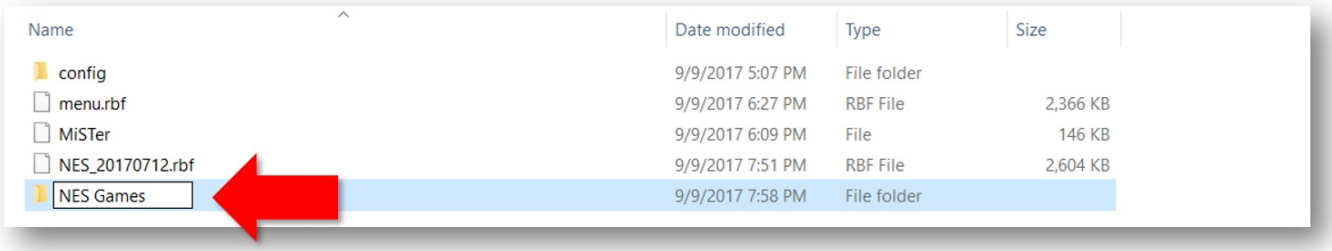
1. Click in the sidebar on Cores > **"NES"** or go directly through this link to the MiSTer-devel/NES_MiSTer release folder. Download the lates `NES_20######.rbf` core file



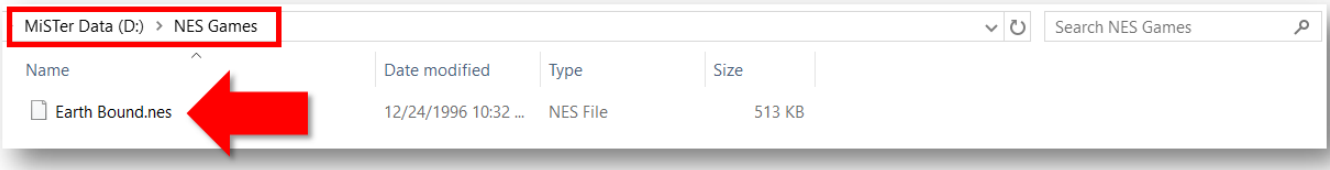
2. Copy the core file to the root of the SD Card. Leave the date in the filename. By this, you know which version you are actually using.



3. Create a new folder and name it for example `NES Games`.



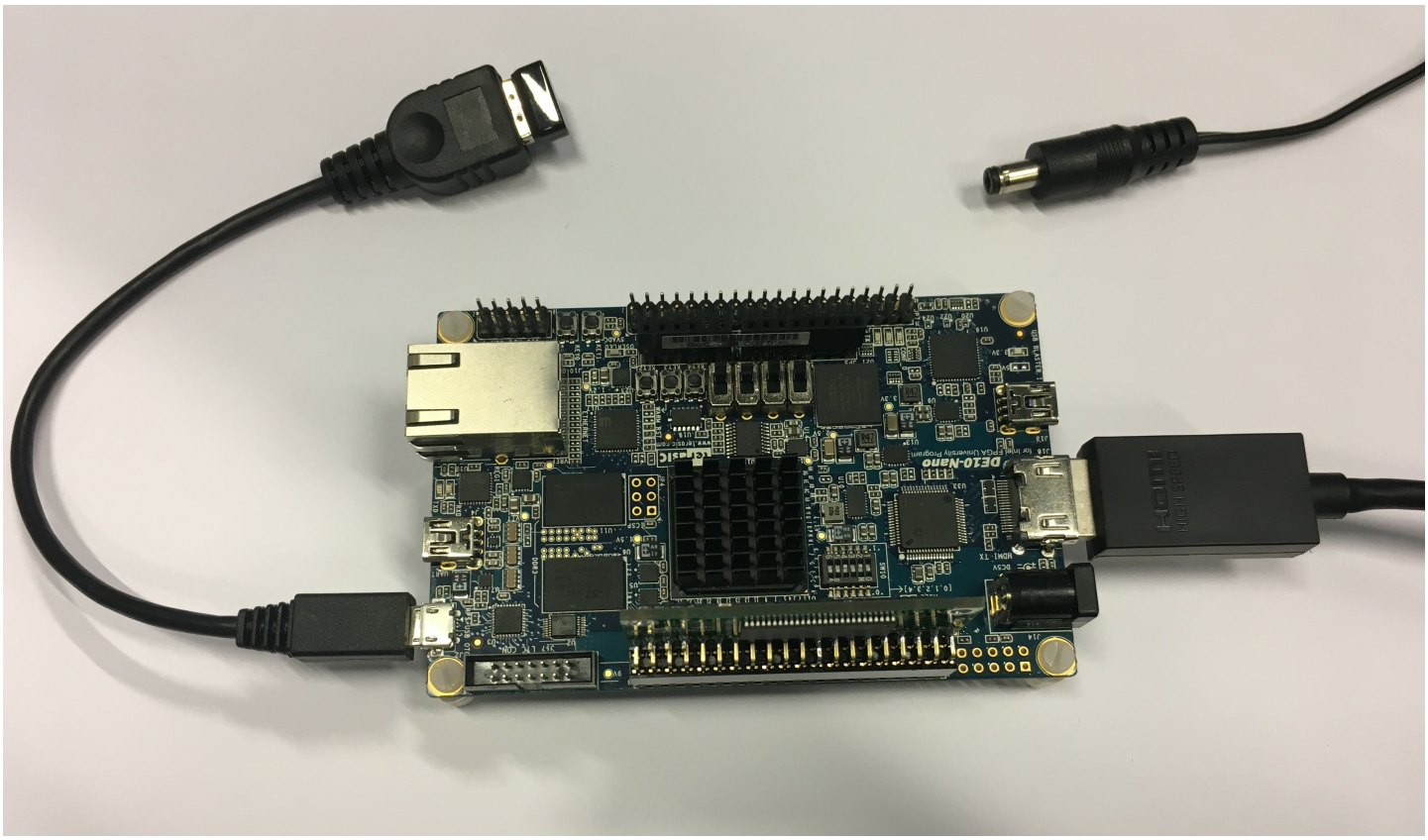
4. Download a `.nes` ROM (Game) file and copy it into your `NES Games` folder. You have to google that by yourself...



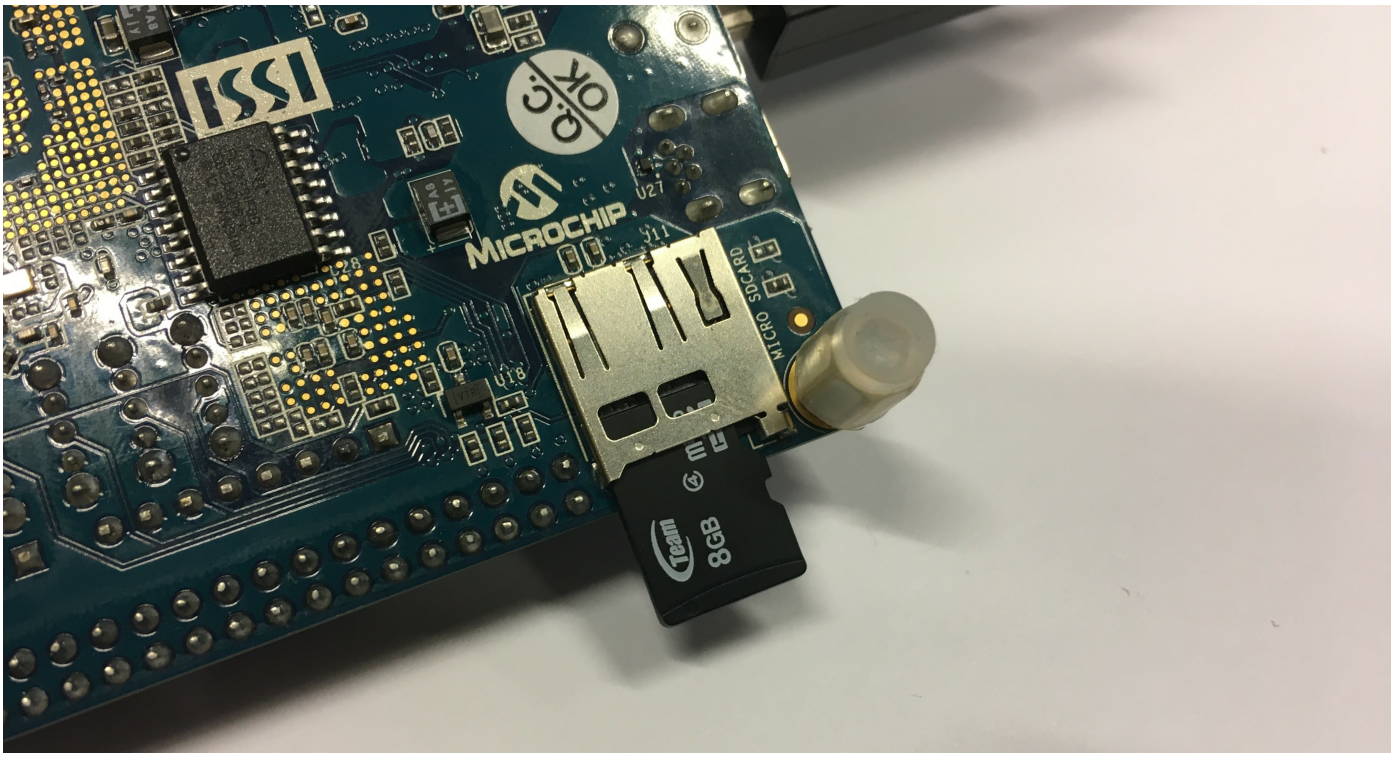
Fire it up!

1. If you're using additional SDRAM, make sure the SDRAM-Board is properly attached to the GPIO header JP1 of the DE10-Nano board.

Connect the DE10-Nano board via HDMI to a monitor and via USB-OTG adapter to a keyboard. Do not connect the power supply as yet.

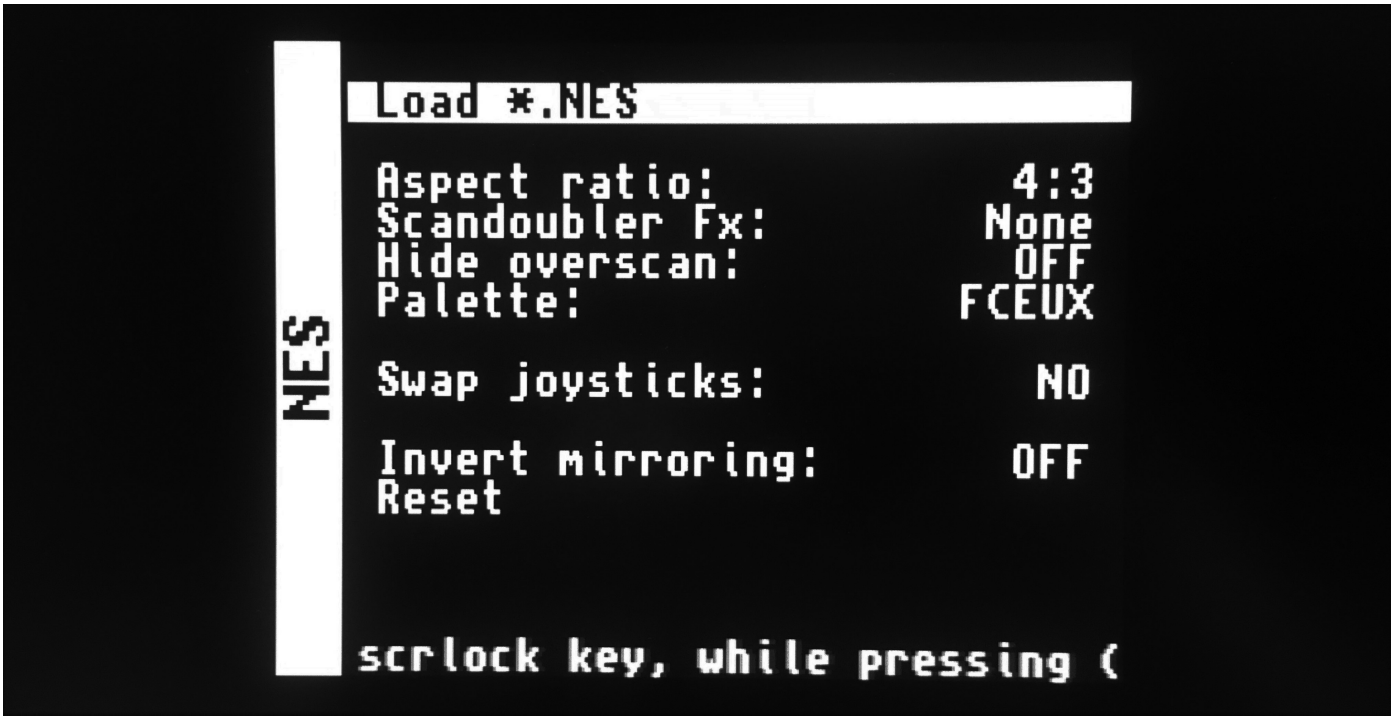


2. Remove the SD card from your PC and insert it in the DE10-Nano board.

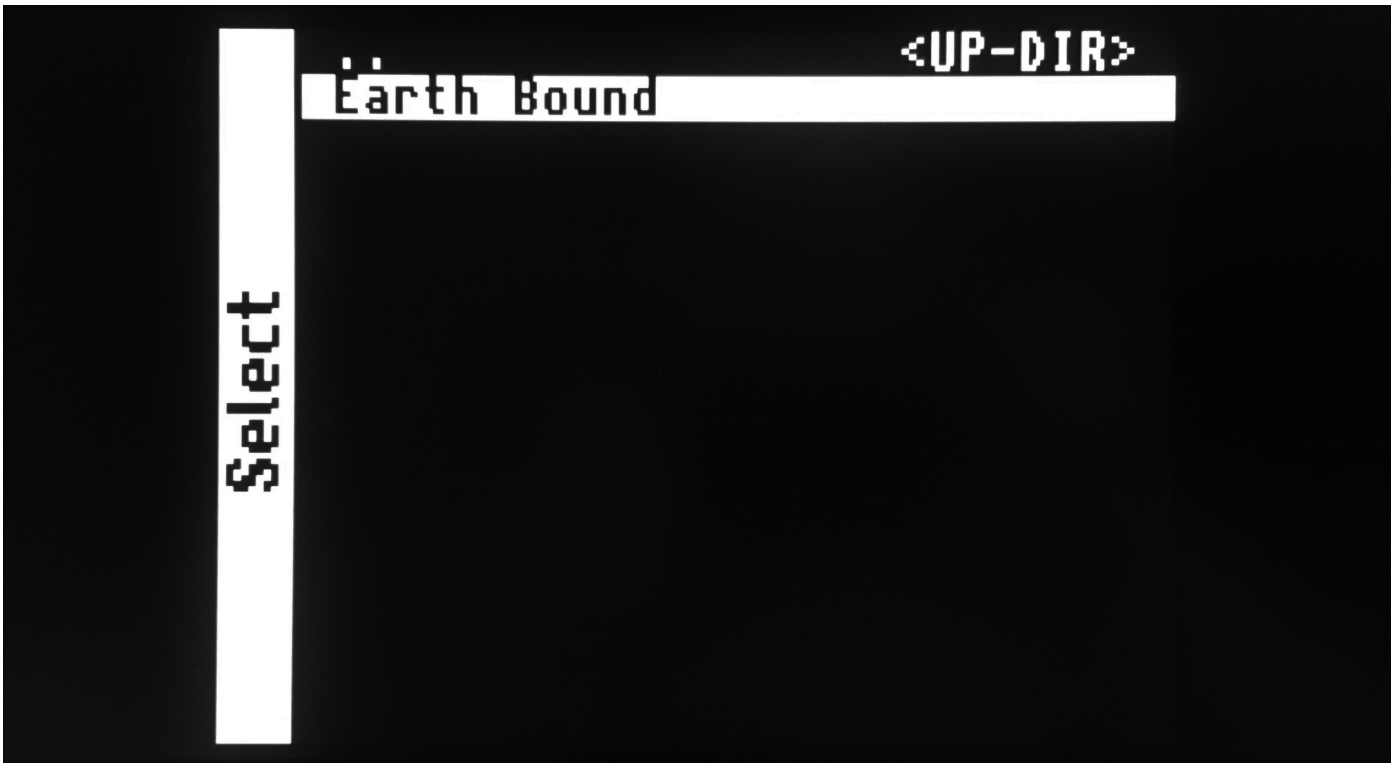


3. Connect the power supply. This will turn on the DE10-Nano board. You will see the MiSTer menu on the monitor. You see in the menu the NES (or Genesis) core we have copied to the SD Card. Hit the Enter key to start it.

4. You will see a black screen. This is normal because no ROM is loaded yet. **Press F12 to bring up the MiSTer menu.** In order to run a game, select "Load *.NES" and hit Enter.



5. This will bring up the SD card root directory. Navigate into your "NES Games" folder and select the ROM you want to start and hit Enter.



6. Congratulations, you have successfully started your first game on your new MiSTer!

Following steps

To get the most out of your MiSTer don't forget to (at least) check out the following pages:

- Configuration Files
- Video Filters
- Input devices

Additional notes

Once you've installed Release_20180115 or later, you can install future updates on MiSTer without removing the SD card. It's done in 2 stages: 1) Copy everything from **files** folder of release to /media/fat using FTP client and then reboot MiSTer (use Left Shift + Left CTRL + Left Alt + Right

Alt combination). 2) Log in via serial console or ssh and type `updateboot` then reboot again.

Usually the bootloader has little or no change and does not always require updating. But for a better experience it's advised to update the bootloader with every release. If by any chance a new version of Linux isn't able to boot with a previous bootloader, then simply use the SD card Installer Tool to update the bootloader (`Update Boot` button).

You may organize the cores into directories (folders) rather than have them stored on the root directory, to make these directories visible simply add an underscore in front of the directory name.

INI file

MiSTer.ini

The MiSTer.ini configuration file contains settings for the MiSTer.

A default copy of the file itself can be found here: MiSTer.ini

8 lines (7 sloc) | 505 Bytes — Raw | Blame | History

```
[MiSTer]
key_menu_as_rgui=0     ; set to 1 to make the MENU key map to RGUI in Minimig (e.g. for Right Amiga)
forced_scandoubler=0   ; set to 1 to run scandoubler on VGA output always (depends on core).
ypbpr=0                ; set to 1 for YPbPr on VGA output.
composite_sync=0       ; set to 1 for composite sync on HSync signal of VGA output.
vga_scaler=0           ; set to 1 to connect VGA to scaler output.
hdmi_audio_96k=0       ; set to 1 for 96khz/16bit HDMI audio (48khz/16bit otherwise)
```

Download and copy the `MiSTer.ini` file to the root of your SD-Card. Open it with your favorite editor (e.g. Notepad++) and change the parameters accordingly to the following description.

List of INI Settings

- bootcore
- bootcore_timeout
- composite_sync
- controller_info
- custom_aspect_ratio
- direct_video
- forced_scandoubler
- hdmi_limited
- hdmi_audio_96k
- jammasd_vid, jammasd_pid
- key_menu_as_rgui
- osd_rotate
- recents
- vfilter_default
- vga_scaler
- vscale_mode
- vsync_adjust
- ypbpr

Addtional Information

- Adding Core-specific Settings
- Switching INI Files On the Fly

bootcore

If specified, selects core to run at startup (instead of menu) `bootcore=lastcore` Automatically loads the last used core `bootcore=lastexactcore` Automatically loads the last used core, matching time stamp e.g. core_yyymmdd.rbf `bootcore=[corename]` Loads [corename]_*.rbf at startup (first file found on SD card) `bootcore=[corename_yyyymmdd.rbf]` Loads [corename_yyyymmdd].rbf at startup (exact file name)

You can still select `Reboot` in the system menu to get back to the main menu core.

bootcore_timeout

Number of seconds to wait before auto core boot. You can comment out this line by putting `;` at the beginning to boot directly into the core with no timeout.

`bootcore=10` Value can be 10 to 30 seconds

composite_sync

Use composite sync as horizontal sync signal on VGA output.

`composite_sync=1` activate composite sync

`composite_sync=0` deactivate composite sync

direct_video

Feature for using the HDMI port with DACs to produce analog video. See this page for more information.

`direct_video=1` to activate it (disables compatibility with HDMI TVs and monitors).

`direct_video=0` to deactivate it.

custom_aspect_ratio

You can set up to two additional custom aspect ratios (ie. 8:7, 10:7, 64:49, 204:137), which will appear in the `Audio & Video` menu of any core.

It is recommended to set these on a core-by-core basis in your config file, as you may want different options depending on what core you're playing.

Note: This is the aspect ratio of the active/visible image, NOT pixel aspect ratio.

```
custom_aspect_ratio_1=8:7
custom_aspect_ratio_2=64:49
```

controller_info

Seconds to display controller settings when starting a new core.

Cores support automated mapping from central joystick mapping if no core-specific joystick mapping was defined. When this is active, MiSTer shows a tiny popup displaying the button assignment. This setting controls the time that pop-up is displayed (and can switch it off)

`controller_info=0` Do not display mapping info pop-up

`controller_info=1` , `controller_info=10` Seconds to display controller mapping pop up

forced_scandoubler

Run scandoubler on VGA output always (depends on core).

Most modern monitors won't support the 15 KHz horizontal sync output signal through the VGA connector. This option doubles the frequency of horizontal sync signal and brings it in a compatible range for modern monitors. Note that this is not a global option and is only valid for the MiSTer Menu. Each core drives the VGA output itself and requires its own setting. Check the core menu of the corresponding core via HDMI to set the scandoubler option if available / necessary.

`forced_scandoubler=1` activate scandoubler

`forced_scandoubler=0` deactivate scandoubler

hdmi_audio_96k

HDMI audio output options

`hdmi_audio_96k=1` 96khz/16bit HDMI audio

`hdmi_audio_96k=0` 48khz/16bit HDMI audio

hdmi_limited

Change between Full Range RGB and Limited Range RGB

`hdmi_limited=0` Full Range RGB (0-255)

`hdmi_limited=1` Limited Range RGB (16-235)

`hdmi_limited=2` Limited Range common DAC variant (16-255)

jammasd_vid, jammasd_pid

USB vendor ID and product ID for JammaSD adapter, required for keypress-to-joystick translation.

Allows a JammaSD joystick to be read as a standard joystick rather than being read as a keyboard.

`jammasd_vid` Vendor ID

`jammasd_pid` Product ID

key_menu_as_rgui

Makes the MENU key map to RGUI in Minimig (e.g. for Right Amiga)

`key_menu_as_rgui=1` set the MENU key map to RGUI

`key_menu_as_rgui=0` dont set the MENU key map to RGUI

osd_rotate

Display OSD menu rotated.

`osd_rotate=0` no rotation

`osd_rotate=1` rotate right (+90°)

`osd_rotate=2` rotate left (-90°)

recents

Set to 1 to enable showing recently played games. Once enabled, you can highlight or select the "Load file" option in a core menu and press the `select` button on your controller to show a list of recently loaded files. In the core selection menu, pressing `select` will show the recently loaded cores. Files and cores that have been moved or renamed since they were last loaded will appear faded out in the recents list.

(Note: using this mode increases writes to SD card, which may increase wear over the long term.)

`recents=0` Default behavior - don't show recent files

`recents=1` Show recent files

vfilter_default

This sets a default filter for all cores if a video filter was not selected in the core settings.

`vfilter_default=default.txt` Use `/media/fat/Filters/default.txt` as default filter

vga_scaler

This option makes the VGA (DB15) connector output of the scaler. In other words, it makes the VGA have the same resolution as HDMI (1080p or 720p, or as per your overall video settings).

`vga_scaler=1` VGA DB15 connector will have the full scaler output

`vga_scaler=0` VGA DB15 will have an independent video output, separate from main scaler (e.g. 240p or 480p)

video_info

Seconds to display video information on startup. Defaults to zero.

`video_info=0` Do not display video info `video_info=1`, `video_info=10` Specify number of seconds to show video info

vscale_mode

Options for integer scaling.

`vscale_mode=0` scale to fit the screen height. Cores that allow for 5x vertical crop mode usually require this vscale mode in order to enable the feature. Ideal for fullscreen 1080p.

`vscale_mode=1` use integer scale only.

`vscale_mode=2` use 0.5 steps of scale.

`vscale_mode=3` use 0.25 steps of scale.

vsync_adjust

Sets the vsync buffer mode for HDMI output. This setting does not affect direct video or analog output from the IO board.

Some HDMI displays can accept somewhat non-standard signals, allowing for lower display latency with MiSTer. It is recommended that you start with a setting of 0, and then try modes 1 and 2 to see if they work with your display or capture device. Different cores will have different results.

`vsync_adjust=0` Default. Buffered 60hz HDMI video output, compatible with most HDMI devices.

`vsync_adjust=1` Adjust output HDMI Vsync to match original Vsync. Lower latency than mode 0, but less compatible.

`vsync_adjust=2` Low-latency mode, using the system's native pixel clock. This mode has the lowest latency, but it's the least-compatible.

ypbpr

Use YPbPr signal on VGA output.

`ypbpr=1` activate YPbPr

`ypbpr=0` deactivate YPbPr

Adding Core-specific Settings

It is possible to specify different settings for different cores; for example, you may prefer to use integer scaling just for the Game Boy Advance core so that you don't need any video filters for smooth scrolling. Simply add a section at the end of the INI file with the core name in brackets and paste your different settings below there, like so:

```
[GBA]
vscale_mode=1
```

Or, suppose you wanted to add an option to select 8:7 aspect ratio for the SNES core:

```
[SNES]
custom_aspect_ratio_1=8:7
```

The Menu core can have its own settings too. Create a section named `[Menu]` to specify settings for it.

Switching INI Files On the Fly

MiSTer currently supports up to 3 additional INI files that can be toggled in the OSD menu, either by going to `Misc. Options` (press `left` while in the menu) or by holding the menu `back` button on your controller and pressing a direction. This is useful if you need to switch between video configurations often. To get started, make copies of your INI file and rename them:

`mister_alt_1.ini` activated by `back + left`

`mister_alt_2.ini` activated by `back + up`

`mister_alt_3.ini` activated by `back + down`

Additionally, you can switch back to your default `mister.ini` by pressing `back + right` or by selecting `Main`.

Your alt INI file will stay loaded across reboots and core changes until you turn the power off; `mister.ini` will always load by default when powering on. Note that if you switch INI files while a core is running, the core will reset.

Using MiSTer

So, you've got your MiSTer and followed the Setup Guide, now what?

Let's get you more familiar with what you can do!

Important Hotkeys to Remember

- F12 - Brings up the Menu in any Core.
- ALT + F12 - Brings up the Select Core Menu.
- F1 - Changes the background.
- F11 - Bluetooth pairing menu (for supported BT adapters)
- F9 - Linux Terminal/Command Line interface (Press F12 to switch back to the MiSTer menu)
- Left Shift + Left Ctrl + Left Alt + Right Alt - Reboot
- Windows + Print Screen - Take a Screenshot (automatically stored in `/screenshots/`)

Getting Started

You'll need a USB keyboard to start and a wired internet connection is highly recommended.

Plug in the USB keyboard and your internet cable, and then power on the MiSTer. The red, orange and green led lights should start pulsing, and you should see the MiSTer menu onscreen after a second or two.

Downloading the Updater Script - update.sh

If you used the recommended setup method for your MicroSD from the Setup Guide, then your MicroSD already has the update.sh script in the `/scripts/` folder. That means you can ignore this section.

If you didn't use the recommended method of installing to your MicroSD then we need to download the script from Github:

1. Press F9 to bring up the Linux prompt.
2. Type "root" for the username and press enter, and then type "1" for the password and press enter.
3. Type `cd /media/fat/Scripts/` and press to go into the Scripts directory. If you get a message that says the directory is not found, then make the directory with `mkdir /media/fat/Scripts/` to get around this, and then do the first line again.
4. Download the update script with `wget --no-check-certificate https://raw.githubusercontent.com/MiSTer-devel/Updater_script_MiSTer/master/update.sh -O update.sh` and press enter.
5. Once it's complete, type `exit` and press enter. If the menu and interface you saw when you first booted your MiSTer doesn't come up, then press F12.

That will download the updater for you in the correct folder. We only need to download that script once. Once its installed, you can simply run the updater from the Scripts menu. We can now update the system from within MiSTer.

Tip - Press F12 to bring up the System Menu (if you press F12 again, it will show the core menu). Use the up and down arrow keys and enter to navigate the menu.

Fixing missing certs

If you are unable to wget as instructed above, this might be because you are missing security certificate files. The default system comes with no security certificate files, which is a bit annoying, as you need to add --no-check-certificate on wget to download anything HTTPS. Lets fix that:

1. Open the linux terminal/command prompt with F9, use `root` as your username and `1` as your password.
2. Type `cd /etc/ssl/certs` and press enter.
3. Type `wget --no-check-certificate https://curl.haxx.se/ca/cacert.pem` and press enter

Assuming it downloaded correctly, you can *now* use wget as nature intended!

Updating our system

Simply press F12 once or twice, until you see an option that says "Scripts" and select it. There will be a warning, make sure you read and understand this. Select "yes" and press enter to continue, and select "update" and press enter to begin your first update. This will download all of the latest releases of the official cores, scripts, and other files so you can start using your MiSTer.

MiSTer cores are regularly updated - sometimes daily, so run "update" regularly!

Ok, you're updated, now what?

File Transfer Setup Summary

Now you need to setup your MiSTer to receive files. Why, you may ask? You don't want to have to remove and put in the MicroSD every time you want to send ROMs to it do you? That can wear out the DE-10 Nano's MicroSD slot.

There are multiple methods of transferring files to the MiSTer. FTP can be turned on with the FTP script, so the MiSTer can serve as an FTP server waiting for an FTP client like FileZilla to connect to it. If SSH is turned on (using the ssh script that the updater should have downloaded), you can use a file transfer client capable of sending files via the SCP protocol like WinSCP. If you turn on your MiSTer's Samba Server, then you can open up the MiSTer's storage directly from Windows File Explorer by typing `\\mister` in the address bar and pressing enter.

Important note: When you are connecting via SSH or using the linux terminal from the MiSTer itself, the MiSTer "root" directory that is often brought up in this wiki is located in `/media/fat/`. So if the wiki refers to the `/games/` folder, that means on the MiSTer that is located in `/media/fat/games/`. If you are reading the MicroSD on your Windows PC with an adapter, or are connecting via SAMBA, then you will start in this root MiSTer folder `/media/fat` and not be able to go up any levels. This is normal.

Transferring ROMs

Each MiSTer core has a different location that it looks to for roms by default, which resides in the `/games/` folder. These are usually explained in the respective core's README file in the github repo. They typically are named after the core, such as:

`/games/Genesis/`
`/games/MegaCD/`
`/games/C64/`

You'll want to start copying the appropriate file backups of your cartridges/discs/cd's - i.e. roms to the appropriate locations for the core.

Some cores require boot.rom's and bios.roms, and other kinds of prerequisite files placed in the `/games/$CORE/...` folder for that core. For instance, the MegaCD core requires a Sega CD BIOS to work, and it would be placed here:

`/games/MegaCD/cd_bios.rom`

Configure Inputs

If you are using a usb gamepad or something similar, you will need to configure your input mapping. Press F12 once or twice until you see an option to `Define joystick buttons`, and select that option. Follow the on-screen prompts, skip anything that isn't relevant to your particular controller with the spacebar or the addon board's user button. If you make a mistake, don't worry, you can start over from the beginning.

For more details on how to configure your joysticks: https://github.com/MiSTer-devel/Main_MiSTer/wiki/Main-Joystick-Mapping

Play The Game!

Now you are likely ready to try and play a game. Use the main menu to navigate to the core you have already transferred the necessary roms and bios files to the core's games folder, and select the game you want to play, and enjoy your new FPGA-based gaming experience! :)

Core Status

Core Status

Cores on MiSTer are the result of the collaboration between many people and extensive testing, sometimes over many years and prior open-source projects. Most console and computer cores have now been compared to original hardware to a high level of precision (e.g. audio capture comparison), with the few issues remaining documented on their respective github repository pages. Please refer to these pages, per core, before submitting any issues.

The core links on the sidebar will bring you to the release folder on github, which will also allow you to find the **Issues page** and the **readme files**, listing all available features and limitations in detail per core.

Core Paths

Standard Core Paths

The standard paths are of the form `/media/fat/games/<CORE>`, where `<CORE>` is the path of the given system.

Please check the respective README file for each core to determine the appropriate full path.

Standard USB Core Paths

They are of the form `/media/usb<0..5>/games/<CORE>`. Where `<0..5>` indicates number of the USB drive mounted by the operating system.

Standard CIFS Core Paths

They are of the form `/media/fat/cifs/games/<CORE>`.

Other Paths

There are other valid paths, that are available for backwards-compatibility reasons, and also to ease testing. You should use the standard paths instead whenever possible.

Path Priority

There is a priority order of core paths. When you plug in a USB drive and it has a folder `/games/PSX` on it (mounted locally as `/media/usb<0..5>/games/PSX` when plugged in), then the MiSTer PSX core will look to that folder on the USB drive instead of the local one on the MicroSD at `/media/fat/games/PSX`. Here is the priority list from Main_MiSTer's file_io.cpp in order of highest priority to lowest:

1. `/media/fat`
2. `/media/usb<0..5>`
3. `/media/usb<0..5>/games`
4. `/media/fat/cifs`
5. `/media/fat/cifs/games`
6. `/media/fat/games`

If the core's folder isn't found in any of these it should create the folder.

Why FPGA?

A typical potential user will eventually ask, "Why do you need to use FPGA while other proven solutions exist, such as Raspberry Pi?"

There are debates about how to refer to the process of simulating real hardware using FPGA. Some people insist it's not emulation but rather true hardware *replication*, while any simulation using a traditional CPU should be referred to as emulation. I have my own opinion here. :) From my point of view, if the FPGA code is based on the circuitry of real hardware (along with the usual tweaks for FPGA compatibility), then it should be called replication. Anything else is emulation, since it uses different kinds of approximation to meet the same objectives. Currently, it's hard to find a core that can truly be called a replica – most cores are based on more-or-less functional recreations rather than true circuit recreation. The most widely used CPU cores – the Z80 (T80) and MC68000 (TG68K) – are pure functional emulations, not replications. So it's okay to call FPGA cores emulators, unless they are proven to be replicas.

To go back to the original question, then, why FPGA, if it's also just emulation? Well, FPGA emulation is fundamentally different than emulation on a CPU. Traditional emulators on CPUs execute code sequentially. This is a tricky method of emulation because real hardware has many chips and all of them work in parallel. The CPU, video chip/logic, audio chip, memory arbiter – all of them are working at the same time. So a traditional emulator has to take care of all these parts and try to emulate the whole orchestra at the same time by quickly "running" from one chip to another. This requires a lot of CPU power to emulate even an old and slow retro computer. Sometimes even a modern CPU working at 100 times the speed of the retro computer is not enough, so the emulator has to use approximation, skip emulation of some less important parts, or assume some standard work of the emulated system without extraordinary usage. Let's take a well-known emulator, UAE, emulating an Amiga. On a Raspberry Pi 3, you can run some Amiga CPU benchmarks and get crazy numbers like 100 times the original 68000 processor. So you may assume you have an emulated Amiga that is 100 times faster than real one. No, you don't. If you run different kinds of demos or games, you will see the video stutters sometimes. For example, if you play the well-known "State of The Art" demo by Spaceballs, you will notice video stuttering at some points, while a real Amiga 600 with 1x CPU speed plays the whole demo very smoothly. This is how traditional emulators on Raspberry Pi work.

FPGA emulation works very differently from traditional emulation on CPU. An FPGA is a large array of simple triggers and other logic – just like any other chip/CPU. The only difference is that specific chips/CPUs have these triggers and logic permanently connected, while FPGA allows you to connect them however you want. A special HDL (hardware description language) describes how to connect all these triggers/logic cells. Everything in FPGA works in parallel like in the original chips/devices. Thus, FPGA is pretty close to the original hardware. FPGA doesn't need high frequencies to emulate retro computers; it works at much lower frequencies than traditional emulators require. Since everything in FPGA works in parallel, it is no problem to handle any possible usage of the emulated system. Developers using FPGA usually concentrate on the specific part to make it work correctly – and it will work as it should in any possible scenario. In the same reference demo, "State Of the Art," using FPGA emulation, you can see smooth video through the whole playback, as on the original hardware.

You may want to ask, "So why not make all emulators on FPGA then?" The answer: FPGA programming is not so trivial. Every bit in FPGA works in parallel, so the developer needs to think in parallel as well :). What is trivial on CPU is not trivial on FPGA – although some parts that are trivial on FPGA cost a lot in CPU code.

What about Lag?

A common concern among retro enthusiasts is whether a device of this sort has *lag* and whether it will create a less desirable experience compared to original hardware.

Every electronic equipment exhibits some kind of *latency*, but it only becomes problematic if this latency causes frames to be missed. A frame is typically 16ms for a system using a 60 frames per second (60 Hz) display.

Lag is only problematic in a few specific cases:

- Some older games were designed to rely on extremely quick response times (e.g. Punch-Out on NES).
- If the latency is different between two players, it could introduce an unfair advantage.
- Lag that isn't constant can be an issue on games that require precise movement. (Most players can adapt to lag that is consistent.)

There are three major categories of lag. **Input**, which involves controllers, mouse, etc, **Processing**, which would involve buffering in the core, execution or delays in code and **Display** which involves the output of the video to your display device.

For a more detailed overview of lag and exploration of it, please refer to this page: https://inputlag.science/

Input

For input, MiSTer primarily uses USB. In this case the overall input lag is the sum of the lag caused by the USB polling and the lag caused by the controller itself, how quickly it processes the signals. The latter is outside the scope of MiSTer and can only be improved by using a better controller. On the MiSTer side only the lag caused by the USB polling can be reduced if the connected USB device supports a lower polling interval. The polling interval is measured in Hz and indicates how often a USB device is polled per second. At 1000 Hz, a USB device is polled 1000 times per second, which means the additional lag caused by the polling is 1 s / 1000 = 1 ms in the worst case and 0.5 ms on average. This is a great improvement compared to the default value of 125 Hz, where a USB device is polled 125 times per second, which means the additional lag caused by the polling is 1 s / 125 = 8 ms in the worst case and 4 ms on average.

If a game is rendered at 60 frames per second, a single frame takes 1 s / 60 ≈ 16 ms to process. One might think that any polling interval below 16 ms would be perfect; however, USB polling happens independently of the vertical sync. Therefore, even with a 1 ms polling interval, there is a chance of 1/16 that the whole frame will be missed and input will be processed on the next frame. The longer the polling interval, the greater the odds of a missed frame. With a polling interval of 8 ms, the odds of input missing a frame are 50%. Native polling rates and input response vary across consoles and even games.

Processing

This is one core advantage of emulation using FPGAs. Unlike software emulators which go through a cycle of executing, and then waiting for a screen refresh, FPGA cores run in real time, as the original hardware did. This means that cores don’t have CPU bottlenecks to slow them down arbitrarily or require additional large buffers to hold data under most circumstances.

Display

MiSTer’s two primary display outputs are analog and HDMI.

The analog output is driven as the original system would have, with no buffering, and so it will be effectively identical to the latency of a real console. From this point of view, the analog output cannot have any form of lag.

When using HDMI output the image must be scaled up to fit the higher resolutions, which requires additional processing. The MiSTer scaler has options which impact its latency. Using `vsync_adjust=2` in the ini file will result in about 4 scanlines of latency, while 0 or 1 will result in up to roughly 2 frames of latency, with the added advantage of being more compatible with displays.

In addition your own television or monitor may introduce more latency, but this varies by device and no definite number can be given on that here.

In summary, if lag is critical to you, **it's best to play on a CRT using a recommended and widely-tested USB controller.** Some users have tested and ranked USB controllers by performance; you can see their results here. An alternate thorough list of tested controllers by misteraddons is also available here

Do keep in mind, however, that even over HDMI MiSTer is capable of providing a better experience than many other devices.

Reducing Lag

Video lag

MiSTer offers options in how to configure its HDMI upscaler, making a tradeoff between compatibility and low latency. These can be set in the MiSTER.INI file at the root of the SD card:

- `vsync_adjust=2` is the best option if it is compatible with your TV. This mode uses the original refresh rate and pixel clock of the core, resulting in no additional latency.
- `vsync_adjust=1` is the second best option, but it adds up to 2 frames of latency. This mode uses a framebuffer but maintains the system's original vsync and varies the pixel clock per core.
- `vsync_adjust=0` is the lesser option, but the most compatible. Up to 2 frames of latency and less smooth scrolling. This mode guarantees 60 hz output with an NTSC standard pixel clock.

Input lag

USB controllers usually have an interval value which the host (MiSTer Linux kernel) respects to poll their inputs at. Most USB devices can actually perform better by being polled more often without any side effects.

To set a higher USB polling rate, you need to go to the "linux" subdirectory on your SD card and rename "u-boot.txt_example" to "u-boot.txt". The aforementioned file contains the following options, which should only be changed if you are encountering problems:

```
v=loglevel=4 usbhid.jspoll=1 xpad.cpoll=1
```

loglevel: 4 is the default value. You can set this to 9 to get debugging messages with dmesg command via SSH. If you just want to know which values of usbhid.jspoll and xpad.cpoll are applied to your controller, there are easier ways to achieve this (see below).

usbhid.jspoll: specifies the interval for USB HID controllers, usually DirectInput.

- 0 is the default value MiSTer uses (even when there is no "u_boot.txt"). In this case the requested value from the controller is used.
- 1 is the recommended value (which means 1000/1 = 1000 Hz polling rate). However, if you ever encounter any issues, try higher integer values. The higher the interval, the higher the possible lag. This shouldn't go above 8 (which means 1000/8 = 125 Hz polling rate).
- To see which value is applied to your controller, you can run `systool -m usbhid -A jspoll` from the Linux shell.

xpad.cpoll: specifies the interval for USB XInput controllers. Most popular controllers use this. There is no practical difference here. XInput is for Microsoft's Xbox consoles and PC.

- 0 is the default value MiSTer uses (even when there is no "u_boot.txt"). In this case the requested value from the controller is used.
- 1 is the recommended value (which means 1000/1 = 1000 Hz polling rate). However, if you ever encounter any issues, try higher integer values. The higher the interval, the higher the possible lag. This shouldn't go above 8 (which means 1000/8 = 125 Hz polling rate).
- To see which value is applied to your controller, you can run `systool -m xpad -A cpoll` from the Linux shell.

Input devices

MiSTer supports many different USB input devices.

Any USB HID compatible device will be recognized.

For recommendations you can refer to Selecting Input Devices.

Keyboard

A keyboard can emulate other input devices, *so basically it is enough to control all cores.*

MiSTer keyboard features a flexible key re-mapping function, mouse and gamepad emulation, and more.

See Keyboard for details

Mouse

MiSTer supports up to 3 buttons (exact number of mouse buttons is core-dependent).

Most USB wired and wireless mice, trackballs and touchpads will work.

Joystick and gamepads

MiSTer has a powerful set of features for USB gaming controllers:

- Up to 6 player support
- Button mapping options
- Auto Fire
- Mouse emulation from joystick
- Debugging options

Joystick player assignment

Up to 6 player controllers are supported (depending on core):

- After a core starts, press a button on any connected controller to make it the P1 gamepad/joystick
- Press a button on a second controller to make it the P2 joystick (if supported by core)
- Keep going for assigning P3, P4, etc.
- To remap, just clear all assignments by restarting the core.

USB Joystick mapping

USB joysticks, gamepads, and keyboards need to be defined in the central menu before use in any core. Please refer to Main Joystick Mapping

Auto fire

Any defined button (except d-pad) supports **auto fire** feature. To activate auto fire, press and keep desired button and then quickly press the button defined as "BUTTON OSD"(for joystick) or "KBD TOGGLE"(for keyboard). To deactivate auto fire, repeat the the same procedure.

Auto fire provides 50ms-1000ms rates. To choose the speed, press and keep one of direction on d-pad and then quickly press the button defined as "BUTTON OSD"(for joystick) or "KBD TOGGLE"(for keyboard).

Mouse emulation

Joystick can emulate mouse if required button **"Mouse Emu"** has been defined in default joystick definition (Menu core). Hold **"Mouse Emu"** button and **"Alt**/**M"**, **Mouse Left**/**Right**/**Middle Btn** will emulate the mouse functions. Also defined analog stick for mouse will be switched to pointer functions. Press **"BUTTON OSD"** while holding **"Mouse Emu"** to toggle mouse emulation permanently. In permanent mouse emulation "Mouse Emu" button becomes a **sniper button** (smaller pointer movements). Only buttons defined for mouse emulation will be switched. Other joystick buttons will continue to act as joystick buttons. Thus, if your game pad has many buttons, you can have both mouse and joystick in one

game pad at the same time (useful for some games, like Walker on Amiga).

Debugging controllers

- Joystick actions can be viewed in serial console while running Menu core.

JammaSD arcade i/o board

MiSTer supports the use of a JammaSD by detecting if the pressed buttons are from player 1 or 2.
You first have to configure player 1 in main menu (as a joypad) (and also remap it in cores if needed).
Player 2 inputs will be auto defined, like if it was a second identical joypad.
JammaSD support was added with a VID/PID that should be the same across all devices, but if your device has a different VID/PID, you can adjust it in the MiSTer.ini file.

Choosing devices

What USB controllers can I use with MiSTer?

Most USB controllers that support HID will work with MiSTer.

There are some known issues with some famous brands which may be worth being aware of, in order to select the best options for your own preferences. More below.

What is the fastest USB controller I can get?

Almost any standard USB controller will work well with MiSTer, however there is a short list of excellent controllers that we can suggest based on user feedback and input lag testing:

- 8bitdo M30 2.4g connected in wired mode
- Sony DS4 in wired mode
- Hori Fighting Commander (later revisions), either PS4 or XBOX1 variant is fine; pick your favorite
- DIY USB adapter using open source firmware available here

Generally speaking, while Bluetooth connected devices will work fine, you can expect the highest amount of input latency while using your controller in this way.

Please keep in mind, “high latency” controllers *should* add, at most, 16 ms (1/60th of a second) or one frame (or rarely, depending on the controller, 32ms) of lag, and not constantly, to your experience, which is not a lot of lag, so if you’re not the type of person who notices that, you can safely just use any decent USB controller. Variable lag (the input response varying between 1 and 3 frames, for instance) is more noticeable than consistent lag. There are very few controllers which add more than 1 frame of lag.

For a set of devices tested with MiSTer, you can refer to this USB controller performance data.

Please see these article for more information about USB controller lag

- https://medium.com/@WydD/controller-input-lag-how-to-measure-it-1ebfd2c9d60
- https://inputlag.science/

Gaming Keyboards, worth it?

High performance and expensive keyboards and mice aren't good for MiSTer. They won't give any benefits, so it's just waste of money. Also these devices have too many functions and many virtual devices cluttering input subsystem which may introduce input lag or be complete unresponsive. They may prevent other devices such as gamepads to work. So try to avoid these gaming Chrismas-Tree like keyboards and mice. Buy a simple one.**

PS3/PS4, XBox360/XBoxOne gamepads

These are known to have some problem with MiSTer. They have accelerometers which constantly sends the events with high rate. Analog sticks also send events even when not touched. Overall, MiSTer receives a flood of events from these controllers, and these extra events may prevent correct button definition. Games may behave incorrectly when using these controllers.

The ideal solution today for these gamepads is to use 3rd-party receivers, such as 8bitdo retro receivers, specifically the **8Bitdo Wireless Bluetooth Adapter**. Not only it gives you wireless access, but also filters out all these unneeded events, while supporting Xbox One S/X, PS3, PS4, Wii, Switch, and 8Bitdo's own gamepads. One receiver will pair with one controller at one time. If multiple controllers are required for multiplayer games, then multiple receivers will need to be purchased.

The Grey/Orange (brick decorated) USB Adapters are functionally the same, after using the latest firmware. Gamepads may switch to different input modes using hotkeys for different functionality. Note that documentation on 8Bitdo's site doesn't specify this, but the update logs for the firmware updates does.

- X-Input mode: Hold SELECT+UP for 3 seconds.
- PSC (Playstation Classic) mode: Hold SELECT+DOWN for 3 seconds. This is useful for gamepads which are limited in buttons (12 total; DPAD counts as 4) and need to access the MiSTer OSD menu. Note that the OSD menu should not be assigned when configuring buttons in

the main MiSTer menu core, as the L+R+START combination will bring up the OSD while in the cores. The combination is hard-coded in MiSTer specifically for 8Bitdo adapters. You may also lose auto-fire/mouse functionality in this mode.

Alternative 8Bitdo adapters, such as the 8Bitdo Console Retro Receiver (SNES, NES, Genesis) are always in X-Input mode when connected via microUSB.

Bluetooth Adapters and Dongles:
Any off-the-shelf bluetooth dongles will work with most wireless controllers like Dualshock4, Xbox, 8Bitdo.
See Bluetooth for details

Joystick mapping

Before using any controller with a MiSTer core, it must be defined in the main menu core

MiSTer's mapping system

MiSTer has a simple three-step mapping system:

1. Define centrally a physical controller into a virtual "MiSTer gamepad" (+ extras)
2. MiSTer gamepad is mapped automatically to cores[1]
3. If needed, you can override at any time per core (and per controller) via OSD menu

1 - Mapping of MiSTer gamepad into cores has two "flavors" of operation, chosen by INI file. More below.

Preconfigured MiSTer mappings

Community-created controller mappings can be found in this repository.

They need to be extracted into SD card folder:

> /media/fat/Inputs/

Alternatively you can define your own mappings following the steps below.

MiSTer Gamepad

The first step is to teach MiSTer to recognize your physical controller.

This is done in the menu shown at startup.

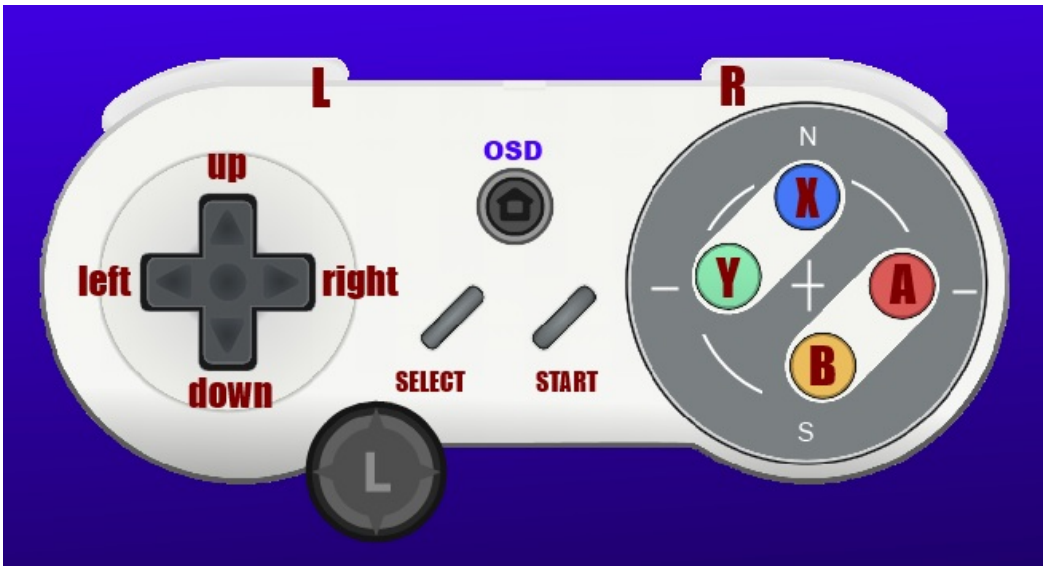
After plugging a keyboard, press F12 to show system settings and select "Define joystick buttons"

MiSTer will then ask you to assign several buttons to your controller:

- Test the D-Pad and analog sticks (if any)
- Four face buttons and two shoulder buttons
- Start and Select
- OSD button or 2-button combo (to use instead of F12 on the keyboard)

- A few extra buttons for advanced functions (more below)

Here is a conceptual representation of the MiSTer Gamepad:



MiSTer internally recognizes more buttons (and two analogue sticks), but the above is the minimum required to work on most cores. You can also ignore the analogue stick as long as you do not use the few cores that require it (e.g. Apple II, Atari 5200).

From USB hardware to MiSTer Gamepad

This step is exclusively handled in the main startup menu. You can override it by re-doing a mapping from inside a core.

MiSTer uses USB HID to handle controllers. Each controller is recognized by a unique USB ID (VID:PID) and declares what it can do (does it have d-pad? how many buttons?). This information does not contain any physical information (which button is "A", where it is located...) and you may want to decide which buttons to specifically assign for each function.

That is where you come in. MiSTer will ask you to:

- Test D-Pad and analogue sticks (effectively a joystick calibration)
- Choose what to use for directions (can be D-Pad, any analogue stick, or even buttons)

Then it will ask you to assing:

- A, B, X, Y, L, R buttons to the MiSTer gamepad (see image above)
- Start and Select buttons
- A button or a 2-button combo to open the OSD

And finally it will ask to select:

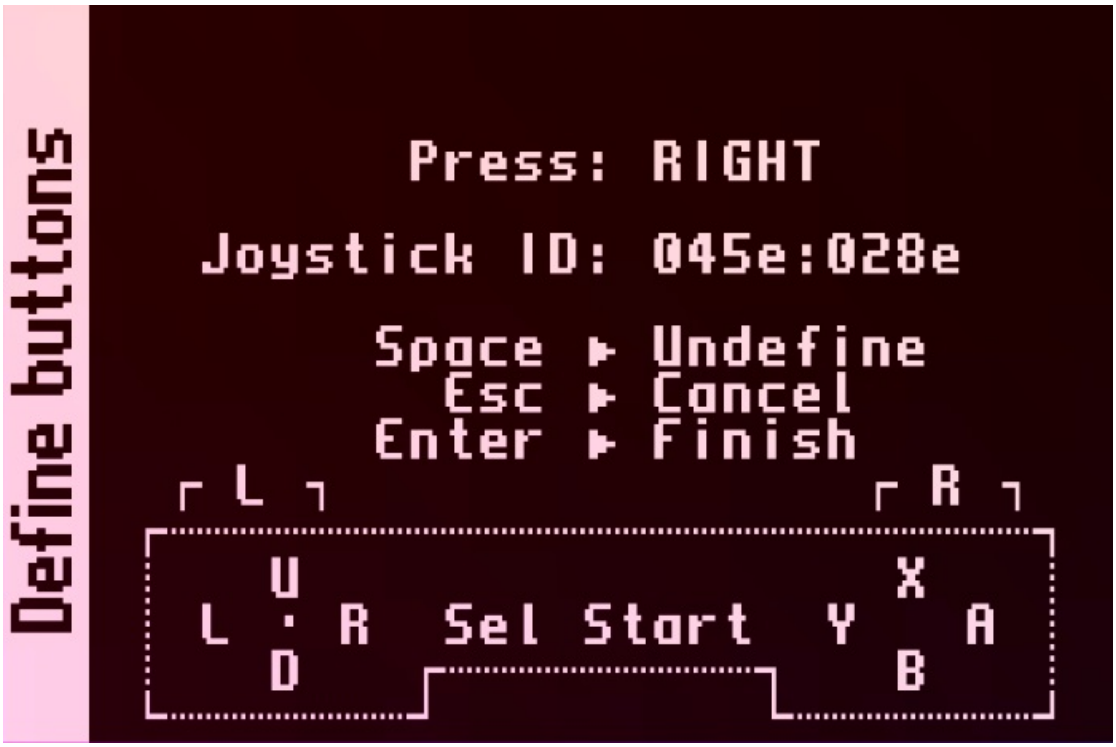
- Alternative directions (also used for mouse emulation)
- Controller buttons to use as mouse buttons (left, right, middle)
- Button to turn on mouse emu (and to use as "sniper" mouse mode)

See below for a more technical description of each step.

During the mapping you can press:

- **ESC** to cancel the mapping operation and return to menu
- **SPACE** to skip a button (that button will remain undefined)
- **ENTER** to end the mapping without mapping any further buttons (keeps maps done so far)

The screen will also display the VID:PID (the USB ID) of your controller.



More technical details on each step

Calibration

- **D-Pad type test** This is an important step since some USB controllers generate analog events from the D-Pad, and it that case a small calibration must be done for all directions to be recognized. Simply press the RIGHT button on D-Pad first. If it generates analog stick event, then MiSTer will ask to press DOWN as a second stage. Otherwise it will proceed to the next steps if no analog event has been detected. If you are defining keyboard keys for joystick emulation, then simply press the RIGHT arrow key and it will skip Stick 1/2 steps and jump to D-Pad keys definition.
- **Stick 1 and 2 analog axes** If your gamepad has no analog sticks then skip this step via the SPACE key. Otherwise, you need to define them for it to be calibrated and use proper analog to digital mapping. (**Note:** Some gamepads like 8bitdo M30 emulate analog stick events on DPAD, so you have to press RIGHT and DOWN for Stick 1! Skip for Stick 2). Note that these definitions have no relation to any specific mapping, it only tells to MiSTer these axes have min-0-max values with 0 as a default position and will be mapped to 2 digital buttons at the both ends. Other analog axes not defined here will be treated as 0-max with only 1 digital button.

Virtual D-Pad and Buttons

- **Right,Left,Down,Up** Virtual D-Pad directions. You can use an analog stick or even buttons if desired.
- **A,B,X,Y,L,R,Select,Start**. Basic buttons, any physical button can be assigned to these. Skip with SPACE if your physical controller is missing some buttons.

Alternate layout and Mouse emulation

- **Right,Left,Down,Up (Mouse and Alt)** - alternative direction control. If your gamepad has analog stick, then you can assign it here. In cores this alternative control will work in parallel to the one defined in the core. So you will be able to control directions with DPAD(or whatever you've defined in core) and analog stick. Some games are good to be controlled by stick, other games are good with DPAD. These controls are used for mouse emulation as well, so if you don't have the stick (or don't want to use it), then define DPAD buttons here again. The same sticks defined here will be used for mouse emulation.
- **Mouse Left/Right/Mid Btn** - buttons for mouse emulation. If your gamepad has many buttons then you can define separate buttons for mouse only, so when in mouse emulation mode both joystick and mouse buttons will be available at the same time. On reduced gamepads you may define the same buttons used for Btn1..Btn4.
- **Mouse Emu/Sniper** - button to switch to mouse emulation. While holding it down gamepad will emulate the mouse. In permanent mouse mode (press OSD button while in temporary mouse mode) this button is used for smaller pointer steps (sniper mode).

System and analog stick selection

- **BUTTON OSD** - important button used to access OSD menu and some additional functions.

- **Stick X/Y** - analog axes. Some cores support or even require analog joystick. This allows you to define exactly which input is used for this. For gamepads this is usually the left stick.

Note: Keyboard can be used as joystick. So you have to define the keyboard as joystick in both Menu core and the core you want to use!

Mapping settings are specific to each device (identified by VID and PID). If you have several identical gamepads/joysticks then they will share the same button layout.

The number of button supported per core varies (up to 32). While defining buttons, you can press "SPACE" to skip (keep undefined) the button, "ESC" to cancel, and "Enter" to stop mapping (i.e. make the rest of buttons undefined).

Multi Button

When overriding gamepad settings inside a core it is possible to define button combinations (e.g. A+B) if you have enough physical buttons.

To allow for this, MiSTer will ask if you want to setup "alternate mappings" after defining a gamepad.

Alternate mappings are active in parallel and allow two kinds of abilities: map two physical button to the same core button (e.g. alternate button for Start), or map one physical button to two-button combo in the core (e.g. map physical X to thr core's A+B).

Bind core button to two physical buttons

The simplest is to map an extra physical button. Only this additional button needs to be defined in the alternate map, as follows:

map	**d-pad**	**A**	**B**
main	*define*	Btn 1	Btn 2
alt	*skip*	*skip*	Btn 3

The result of using this setup ia that you will have physical buttons 2 and 3 mapped to the core button B.

This also works with gamepad directions. For example, for games that use Up as a jump button, it is possible to map a physical third button to jump.

Bind physical button to core button combo

A more advanced setup is to use both mappings in parallel to make one physical button push two core buttons at the same time.

See the table below:

map	**d-pad**	**A**	**B**
main	*define*	Btn 1	Btn 3
alt	*skip*	Btn 3	Btn 2

The result of this is that Button 1 will be A, Button 2 will be B, and Button 3 will be A+B.

This is particularly useful for the Neogeo core where some fighting games use A+B and C+D as "strong hit".

(note: unfortunately NeoGeo games are not consistent with each other - so it is not possible to set one mapping for all games)

Example: NeoGeo Breakers / Breakers' Revenge

Assume a six button arcade controller (for Street Fighter 2) with 3 punch buttons on top and 3 kick buttons below.

We will map the buttons so that Breakers / Breakers' revenge has the same kind of layout.

This particular game uses the following attack buttons:

- A: Weak Punch
- B: Weak Kick
- C: Strong Punch
- D: Strong Kick

In addition, most characters have an extra "command" move mapped to:

- Weak Punch + Strong Punch
- Weak Kick + Strong Kick

To match the Street Fighter 2 layout, we can assign this across six buttons:

-	Weak	Strong	Weak+Strong
Punch	A	C	A + C
Kick	B	D	B + D

Which translates to the following buttons:

-	Weak	Strong	Weak+Strong
Punch	Btn 1	Btn 2	Btn 3
Kick	Btn 4	Btn 5	Btn 6

To obtain this, input the following assignment during main and alternate mapping:

map	d-pad	A	B	C	D
main	*define*	Btn 1	Btn 4	Btn 3	Btn 6
alt	*skip*	Btn 3	Btn 6	Btn 2	Btn 5

(in alternate mapping, d-pad and all other buttons can be skipped)

Example: NeoGeo King of Fighters / Fatal Fury Special

Both of these game use the same attack layout as Breakers but their button combination differs:

-	Weak	Strong	Combo
Punch	A	C	A + B
Kick	B	D	C + D

To map this, we need to switch things around a little bit:

map	d-pad	A	B	C	D
main	*define*	Btn 1	*Btn 3*	*Btn 2*	Btn 6
alt	*skip*	Btn 3	*Btn 4*	*Btn 6*	Btn 5

Example: NeoGeo Samurai Shodown

This game has yet another layout, changing the order of base attacks:

-	Weak	Medium	Strong
Slash	A	B	A + B
Kick	C	D	C + D

We can map this as follows:

map	d-pad	A	B	C	D
main	*define*	Btn 1	Btn 3	*Btn 4*	Btn 6
alt	*skip*	Btn 3	*Btn 2*	Btn 6	Btn 5

Counter Example: The Last Blade 1 and 2

Very similar to the above but a different button combo:

-	Weak	Medium	Special
Slash	A	B	*B + C*
Kick	C	D	C + D

In this case, it is not possible to map C three times *so we must choose whether to map B+C or C+D*.

Option 1: Mapping B+C to button 3:

map	d-pad	A	B	C	D
main	*define*	Btn 1	Btn 2	Btn 3	Btn 5
alt	*skip*	*skip*	Btn 3	Btn 4	*skip*

(Btn 6 remains unassigned)

Option 2: Mapping C+D to button 6:

map	d-pad	A	B	C	D
main	*define*	Btn 1	Btn 2	Btn 6	Btn 5
alt	*skip*	*skip*	*skip*	Btn 4	Btn 6

(Btn 3 remains unassigned)

Bluetooth

MiSTer supports Bluetooth input devices.

Supported devices: Gamepads, Keyboards, Touch pads(as a part of multifunctional keyboard). Support for mouse and standalone touch pads is unclear. Need to be tested.

Bluetooth host must be a standalone USB BT dongle. Multifunctional dongles such as WiFi+BT probably won't work. Confirmed to work dongles based on CSR8510 and BCM20702 chips. Most (if not all) BT dongles you can buy today are based on these chips.

BT pairing dialog.

Either press and hold OSD button on I/O board for 3 seconds, or F11 key (only when OSD is active).

Pairing for gamepads and keyboards

- Put your gamepad/keyboard into pairing mode.
- Invoke BT pairing dialog.
- MiSTer will try to find and pair the device (Keyboards require to enter pin 0000 and press enter).

Pairing for Dualshock 3 and Sixaxis gamepads

- Connect the gamepad through USB to MiSTer.
- Press PS button and wait for lights stop to blinking and only one remain active.
- Disconnect the gamepad - lights will start to blink and then only one will remain active if succeeded.

Note: Dualshock 4 gamepad is a standard BT gamepad. So follow the first paring method.

Notes / Troubleshooting

- Only single BT dongle is supported.
- Depends on environment condition (how many WiFi spots and active BT devices are around) you may connect more or less BT devices at the same time. In my place i could successfully connect 3 BT devices at the same time. The 4th one couldn't be connected till i turn off one of connected. Other BT dongle allowed only 2 gamepads at the same time.
- After MiSTer reboot many BT devices won't re-connect automatically. Some devices will shutdown them selves immediately, other devices need to be turned off manually then on.
- Bluetooth support was added to MiSTer_20190406. Make sure to update the Linux image, menu core and mister main. As of 2019-4-17 the updater script doesn't update the Linux image by default, use the SD Installer to update the base Linux image.
- Spotty connections and difficulty pairing have been reported with non-powered USB hubs. A powered USB hub is always recommend.
- BCM20702 BT dongles may stuck in RF unresponsive state after reboot. From driver point of view device looks like working, but none of BT devices able to connect. Currently the only fix is to re-plug the dongle. CSR based dongles have no such issue.
- Console/SSH bluetooth debug commands:
  - You can SSH to the MiSTer and run `hcitool dev` to see if your BT dongle is recognized.
  - `hcitool scan` will scan and print a list of recognized Bluetooth clients.
  - `btpair` starts the same Bluetooth pairing script accessible via F11 in the MiSTer menu.
- MiSTer may not pair with Bluetooth controllers if other BT devices are present and scanning, such as Samsung Smart TVs. Turn these devices off to complete pairing; afterwards you can turn them back on.

Wiimote

Wiimote is supported natively from Linux release 20190510. It needs to be paired just like any other BT device. You must use red sync button on the back of Wiimite (Pairing by buttons 1+2 doesn't work!).

It's recommended to use the first Wiimote version (the one without integrated Motion Plus). This Wiimote just need to be paired once and it will automatically connect on next powering the Wiimote. Second version of Wiimote (with integrated Motion Plus, TR) is also supported but it won't be able to automatically connect later (probably due to very short time it gives to re-connect), so you have to pair it every time you want to use. 3rd party Wiimotes may or may not work, or work with problems. So it's advised to use official Wiimote.

Nunchuck and Classic Controller connected to Wiimote are supported. Mayflash Gamecube adapter for Wiimote is also supported.

To fully utilize the Wiimote you have to connect IR bar to power. You can buy a 3rd part IR bar with USB connector or modify original one to get the power from USB.

Dolphinbar

Dolphinbar is supported at some degree. You can use modes 1 and 2 as a light gun or mouse control in some cores (Wiimote is connected to P2 joystick, so you need to select Light Gun on Joystick 2 port in the core with mouse as a trigger). Mode 3 can be used as a traditional gamepad. Mode 4 (the main mode) is not supported due to absence of Linux drivers.

Overall Dolphinbar is quite useless and not recommended due to non-working mode 4. Native Wiimote support can do everything with traditional dummy IR bar.

Keyboard Handling

Keyboard

MiSTer supports keys re-mapping which is useful for reduced or localized keyboards. Key remapping is system wide, so every core will have same key map. Keep in mind it's not macro definition, so single key is remapped to another single key. Some multimedia keys generate several key codes - these keys cannot be remapped. Each keyboard model has its own key map stored in /**media/fat/config/kbd_[VID]_[PID].map** file. To reset all keys to default state, simply delete appropriate map file. Key remapping is available through Menu core only.

Joystick emulation

Keyboard can be switched to joystick emulation. You need to define keys used for joystick emulation the same way you did for joysticks. Auto fire is also supported the same way as for joysticks. Button defined for "KBD TOGGLE" provides a quick switch between keyboard and joystick for defined keys.

Mouse emulation

Keyboard can be switched to mouse emulation. You need to define mouse emulation buttons in Menu core the same way as for joystick.

Emulation switch

To switch between emulation modes press **NumLock** or **ScrLock** till desired mode is selected.

Switching sequence is **Mouse >> Joy1 >> Joy2 >> None**

LEDs on keyboard display the emulation modes:

- Mouse emulation: NumLock LED + ScrLock LED
- Joystick 1 emulation: NumLock LED.
- Joystick 2 emulation: ScrLock LED.

Common functional keys/combos used in cores

- **F12** - open/close OSD menu/submenu
- **Alt-F12** - quick core selection (like in Menu core).
- **LCtrl+LAlt+RAlt** - presses the "USER" button which usually is reset in emulated system.
- **LShift+LCtrl+LAlt+RAlt** - MiSTer reset (load Menu core).

Notes:

- Some systems provide writing support which requires additional attention to how you reset/shutdown the MiSTer. MiSTer tries not to keep any pending writes and writes physically to the disk as soon as possible. Still, safer way to reset the MiSTer from core which probably was writing to disk recently is using combo **LShift+LCtrl+LAlt+RAlt** - this will flush all caches to disk before restart. Cores without write can be restarted by hard reset button or powered down without special attention.
- LCtrl+LAlt+RAlt sequence can be replaced by some other well known combos through INI file.

DIY Arcade Input

You have assembled an arcade cabinet and looking for connecting your joysticks and buttons? You don't need to buy any of those expensive adapters. All you need is USB keyboard which you can disassemble.

After disassembling the keyboard you need to find the row and column contacts on board for following keys:

```
5,         1P coin
1,         1P start (shift key)
UP,        1P up
DOWN,      1P down
LEFT,      1P left
RIGHT,     1P right
LEFTCTRL,  1P button 1
LEFTALT,   1P button 2
SPACE,     1P button 3
LEFTSHIFT, 1P button 4
Z,         1P button 5
X,         1P button 6
C,         1P button 7
V,         1P button 8

6,         2P coin
2,         2P start
R,         2P up
F,         2P down
D,         2P left
G,         2P right
A,         2P button 1
S,         2P button 2
Q,         2P button 3
W,         2P button 4
I,         2P button 5
K,         2P button 6
J,         2P button 7
L,         2P button 8

9,         Test
TAB,       Tab (shift + 1P right)
ENTER,     Enter (shift + 1P left)
P,         P (pause) (shift + 1P down)
F1,        Service
F2,        Test
F3,        Tilt
```

You can use multi-meter to find the contacts.

Then connect your buttons/joystick to board according to table above. Your hardware part is done. Connect your modded arcade keyboard to USB and start Menu core, then go to `Define Joystick buttons`. Press any button, and you will see "Keyboard ID: XXXX:YYYY" where XXXX:YYYY is HW identifier of the keyboard. Press ESC to ext from this dialog. Now open MiSTer.ini and add following options there:

```
jamma_vid=XXXX
jamma_pid=YYYY
```

XXXX and YYYY are those IDs you saw for your modded keyboard. Save and restart Menu core. Go to `Define Joystick buttons` again and assign buttons using P1 buttons. Note it now writes "Joystick ID: XXXX:YYYY" (instead of Keyboard) which means it recognizes your modded keyboard as jamma 2-player input device. P2 buttons will be automatically assigned the same as for P1.

Now your Arcade input device is ready to use. Unlike normal gamepads/joysticks, this device will always assign P1 and P2 inputs statically regardless P1 or P2 button was pressed first.

WiFi

NOTE: This page will be getting rewritten soon.

For now, the easiest way to configure WIFI is by pressing the F12 button for the MiSTer menu, go to the scripts folder and run the WIFI script. Better instructions coming soon!!

Starting from Release 20180115 MiSTer supports some WiFi USB modules.

Enable WiFi connection

- locate the file **linux/_wpa_supplicant.conf** (example: /media/fat/linux/_wpa_supplicant.conf)
- open in text editor **supporting Linux/Unix** line endings (for example Notepad++)
- replace **put_your_SSID_here** with your actual WiFi network name and **put_your_password_here** with your WiFi password.
- sometimes you need to change the country code from TW to yours.
- rename **_wpa_supplicant.conf** to **wpa_supplicant.conf**
- reboot the MiSTer

In Menu core you will see WiFi icon when WiFi is connected.

WiFi USB dongles Confirmed to work (running command "lsusb" on linux will show if USB ID matches)

- ASUS USB AC53 nano rev A1.
- CanaKit USB WiFi dongle - Works out of box (USB ID: 148f:5370, driver: rt2800usb, firmware: rt2870.bin)
- Comfast CF-812AC (USB ID 0bda:b812, 5GHz capable, works out of the box)
- D-Link DWA-171 HWVer: A1. (NOTE! D-Link DWA-171 HWVer: C1 / USB ID 0bda:1a2b does NOT work out of the box)
- Edimax EW-7612UAn V2
- Edimax EW-7811UN (USB ID 7392:7811)
- Edimax EW-7822ULC (USB ID 7392:b822, 5ghz capable)
- Netgear A6100 (USB ID 0846:9052, 5GHz capable, works out of the box, driver: rtl8821au)
- Netgear A6150 (USB ID 0846:9055, driver: rtl88x2bu)
- TP-LINK TL-WN823N V2 (needs copy rtl8192eu_nic.bin to /lib/firmware/rtlwifi)

Some WiFi firmwares can be found here: https://github.com/wkennington/linux-firmware

Compiling and installing custom WiFi drivers

Instructions for rtl8188fu based adapters (like the Zapo RTL8188 USB stick) can be read here:

MiSTer custom WiFi driver compilation

Steps can be adapted for other WiFi adapters.

Troubleshooting

If you are having trouble with staying reliably connected to your WiFi, try separating the SSID's for your 2.4GHz and your 5GHz connections on your Router's firmware configuration. This should help solve the problem. If not, it is possible there is something wrong with your WiFi adapter, or it is not fully supported by the current build of Linux for the MiSTer project.

FTP, SSH/SFTP

MiSTer board can be access through on-board Ethernet port. System has **FTP**, **SSH**, **SFTP** services running.

User name: **root** Password: **1**

When using FTP, make sure your file transfers are in Binary not ASCII. ASCII-type transfers will result in corruption of that data. A good FTP client for Windows that defaults to Binary and supports FTP as well as SFTP and SCP (ssh copy) is WinSCP.

By default, DHCP is used to acquire an IP address for the board. You can find it from your router (easier way), or from console connected to the board using the command **ifconfig**

The default MAC address for the on-board Ethernet port is `02:03:04:05:06:07`. Please consider this when connecting more than one MiSTer board via the on-board Ethernet port to your network.

> TODO: How to setup a custom MAC address.

Setting up a static IP address

Currently the best way to do this is using `connmanctl` (try `connmanctl help` for more info).

- Create /var/lib/connman directory so changes will persist across reboots

```
# mkdir /var/lib/connman
```

- Find your on-board network service name.

```
# connmanctl services
*AO Wired               ethernet_020304050607_cable
```

- Setup IP address (e.g. `192.168.1.123`, should be unused), subnet mask (e.g. `255.255.255.0`) and gateway (e.g. `192.168.1.1`, typically your router IP address). To configure the right device use the service name returned from above command (e.g. `ethernet_020304050607_cable`). This will disable the use of DHCP.

```
# connmanctl config ethernet_020304050607_cable --ipv4 manual 192.168.1.123 255.255.255.0 192.168.1.1
```

- Setup one or more DNS server(s) (e.g. `192.168.1.1` from your router, `8.8.8.8` from Google DNS.

```
# connmanctl config ethernet_020304050607_cable --nameservers 192.168.1.1 8.8.8.8
```

You can write a script to help with typing and remembering your settings.

> You can also setup your on-board network connection by editing `/etc/network/interfaces`. This is currently not advised. Because if you have a DHCP server in your network, the network stack will still contact the DHCP server and assign the returned IP address regardless, leading to usually unwanted behavior.

(Re)Enabling DHCP

```
# connmanctl config ethernet_020304050607_cable --dhcp
```

Other

mc (midnight commander) file manager is available for easier folder navigation.

The root of SD card: /**media/fat**

Related: Serial Console Access

Samba / Windows Network Shares

You can access the MiSTer through Samba network.

Why Should You Use Samba?

Using Samba you can work with files much the same way that you work with files locally on your PC, with a File Explorer, clicking and dragging, etc... This is much easier for most people.

Samba (also known as SMB) is the native protocol for Windows shares. MiSTer already has FTP and SSH services, but Samba has a special feature: Windows treats Samba shares as a local filesystem.

For example, with FTP/SFTP,, if you want to view or edit a file, then you need to download it fully first. With Samba, only a small required portion of file will be loaded. If you are editing a large file of 100MB+, then it makes big difference. Similarly, if you want to use a HEX editor, you don't need to download the whole file. Just open the file in a HEX editor and only small portion will be loaded where you can quickly edit required bytes and save it back quickly.

This means that with Samba access, you can mount VHD files on your PC without downloading them! Use a utility such as ImDisk to can mount VHD files as a local disk (mount it as removable store for easier un-mounting).

Instructions:

By default Samba service is not active. Follow the following instructions to turn it on:

1. Rename `/media/fat/linux/_samba.sh` to `/media/fat/linux/samba.sh` in the terminal/console on the MiSTer by running the following command:

   ```
   mv /media/fat/linux/_samba.sh /media/fat/linux/samba.sh
   ```

2. Reboot the MiSTer
3. You can access the MiSTer in your File Explorer either by IP address (e.g. `\\192.168.1.100\`) or by hostname (e.g. `\\MiSTer` or `\\mister` as it is not case sensitive).

OPTIONAL: Edit this file if you want a specific username and password (default is user `root` with pass `1`)

Notes:

- Make sure you've closed all opened remote files and un-mounted all remote VHDs before restarting the MiSTer or start the cores using the same VHDs in order to prevent the data corruption!
- MiSTer's default samba workgroup is `MiSTer`. Usually you will not need this to log in.
- To map the MiSTer as a "Network Drive" in Windows, you will need to use a share path (e.g. `\\MiSTer\sdcard\`) instead of just the hostname. This is due to a limitation of Windows itself ever since Windows Vista. It can be disabled by enabling SMBv1 client services manually in Windows but this is not advised as SMBv1 is not secure and has multiple security exploits.
- Alternatively, you can just "Map Network Location" instead to `\\mister\` directly to avoid this.

Troubleshooting:

If you're using a Windows OS (Vista and above) while trying to access the share and the credentials do not work, there may be a possibility that the LAN Manager authentication level is not being worked out correctly between the Windows OS and the Samba daemon on MiSTer.

The error message may manifest itself as a **The specified network password is not correct** error. A fix is to lower the Samba NTLM authentication level on the MiSTer to NTLM v1.

This is done with the following steps:

1. SSH into your MiSTer instance.
2. Edit the Samba configuration file.

   ```
   nano /etc/samba/smb.conf
   ```

3. Append under global, where the keyword **yes** signifies ntlmv1-permitted, which allows for NTLMv1 and above for all clients (against MiSTer). By default the value is not set explicitly and is **no**, which equates to ntlmv2-only. Further information is specified in the **ntlm auth (G)** section

at https://www.samba.org/samba/docs/current/man-html/smb.conf.5.html.

```
[global]
min protocol = SMB2
ntlm auth = yes
```

4. Reboot MiSTer or restart the Samba daemon.

```
/etc/init.d/S91smb restart
```

You should see no errors when manually restarting. e.g.

```
/root# /etc/init.d/S91smb restart
Shutting down SMB services: OK
Shutting down NMB services: OK
Starting SMB services: OK
Starting NMB services: OK
```

Alternatively, on the Windows OS, change the Windows registry HKLM\SYSTEM\CurrentControlSet\Control\Lsa\LMCompatibilityLevel to 3. Note that you may/may not have permission depending on your security policy or Administrator rights. This is described by Microsoft at https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-lan-manager-authentication-level.

Internet for Amiga/ao486

Starting from 2018 may 7 release MiSTer supports serial (UART) connection from FPGA to Linux. Linux OS runs PPP or Console daemon on this connection allowing access the internet or Linux shell from FPGA cores.

Cores supporting serial connection

- **Minimig**. Tested on Roadshow TCP/IP, AmiTCP and Miami.
- AmiTCP provides more complete solution with ftpd daemon. There are many other 3rd party addons are based on AmiTCP, so it's advised to use this package.
- Roadshow works very well, it is fully compatible with AmiTCP and offers additional extensions. Follow these Instructions for complete setup. It is still a paid for and supported product, you can find more information here.
- Miami was successfully tested. The Miami settings that worked: use PPP connection via serial.device, set baud rate to 115200, RTS/CTS to on, and enable 8N1. Set modem to nullmodem. Manually enter an IP suitable for your lan ending in 254, e.g. 192.168.1.254. Manually add a DNS server, e.g. 8.8.8.8 for Google DNS. Term v4.7 has been used to test console connection.
- **ao486**. Currently only console connection has been tested using Dos Navigator's integrated Terminal and Kermit 3.15. PPP should work under Win95. DOS tools are here : dos_ftpd.zip. The DOS FTP server included does not support passive mode, so set your client to use active.
- **C64**. Serial connection.

OSD provides an option to switch between PPP and Console on these cores. Both console and PPP are using baud rate 115200 8N1 mode with hardware RTS/CTS flow control for stability.

Console connection

Using this connection with supported terminal application on FPGA core, you can access the Linux shell and do some file managements or Linux settings if required. No special settings are required of Linux.

PPP connection

Using this connection core may have internet connection. More important, the core may run ftp daemon and provide access to its filesystem, so you can use FTP client on PC to move the files to/from the emulated system.

PPP daemon uses **/media/fat/linux/ppp_options** (linux\ppp_options of PC) file. Most likely you don't need to modify it. Recent update assigns IPs automatically. Core gets <your_net>.254 IP (for example 192.168.1.254). If you want other IP, then modify ppp_options file. For correct PPP work, make sure you see a network icon in Menu core before starting the other core. Otherwise PPP link won't get IPs. If you've started core earlier, then simply connect the core to PPP and disconnect. Next connection will get correct IP. Or you can switch UART mode in OSD to renew the IP.

NOTE: I'm looking Amiga and MSDOS terminal supporting color and control codes of linux, so it will be possible to use Midnight Commander in terminal connection. If you know such terminal application, then let me know.

PPP connection in Windows 95 on ao486

Unfortunately winsock and winsock2 provided by Microsoft do not work with the ppp connection when in Windows 95. The following steps will allow you to get it working.

1. In the Mister System Menu (Win/F12) set the "Uart Connection" to "PPP" and save it.
2. In Windows 95 ensure the COM1 device is installed in Start->Settings->Control Panel->System

   Device Manager Tab, there should be a twisty called Ports(COM & LPT) and under that a "Communications Port (COM1)"
3. If it doesn't exist go to Start->Settings->Control Panel->Add/New Hardware and it should be automatically added.
4. Get the replacement PPP client
   Download the software. There are other newer versions available BUT be warned only version 3.0 will work.
   Trumpet Winsock 3.0 or Official Homepage

(I extracted the file from the disk image and uploaded it using the DOS ftp client documented above)

5. License the Software
   This software is still shareware (time limited) please license it appropriately. Once you acquire a license you can put the details in Tcpman in the "Special" menu in "Password registration"

6. Configure Software

   1. Start Tcpman

   2. Under File->PPP Options ensure all checkboxes are unchecked and the text boxes are blank.

   3. Under File->Setup Enter an "IP Address" suitable for your LAN eg 192.168.1.254 and a "DNS Server(s)" 192.168.1.1

      Under the Driver section select the PPP radio button and click on "Dialer settings..."

   4. In the "Dialer settings..."

      "COMM port" COM1

      "Baud rate" 115200

7. Using the software (important, be patient)

   Win95 is rather slow so let it start fully before starting the PPP manager (Tcpman)

   Once it is started it will begin syncing with PPP on the linux host . . . Be patient it takes a few seconds.

   When you see the PPP[C021] SND and RCV you can start your TCP/IP program

I have found it to be a little complicated to get started, but once it is running it is rock solid and supports multiple client programs at once.

Serial connection on C64

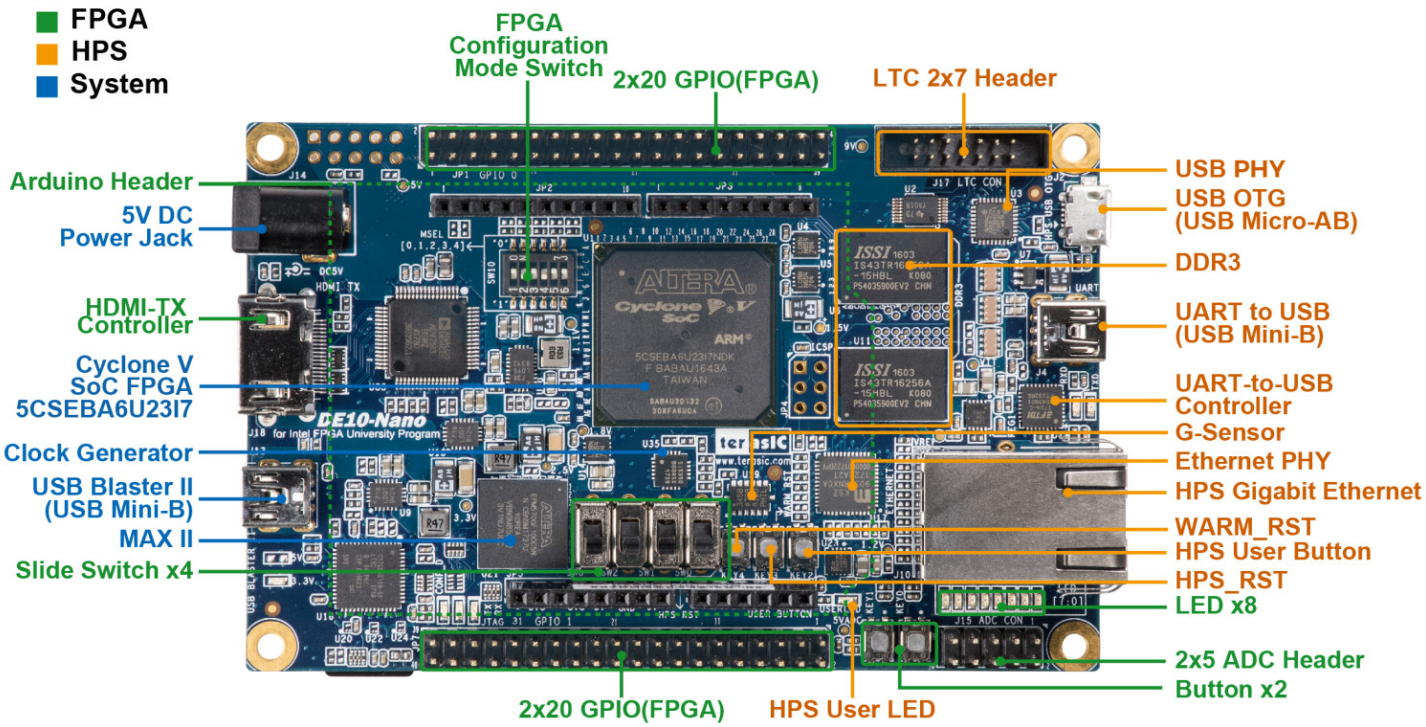
The following is an example for connecting to a BBS using Striketerm 2014.

1. Start the C64 core (please note that custom kernels may remove functionality required, if in doubt use the built in kernel).
2. In the Mister C64 Menu (Win/F12) set the "User Port" to "UART", and save it.
3. In the Mister System Menu (Win/F12) set the "Uart Connection" to "Midi", "Remote", "TCP" and save it.
4. Load Striketerm 2014 from d64. Available from here
5. Keep the defaults in the Main Menu (F1), ensure you are running at 2400 baud.
6. Save a BBS into the Addressbook (F5), you can get some from here
7. Surf the BBS very slowly . . .

Console connection (Serial UART)

The DE10-nano board has console port. The console allows you to login into Linux without a network connection and also provides some debug/info which sometimes useful to track the problems.

Refer to the **UART-to-USB (USB Mini-B)** connector on the board right side in the picture below:



How to connect

Connect the DE10-nano board to a PC using the **UART-to-USB (USB mini type B)** connector next to micro-USB. The PC will recognize it as virtual COM port. Use any console application to connect to this COM port. I recommend Putty.

Verify that COM port settings are correct:

- Speed (baud rate) - 115200 bits per second
- Data bits - 8
- Stop bits - 1
- Parity - none
- Flow control - none

Linux Connection

1. Be sure to be a member of the **dialout** group.
2. Your serial port is likely to be **/dev/ttyUSB0**
3. Configure it: `$ stty -F /dev/ttyUSB0 115200 cs8 -cstopb -parenb -ixoff -ixon`
4. Look at the output while dumping it to a file: `cat /dev/ttyUSB0 | tee mr.log`

Now mr.log has a copy of all the information shown in the screen.

U-Boot command prompt

To interrupt u-boot and get into the u-boot command prompt once connected to the DE10-Nano, hold 'ESC' on the PC and then power on or reboot the Nano (using the reset button). Startup should be interrupted and you should see a '=>' prompt. Here you can edit the kernel boot options etc.

Cheat Engine

Setup

Cheats should be stored in: `cheats/system/rom_filename.zip` where system is name of the core like NES, SNES, Genesis, etc..

So, for example: `cheats/NES/Taboo (USA).zip` for the NES ROM `Taboo (USA).nes`

The filename of the zip must match the ROM name exactly. Cheats will be loaded automatically when you load a ROM, and can be enabled and disabled from the menu in supported cores.

If the .zip cheat file does not match the ROM name, the cheat engine will automatically check the ROM CRC32 and select the appropriate .zip cheat file with matching CRC32 accordingly. ROM name matching has priority over CRC32 checking.

Individual cheats are in .gg format and should be stored in zip files. Packs of codes can be downloaded from https://gamehacking.org/ by selecting the MiSTer format and choosing to save all codes for a given game.

Making your own codes

All types of cheat codes for 16 bit systems and earlier can be decoded into four pieces of information: An address, a compare value, a replace value, and usually a flag to say if the compare value is used or not. The format for a gg file is in binary as 4 32 bit integers in little-endian byte order.

For example, if the Address is `0xFF1CA0` and the compare value is `0xB5` and the replace value is `0xFF`, the file would look like this:

```
01 00 00 00 A0 1C FF 00 B5 00 00 00 FF 00 00 00
```

The first four bytes are little-endian `0x00000001` for "compare enabled", the second four are little-endian address, third set are compare value, and fourth is replace value. Note that not all codes use a compare value.

For cheats with multiple codes, simply add another 16 bytes at the end of the file in the same format as the first.

You can decode game genie codes into these values with tools like this: https://gamehacking.org/system/nes

Video Filters

Introduction

If you use MiSTer you should be using a custom filter for upscaling if you want high quality without uneven scaling and shimmering during scrolling.

The MiSTer Filters and Gamma Github repository is here: MiSTer Filters and Gamma Github

As of November 2018, MiSTer can use custom filter coefficients to define the interpolation used for scaling over the HDMI output (and VGA too with a mister.ini option). This allows MiSTer to scale images using well-known image scaling algorithms such as bicubic or lanczos scaling as well as other scaling methods better suited to scaling pixel graphics. Special effects such as scanlines and lcd effects are also possible.

This github repository houses all of the Interpolation Filters that have been created so far for MiSTer: MiSTer Filters/Gamma Github Repository

As of November 2019 this github also contains Gamma Lookup Tables to allow MiSTer apply a gamma curves to cores with an updated framework. James-F has created a number of curves that are included in the Gamma folder of this repository.

How to use filter coefficients and gamma tables

To use any of the pre-made filter coefficients (or your own) you need:

- An updated version of MiSTer (Main_MiSTer). The first release with support was MiSTer_20181116.
- A supported core. Nearly all cores should have support now, but some Arcade Cores still have not been updated to the newer MiSTer framework as of November 2019.
- Filters. Filters are text files containing the coefficients for a specific interpolation method. All filters must go in the Filters directory on your MiSTer SD card. The most common way to obtain a set of filters is to run the MiSTer updater script (if your MiSTer has network access). You can also download a release ZIP file from this repository or download a copy of this repository and copy the Filters folder over to your SD card by hand.

Filter Releases are here: MiSTer Filters/Gamma Release Folder

- Gamma. Gamma tables are text files containing the R,G,B entries of a gamma lookup table. All gamma LUTs must go in the Gamma directory on your MiSTer SD card. You can obtain gamma tables in the same way as the filters: either using the MiSTer updater script or by copying the Gamma folder from this repository over to your SD card by hand.

Once you have updated MiSTer and Cores and your filter/gamma coefficients in the right place you simply

- Start a supported core
- Go to page 2 of the core's OSD menu and under HDMI Scaler: change the option from "Filter - Internal" to "Filter - Custom"
- The option below "Filter - Custom" should now be enabled and will allow you to choose your filter from the files in your /Filters folder.
- Gamma Tables are implemented the same way and are just below the Filters options on page 2 of the OSD. If you have Filters settings but not Gamma settings then you're using a core that does not yet support Gamma LUTs.

Frequently Asked Questions

Q: What filters should I use?

A: Use "Interpolation (Sharp)" and "SNES Interpolation (Sharp)" if you don't care for scanlines and just want good interpolation.

But all of the filters Filters root are "Recommended Filters". So Interpolation (Sharp), Scanlines (Sharp), Scanlines (Soft), Vertical Scanlines (Sharp), Vertical Scanlines (Soft), LCD (Monochrome), and LCD (Color) are good defaults for most MiSTer cores.

Q: Why would I want to use custom filter coefficients?

A: The default scaling algorithm of MiSTer is Nearest Neighbor/Lanczos and is not ideal for upscaling pixel graphics. The filters allow much better looking scaling and special effects such as scanlines.

Q: Doesn't MiSTer already have scanlines through the "Scandoubler FX" option in the OSD?

A: Yes, but the scanlines available with "Scandoubler FX" are aligned to the scandoubled image and the scanlines through the filter coefficients are aligned to the original pixels in the scanline. So you achieve better scanlines with filter coefficients in most cases.

Q: Where do I get sets of Filter Coefficients or Gamma LUTs?

A: Here. Or the update script which will get them from here. This is the official place to get these filters as of Dec 03, 2018

Q: Why are some filters labeled "SNES"?

A: MiSTer's SNES core outputs a 512 pixel wide image even when the SNES is in it's 256 pixel wide mode. This makes the horizontal interpolation too sharp. The SNES specific filters take this into account.

Q: What do some filters have "NN" appended to the name?

A: Some people set MiSTer to scale to do integer scaling for the vertical upscaling (by setting vscale_mode=1 in mister.ini). If you do this you can use the "NN" filters that only enable interpolation for horizontal scaling.

Q: Where is a filter that does XXXXX?

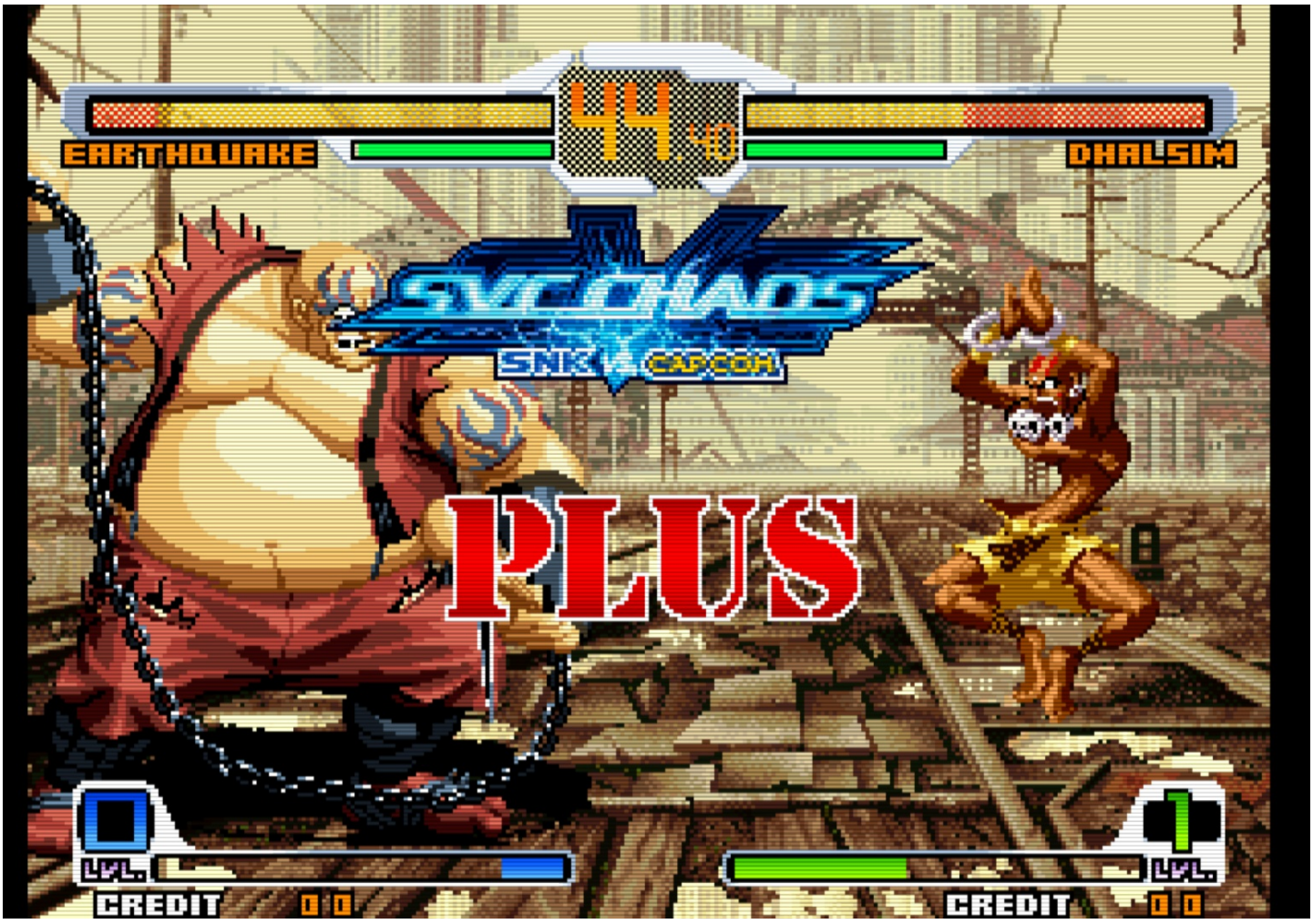
A: Check the subfolders for many filters. Only a few are in the Filters root folder to make choosing a filter easier for newbies and non-technical people. But there are many more to choose from.

What do some filters look like?

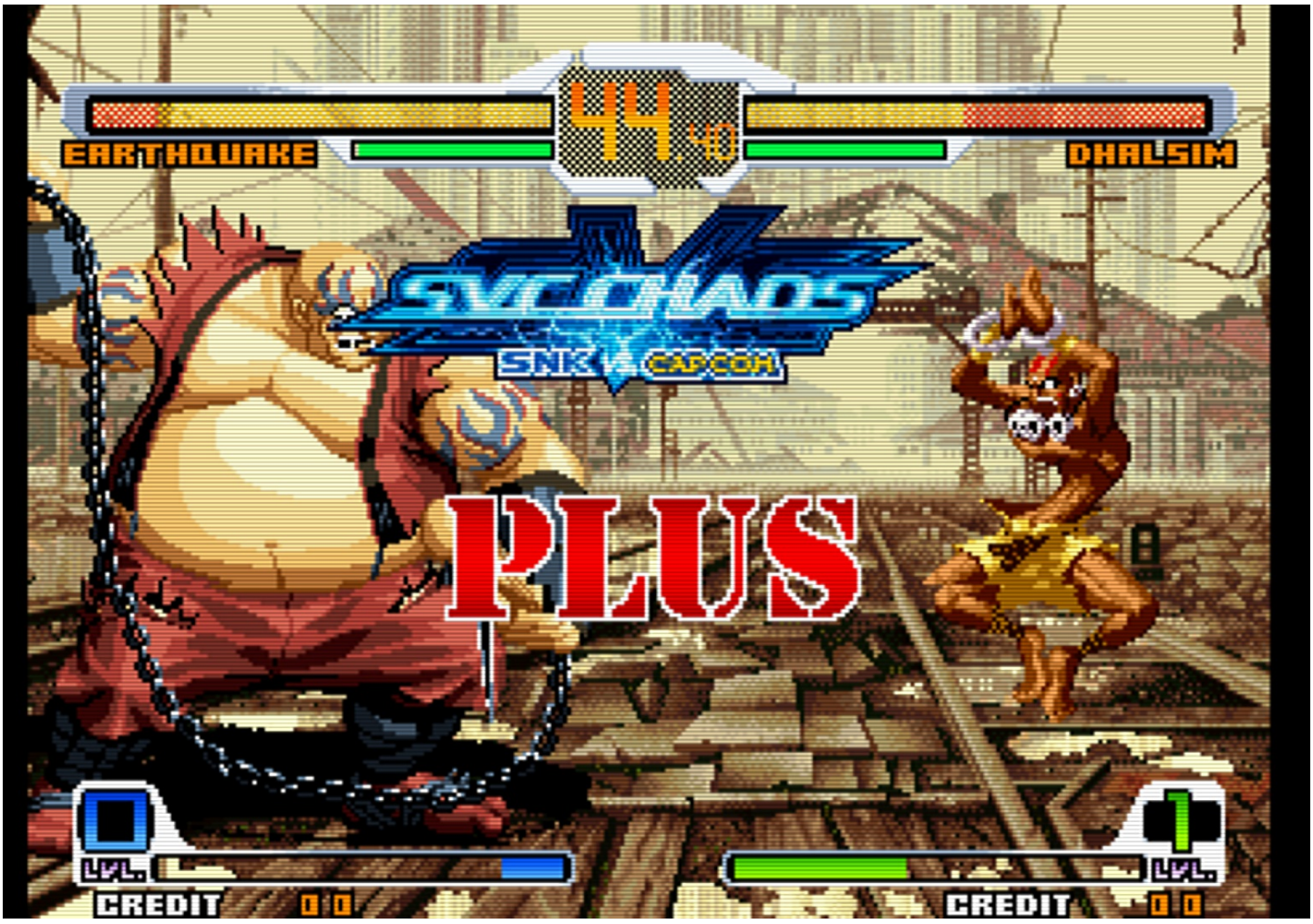
The Filters folder contains a set of recommended filters for general use. There are samples in the Samples folder of this repository. View these images at full size to make a proper evaluation.

Interpolation (Sharp):

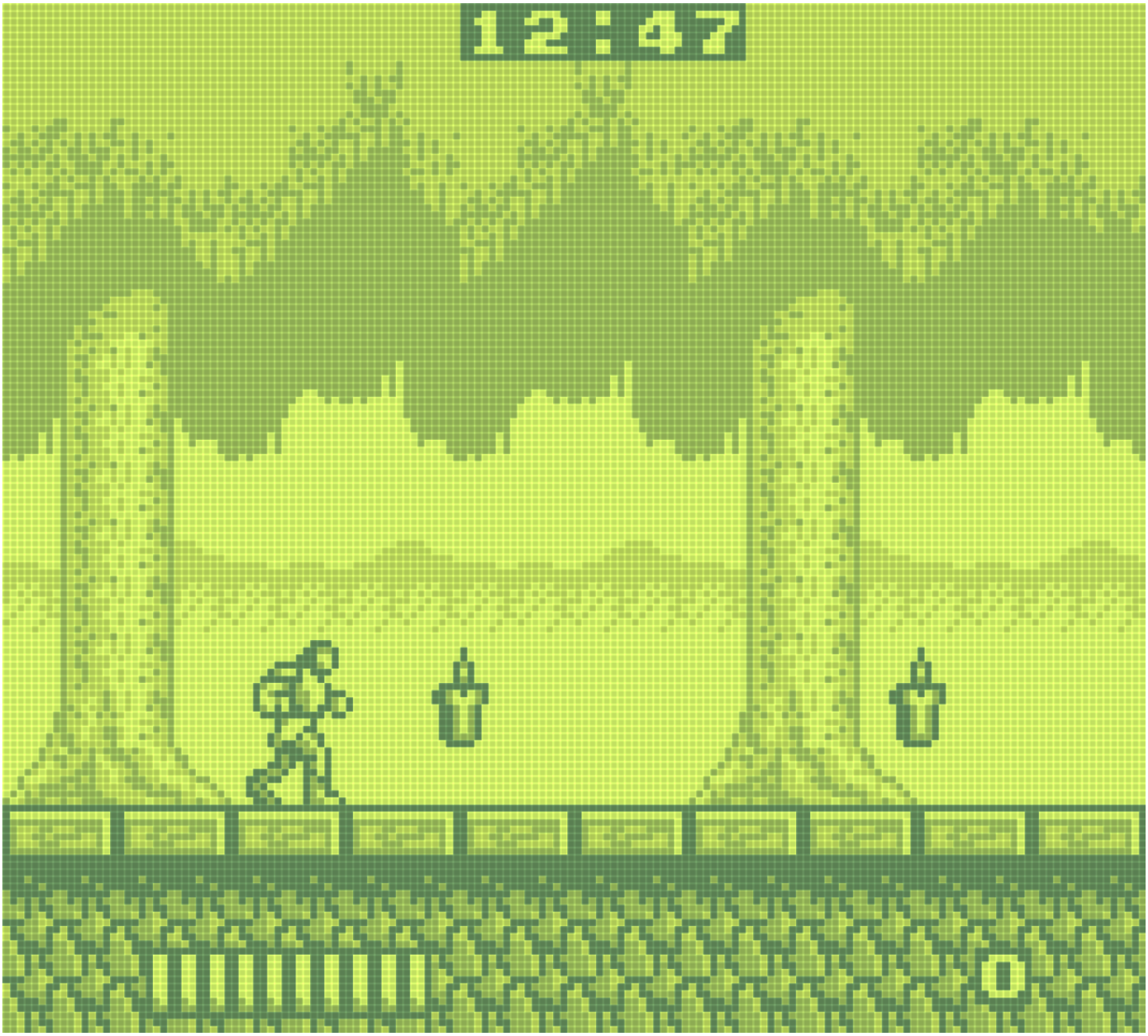
Scanlines (Sharp):



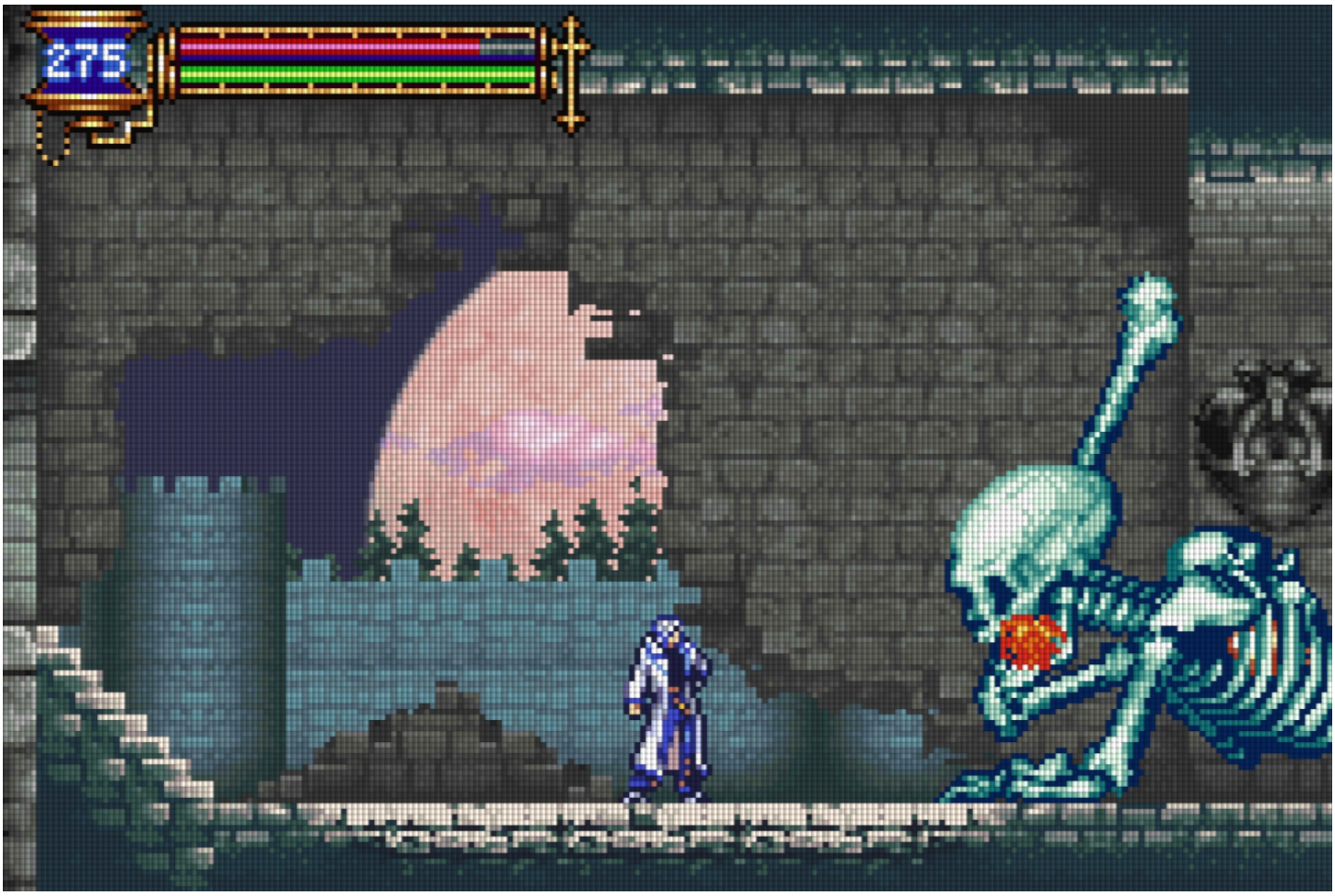
Scanlines (Soft):



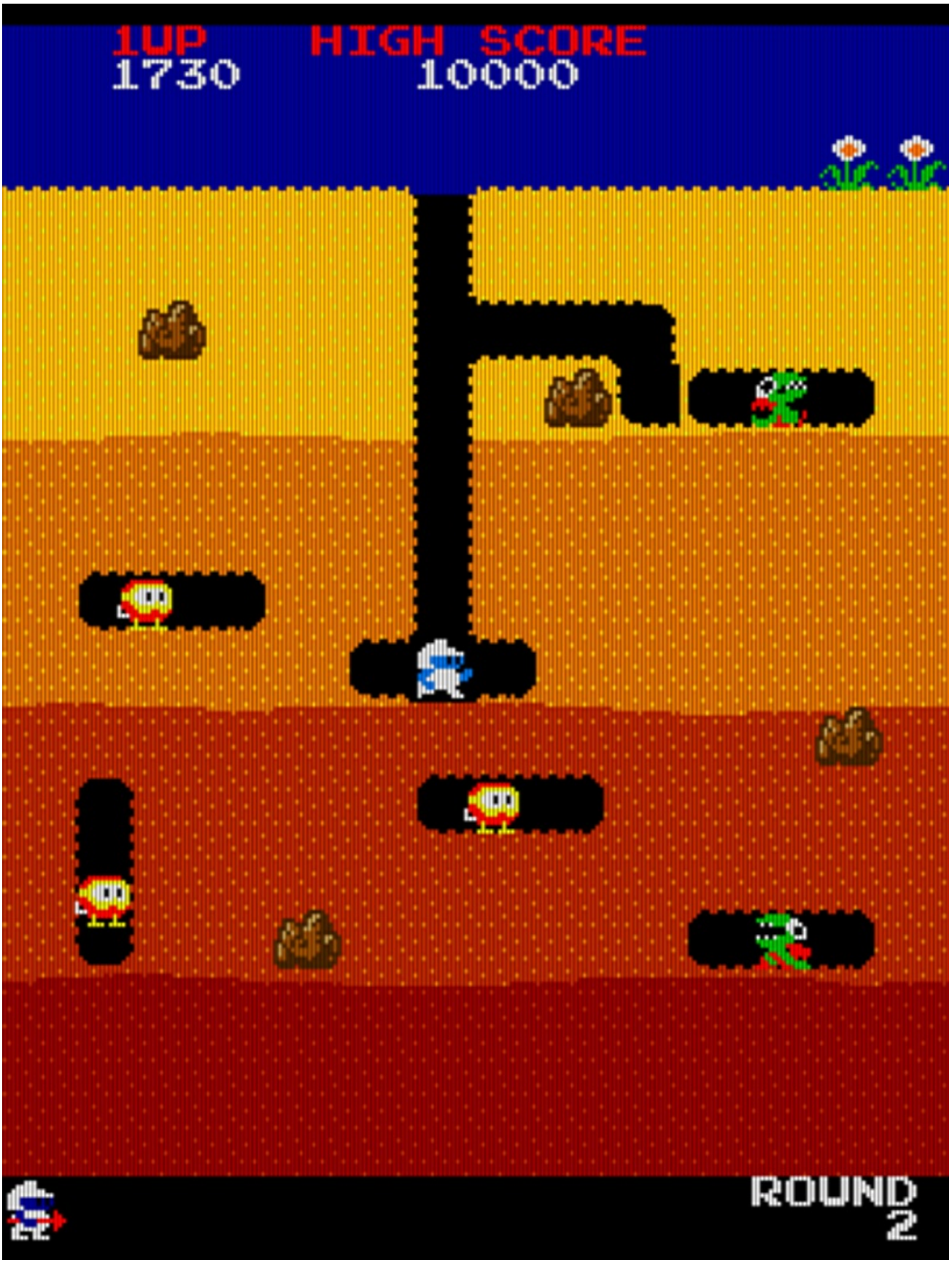
LCD Effect (Monochrome):



LCD Effect (Color):



Vertical Scanlines (Soft):



Technical Information

The commercial VIP scaler implements a generic 4 tap, 16 phase polyphase filter. The open source ASCAL scaler that MiSTer uses now implements the same type of filter for scaling. Details are on page 189 of the VIP scaler docs here: Intel VIP Scaler Doc

The Zipcores Application Notes pdf explains the workings of the filter much better than the ALtera/Intel docs: Zipcores Application Notes

Most of the currently available filter coefficients were generated with the Matlab code here: https://github.com/ghogan42/Filter-Coefficients-For-MiSTer

Tips for understanding the MiSTer filter coefficient text files:

- There are separate coefficients for horizontal and vertical scaling.
- Each row of the Filter Text File list the coefficients for taps T0, T1, T2, T3 for a paricular phase.
- The first row is phase 0 and corresponds to the center of tap T1.
- The ninth row is phase 8 and corresponds to the halfway point between T1 and T2 (so it's the pixel edge between T1 and T2).
- Then last row is phase 15 and corresponds to 15/16th of the way from T1 to T2 (so one phase before the center T2).

Sample Coefficient Set. Note the following:

- This is a 2 tap filter because we only have non-zero coefficients for taps T1 and T2
- The vertical coefficients don't sum to 128 for the middle phases. Since they sum to less than 128, the output will be darker. That's how scanlines are implemented.

```
# range -128..128
# sum of line must not exceed the range!

# Sharp Bilinear on x-axis and y-axis
# 40% Scanlines on y-axis

# horizontal coefficients
  0, 128,   0,   0
  0, 128,   0,   0
  0, 127,   1,   0
  0, 125,   3,   0
  0, 120,   8,   0
  0, 112,  16,   0
  0, 101,  27,   0
  0,  85,  43,   0
  0,  64,  64,   0
  0,  43,  85,   0
  0,  27, 101,   0
  0,  16, 112,   0
  0,   8, 120,   0
  0,   3, 125,   0
  0,   1, 127,   0
  0,   0, 128,   0

# vertical coefficients
  0, 128,   0,   0
  0, 126,   1,   0
  0, 116,   5,   0
  0, 102,  10,   0
  0,  86,  16,   0
  0,  71,  21,   0
  0,  57,  26,   0
  0,  46,  31,   0
  0,  38,  38,   0
  0,  31,  46,   0
  0,  26,  57,   0
  0,  21,  71,   0
  0,  16,  86,   0
  0,  10, 102,   0
  0,   5, 116,   0
  0,   1, 126,   0
```

Audio Filters

Temporary quick introduction

Cores need to be update with new framework. Cores released from 2020/06/18 and later should support audio filters. Filter setting files should be placed into `Filters_Audio` Each file is simple text file with extension `txt`. Example of filter file:

```
# version
v1

# sampling frequency
# must be multiple of 48000 for best results
7056000

#base gain
0.00000316778645516

#gain scale for X0
2

#gain scale for X1
1

#gain scale for X2
0

#coefficient for Y0
-1.9974813602

#coefficient for Y1
0.9974845280

#coefficient for Y2
0
```

Screenshots

MiSTer has the ability to capture screenshots of a running core. This is useful for many things, including "saving" passwords.

Press **Win** + **Print Screen (⌘ + F13)** to capture a screenshot from the scaled video output. In order to instead grab a screenshot at the core's native video resolution, press **Win** + **Left Shift** + **Print Screen**.

Output files are stored in /media/fat/screenshots/

Desktop Linux

Here is the release with full-featured Desktop Linux with GUI and Update 20180502 (custom video mode support)

Make sure you use MiSTer Linux release from 2018 Apr 7 or later. Extract the archive to the root of MiSTer SD card FAT partition.

Desktop Linux is based on Ubuntu 16.04 with LXDE distributed by Terasic with DE10-nano board. It's light-weight linux. Frame buffer has been completely rewritten with configurable resolution and integrated scaler. Audio output is also implemented unlike original Terasic version. VGA and all MiSTer audio outputs are supported.

How it works? If core (rbf file) has the text file with the same name, then it means MiSTer will reboot and will use additional sets of u-boot commands from that txt file. That's how alternative kernel and linux image are used. It's not really tied to Linux only. If there will be other cores requiring special code on ARM side, then it will be run the same way. Even bare-metal projects can be loaded.

One RBF file can have several configs using RBF as a base name + additional suffix. Several configs are included in release. Video card emulated in FPGA has 2 resolutions:

- HDMI resolution comes from the the board to HDMI and VGA.
- Linux resolution which Linux sees. It will be boxed and then up/down scaled to HDMI resolution.

Thus Linux may have any resolutions up to 1920x1080, including non-standard ones.

All video parameters are passed as kernel parameters in config file (txt). Parameters are:

- altvipfb.video_mode - video mode number. Same as in MiSTer.ini (including custom modes, new format). Default is 1280x720@60
- altvipfb.aspect - set to 0 if you want to stretch Linux resolution to full screen. By default aspect=1. If bgwidth or bgheight are set, then aspect ratio is ignored in favor to explicit box dimensions.
- altvipfb.width - horizontal resolution for Linux. Default is the same as HDMI resolution width.
- altvipfb.height - vertical resolution for Linux. Default is the same as HDMI resolution height.
- altvipfb.format - color format. Default is 8888 i.e. 32bit. Other possible values are 565 and 1555 - both are 16bit colors. 565 is good alternative to 8888. It requires 2 times less memory and works faster than 32bit.
- altvipfb.bgr - set to 1 if you want to swap R and B components. Usually you don't need to use this parameter. Default is 0.
- altvipfb.bgwidth - box width. Default is the same as width.
- altvipfb.bgheight - box height. Default is the same as height.

Box resolution is added around Linux resolution before up/down scaling. It's used to add horizontal/vertical fields in order to correct aspect ratio - i.e. letterboxing. By default Linux resolution will be boxed automatically, so bgwidth/bgheight usually are not required.

Check supplied config files to see how to use the parameters. Leave other text as-is if you don't know what it does. Text file uses Linux line endings, so you need to use compatible text editor if you edit it on Windows PC.

Note: Linux image is mounted as read-write, so avoid from turning off or reset without proper shutdown! I suggest to choose "Reboot" instead of "Shutdown" so MiSTer will reboot into MiSTer menu where you can simply turn off the power. DE10-nano board has no power off feature, so if you choose "Shutdown" it will end by black screen and it will be hard to tell if shutdown procedure is finished already or not. Every reboot from Linux will reboot to Menu core.

MIDI

When running the Minimig and ao486 cores, once a ALSA compatible USB MIDI device is attached, two additional ‘UART Connection’ menu options (‘USBMIDI’ and ‘USBMIDI-38K’) will be available in addition to ‘None’, ‘PPP’ and ‘Console’.

Minimig

‘USBMIDI’ - This option is used with the Amiga / Minimig core. This option sets the UART connection speed to 31250 baud which is the standard MIDI speed.

Many Amiga applications and most games don’t require any additional drivers for MIDI. Some “newer” applications may require the CAMD driver.

Aminet : CAMD

ao486

‘USBMIDI-38K’ - This option is used with the ao486 core. This option sets the UART Connection speed to 38400 baud. (The MIDI speed of 31250 baud is not a standard speed DOS PC UARTs were capable of doing)

While some sequencer applications and Microsoft Windows may support MIDI on the serial port, DOS games typically require a MPU-401 interface which ao486 unfortunately lacks. In lieu of hardware MPU-401 capability the SoftMPU TSR can be used with a good degree of success.

SoftMPU

SoftMPU

SoftMPU requires the QEMM memory manager be installed. For testing QEMM 8.03 was used. QEMM “stealth” option seems to be incompatible with ao486 so it is advisable to skip that part of the optimize process. It’s a good idea to run the QEMM optimize application again after installing SoftMPU (in the AUTOEXEC.BAT) to get as much of the lower 640K conventional RAM free as possible.

Although less common, some DOS games and applications require MPU-401 interrupts. This option can break compatibility with others software not requiring interrupts.

Starting SoftMPU without MPU-401 interrupts:

```
SOFTMPU.EXE /MPU:330 /OUTPUT:COM1
```

Starting SoftMPU with MPU-401 interrupts:

```
SOFTMPU.EXE /SB:220 /IRQ:5 /MPU:330 /OUTPUT:COM1
```

The Rev.0 Roland MT-32 used in testing required the ‘DELAYSYSEX’ switch to prevent buffer overflow for certain games but made Sierra games upload sysex commands excessively slowly. This is not necessary for General MIDI modules and newer revision MT-32s.

```
SOFTMPU.EXE /MPU:330 /DELAYSYSEX /OUTPUT:COM1
```

MidiLink

The 'midilink' daemon currently supports following switches / options:

```
TESTSER  - this option sends a test message to the serial port once
           the daemon is started.

TESTMIDI - this option sends a middle 'c' note to the MIDI device
           once the daemon is started.

QUIET    - this option suppresses MIDI debug output.

38400    - this option sets the serial speed to 38400 baud
           (default is 31250 baud) - used with ao486 core.
```

MidiLink Github

MIDI Adapters reported to work

- Creative EMU XMIDI (Known to mangle SYSEX messages)
- Roland UM-ONE
- M-Audio Midisport Uno

Customizing your setup

The MiSTer menu can be customized in a number of ways.

eg You can add a cool font, or change the background image for the main menu.

To change the default font

Create a /media/fat/fonts folder on your SD card.

Next download the fonts from here - MiSTer Fonts

Copy the pf files downloaded from that url to the /media/fat/fonts folder

Next edit /media/fat/MiSTer.ini

Find the

font= line, and edit to use a new font. Be careful not to add a / before the font folder!

eg

If you want to use the C64 font - change the font line to

font=fonts/Computer_C64.pf

To add a background image

Find a suitable png or jpg file that you like

Copy your desired image to /media/fat/menu.jpg or /media/fat/menu.png (as appropriate)

reboot

Press F1 once rebooted into MiSTer to select your background.

To use multiple background images

There is also a way to have MiSTer choose from a selection of wallpaper images at random. On the root of your micro SD card (`/media/fat`), you can create the following folder:

`wallpapers`

Remove the `menu.jpg` / `menu.png` file (if present) from the root of the micro SD card. Populate the wallpapers folder with your desired images. Press `F1` to cycle through the included default images until you see one of the custom wallpaper images you have added, then leave it on that image. On the next restart, a different image will be chosen at random.

You can also create the following additional folders:

`wallpapers_alt_1` `wallpapers_alt_2` `wallpapers_alt_3`

MiSTer supports up to three additional configuration profiles. The active `MiSTer.ini` or `MiSTer_alt_*.ini` file will determine from which folder the images are selected. If any of your configuration profiles use different resolutions, you can populate the corresponding wallpapers folders with only images of that profile's resolution. Each of these folders can have completely unique multiple images, or you can place just one image in each folder to have greater control over which background is used for certain configuration profiles.

Add your favorite games to the menu

You can make your favorite games accessible in the main menu or inside a folder. Read more in the Loading games via MGL files section.

Arcade Roms and MRA files

Configuring Arcade Roms

MRA Format

Because some arcade boards can change games by just putting in new roms, it made sense to move the RBF files out of sight from the menu list, and browse the MRA files instead. These MRA files specify which RBF file to use, and which mame rom zip files to create on the fly into a rom to pass to the arcade core. They will create the old a.pacman.rom style rom on the fly from mame roms, either merged or non-merged.

Here is an example of where the files might go:

```
/_Arcade/<game name>.mra
/_Arcade/cores/<game rbf>.rbf
/_Arcade/mame/<mame rom>.zip
/_Arcade/hbmame/<hbmame rom>.zip
```

There are other locations for these files based on search paths.

MRA Format

```
<misterromdescription>
  <name>Donkey Kong (US set 1)</name>
  <mratimestamp>201911270000</mratimestamp>
  <mameversion>0216</mameversion>
  <setname>dkong</setname>
  <year>1981</year>
  <manufacturer>Nintendo of America</manufacturer>
  <category>Maze / Monkeys</category>
  <category>Platform</category>
  <category>Platform / Mario Bros.</category>
  <rbf>DonkeyKong</rbf>
<!-- switches element taken from "Pac-Man (Midway).mra"; for demonstration purposes only -->
  <switches default="FF,FF,C9">
    <dip bits="15"    name="Cabinet" ids="Cocktail,Upright"/>
    <dip bits="16,17" name="Coinage" ids="2c/1cr,1c/1cr,1c/2cr,Free Play" values="3,1,2,0"/>
    <dip bits="18,19" name="Lives" ids="1,2,3,5"/>
    <dip bits="20,21" name="Bonus Life After" ids="10000,15000,20000,None"/>
    <dip bits="22"    name="Difficulty" ids="Hard,Normal"/>
  </switches>
  <buttons names="Jump,Start 1P,Start 2P,Coin" default="A,Start,Select,R"/>
<!-- rom index 1 or any other index can pass additional information to a rom.
useful to say this rom is game A or game 1.  Use it in case of multiple games for
the same RBF, ie: Dig Dug 2, Mappy -->
  <rom index="1">
    <part>0A</part>
  </rom>
<!-- romstruct element taken from "Two Tigers.mra"; for demonstration purposes only -->
  <romstruct>
ROM structure
00000 - 0BFFF  48k CPU1
0C000 - 0FFFF  16k CPU2
10000 - 13FFF  16k GFX1
14000 - 1BFFF  32k GFX2
  </romstruct>

<!-- the nvram node is used to download and upload the nvram memory. When the user hits "Save Settings" in the OSD it triggers ioctl_upload. This isn't available in Donkey Kong bec
use it didn't originally have NVRAM -->
<nvram index="4" size="1024"/>



<!-- rom index 0 is the standard rom. The zip will be added to the name inside the part, unless the
part has it's own zip. The md5 will be checked at the end. A file not found error is reported before an md5
error. -->
  <rom index="0" zip="dkong.zip" md5="05fb1dd1ce6a786c538275d5776b1db1" type="merged|nonmerged|split">
    <part crc="ba70b88b" name="c_5et_g.bin"/>
<!-- interleave element taken from "Crater Raider.mra"; for demonstration purposes only -->
    <interleave output="32">
    <part crc="579a8e36" name="crvid.a4"  map="0001"/>
    <part crc="2c2f5b29" name="crvid.a3"  map="0001"/>
    <part crc="5bf954e0" name="crvid.a6"  map="0010"/>
    <part crc="9bdec312" name="crvid.a5"  map="0010"/>
    <part crc="4b913498" name="crvid.a8"  map="0100"/>
    <part crc="9fa307d5" name="crvid.a7"  map="0100"/>
    <part crc="7a22d6bc" name="crvid.a10" map="1000"/>
        <part crc="811f152d" name="crvid.a9"  map="1000"/>
    </interleave>
<!-- begin kill screen fix -->
    <patch offset="0xf7d">
FE 04 38 02 3E 04 47 A7 17 A7 17 A7 17 80 80 C6
28
    </patch>
<!-- end kill screen fix -->
    <part crc="d6412358" name="c-2j.bpr"/>
    <part crc="b869b8f5" zip="another.zip" name="v-5e.bpr"/>
    <part crc="b869b8f5" name="v-5e.bpr" offset="1024" length="1024"/>
    <part repeat="3328">00</part>
    <part>
80 80 80 80 80 80 7F 7F 7F 7F 7F 7F 7F 80 80 80
    </part>
  </rom>
<!-- If the first rom0 fails the md5 checksum, it will go ahead and try again if another entry is present.
Otherwise it will skip the additional entries.
-->
  <rom index="0" zip="dkong.zip" md5="05fb1dd1ce6a786c538275d5776b1db1">
  </rom>
</misterromdescription>
```

When creating a core you can pass additional data in using ioctl_index > 0.

```
// Retrieve Title No.
always @(posedge clk_sys) begin
  if (ioctl_wr & (ioctl_index==1)) tno <= ioctl_dout[3:0];
end
```

Explanation for XML elements and attributes

- **name**: As an element this indicates how the rom should be called. The value should be taken from MAME. As an attribute this indicates an external rom file (or part thereof) that should be loaded by MiSTer.
- **mratimestamp**: This indicates the date on which the *.mra file was created (useful for other users to determine if there is a newer version of the* .mra file available to which they should upgrade). The date format is yyyymmdd.
- **mameversion**: This indicates on which version of a MAME romset the *.mra file is based (which version was used for testing). The dot in MAME's version numbering is omitted.
- **setname**: This indicates the name of the romset used as given by MAME. It is used to overwrite the core ID specified in the *.rbf file so that individual settings can be saved on a per romset basis.
- **year**: This indicates the year the game was released. The format is YYYY and the value should be taken from MAME.
- **manufacturer**: This indicates the manufacturer of the game. The value should be taken from MAME.
- **rbf**: This indicates the filename (sans path and extension) of the core that should be used to run the game.
- **buttons**: This indicates the default button mapping for the game. As the name implies, it only implies to buttons, not to directional controls. The **names** attribute specifies how the buttons are called in the MiSTer OSD while the **default** attribute specifies how the buttons are mapped to the virtual MiSTer gamepad.
- **switches**: This is to define the dip switches present on the arcade board. As this is a bit more complex, there is a specific section for it below.
- **nvram**: This tells MiSTer to save the NVRAM data on Save Settings in the OSD. If an NVRAM was saved, it will be sent to the core as well
- **romstruct**: This is to give information about the rom structure, how the rom memory is mapped. Information contained herein is intended as a comment for other developers and not actually used by MiSTer.
- **remark**: This element is intended for general remarks. Information contained herein is intended as a comment for other developers and not actually used by MiSTer.
- **rom**: Part, patch and interleave elements must have a rom element as a parent element. An **index** attribute with a value of 1 is used to pass information to multigame cores, which machine configuration to use. For passing actual rom data you need an index value of 0. The **md5** attribute is used to specify the md5 checksum. The md5 checksum is calculated and checked by MiSTer after the whole rom is created according to the given *.mra file. You can use the Linux console to find out the checksum calculated by MiSTer when it starts the core (the command line to do so is '/media/fat/MiSTer rbffilename mrafilename'). Roms whose checksum differ from the one specified in the* .mra file won't load. Specifying the md5 checksum as “none” will disable this check.
- **part**: The part element is only allowed inside a rom element. It is used for specifying a part of the rom. The content of a part can either be embedded as a hex value into the *.mra file itself (if copyright law allows) or there can be a reference to an external file via the name attribute. If the external file is in a different zip file than the parent rom element, the filename of the zip file must be specified via the **zip** attribute.
  - The **crc** attribute specifies the CRC32 checksum of the respective part (format: eight hex digits). If a crc value is specified, MiSTer will try to select the appropriate file by crc (and only revert to the filename specified by the attribute name, if the crc value does not match). This increases compatibility of the *.mra file with different MAME versions of the romset. As crc values are only relevant for external files, a crc value should only be given to a part element when there is a reference to an external file.
  - The **repeat** attribute only applies to embedded parts and indicates for how long (not how many times) the sequence should be repeated. By default decimal numbers are assumed. Hex numbers must be preceded with "0x".
  - Inside the rom element, the **offset** and **length** attributes only apply to external files. To use the offset attribute for internal data, the patch element must be used. Offset indicates the location inside the file from where MiSTer should start reading and length indicates the length of the sequence that MiSTer should be reading. By default decimal numbers are assumed. Hex numbers must be preceded with "0x".
  - The **map** attribute specifies to only read certain parts of the corresponding part element. For example a value of "0001" (binary) means

to read only every fourth byte while a value of "1010" (binary) means to read only every first and third byte. To byteswap a part you need to set a map attribute value of 21 (decimal) and enclose the respective part element in an interleave tag.

- **patch**: The patch element is only allowed inside a rom element. It is applied after the whole rom is created from its parts and overwrites the content of the rom with the content of the patch element, starting from the specified offset and for the length of the patch element's content.
- **interleave**: The interleave element is only allowed inside a rom element and can only be used as parent element to one or more part elements. Together with the mandatory **output** attribute the interleave element can be used to specify that the children part elements should be read in a certain way. For example an output-value of "32" (decimal) indicates that the children part elements ROM data should be treated as 32 bit.

Dip Switches

Dip switch support in the latest version of MRA is used instead of the status bits. The DIP config str is listed in the core, and the core is responsible for reading the up to 64bits of dip data that is sent via ioctl_index 254.

```
reg [7:0] sw[8];
always @(posedge clk_sys) if (ioctl_wr && (ioctl_index==254) && !ioctl_addr[24:3]) sw[ioctl_addr[2:0]] <= ioctl_dout;
```

Switches is the dip switch setting. The **default** are the default bytes. These are used so you can default the arcade into the proper factory settings. This is useful when the factory settings aren't all OFF/OFF/OFF. Note that in the **default** hexadecimal string, the leftmost byte refers to DIP bits 7:0, the next byte to 15:8 and so on. The most significant byte thus occupies the rightmost part of the string.

```
<switches default="FF,FF,C9">
```

The value C9 will apply to DIP bits 23:16.

The dip tag let's you put in a dip switch entry. The bit number (starting at 0) is based on the way the core was written. Often MAME source code can be used to understand the bits. The numbering will depend on the rom used, the arcade core, and how things are laid out.

- **bits**: bit number to fill out sent to the core in ioctl_data
- **name**: title for the OSD
- **ids**: title for each option in the OSD
- **values**: if you want the values to be different than 0,1,2,3 you can reorder them

```
<dip bits="16,17" name="Coinage" ids="2c/1cr,1c/1cr,1c/2cr,Free Play" values="3,1,2,0"/>
```

Loading games via MGL files

!!! NEW --- Under construction !!!

MGL files run games automatically

You can add your favorite games to the main menu (below the Arcade, Computer, Console etc. folders) - or - e.g. inside a folder for instance called 'Favorites'. It needs to be named _Favorites in the file system to show up in the menu.

Each of your favorite games must have an MGL file to be accessible in the menu.

At the time of writing (late February 2022) MGL is not supported on the Atari ST, Minimig, Archie and Sharp cores.

MGL Format

Examples:

```
<mistergamedescription>
   <rbf>_console/snes</rbf>
   <file delay="2" type="f" index="0" path="dummy.snes"/>
</mistergamedescription>
```

```
<mistergamedescription>
   <rbf>_console/gameboy</rbf>
   <file delay="1" type="f" index="1" path="dummy.gbc"/>
</mistergamedescription>
```

```
<mistergamedescription>
   <rbf>_Computer/C64</rbf>
   <file delay="1" type="f" index="1" path="dummy.prg"/>
</mistergamedescription>
```

- `<rbf>`: path to the core and its name. If the core is outside the _console folder just its name.
- `<file delay=` amount of seconds to wait before load/mount (e.g. 1:turbografx16, neogeo, gba, gameboy, genesis, c64 and 2:snes).
- `type=` f: load file. s: mount
- `index=` 0 for most console cores but 1 for e.g. the NeoGeo and Gameboy cores.
- `path=` path to game file inside the core's games folder. E.g. Puzzles/Dummy.bin

All parameters must be present.

If a core pops up the OSD menu after loading the game or the game doesn't load try increasing the delay. It could also be due to wrong the index number, file type or game file path so check those too.

Until a complete **list of cores and their index numbers and file types** is ready you can test different index numbers and the file types listed above

... or look at the code for each core here on Github. E.g. for the **TurboGraphics** core look at the file **TurboGrafx16.sv**

Find a section that starts with `parameter CONF_STR` and in that section look for the ROM type you are loading/mounting.

In this case it will be `"S0,CUECHD,Insert CD;"` Notice the S0, that is the "s" type with index of 0.

Addons overview

MiSTer FPGA Addons

While the the MiSTer platform can be used with just the basic setup mentioned [here], its features can be greatly expanded with the use of additional expansions.

There are several options available and each has its own page. Here is a brief overview of each addon and a link to its page for further information.

- SDRAM Module
- IO board
- USB Hub
- RTC Board
- ADC-in)
- Analog video output compatibility
- Direct Video
- Case

How to get boards?

When you've gotten your DE10-Nano FPGA board, you will probably want to have the expansions like the SDRAM or IO board. There are two ways to do that:

1. Buy

Check the sale threads:

- Official Add-On Sellers **NOTE: If you will find some sellers not mentioned in that thread, please post there the link and ask before purchase. There are some scammers who improperly solder the boards and sell even with higher price than you can get on the forum.**
- In general, it's best to prefer buyers that have been around for some time, and have some track record. Many are members of the forums and discord and you can find them easily there. It is also recommended to only buy official versions of the addons first, since MiSTer upgrades are not guaranteed to be compatible with unofficial hardware.

2. Do It Yourself

If you are a bit technically gifted or just want to learn how to assemble PCBs by yourself, then head over to the "Assembly (DIY)" guides in the wiki sidebar and make your own boards! You can discuss and share your experiences with us here:

- Sub-forum about Add-ons

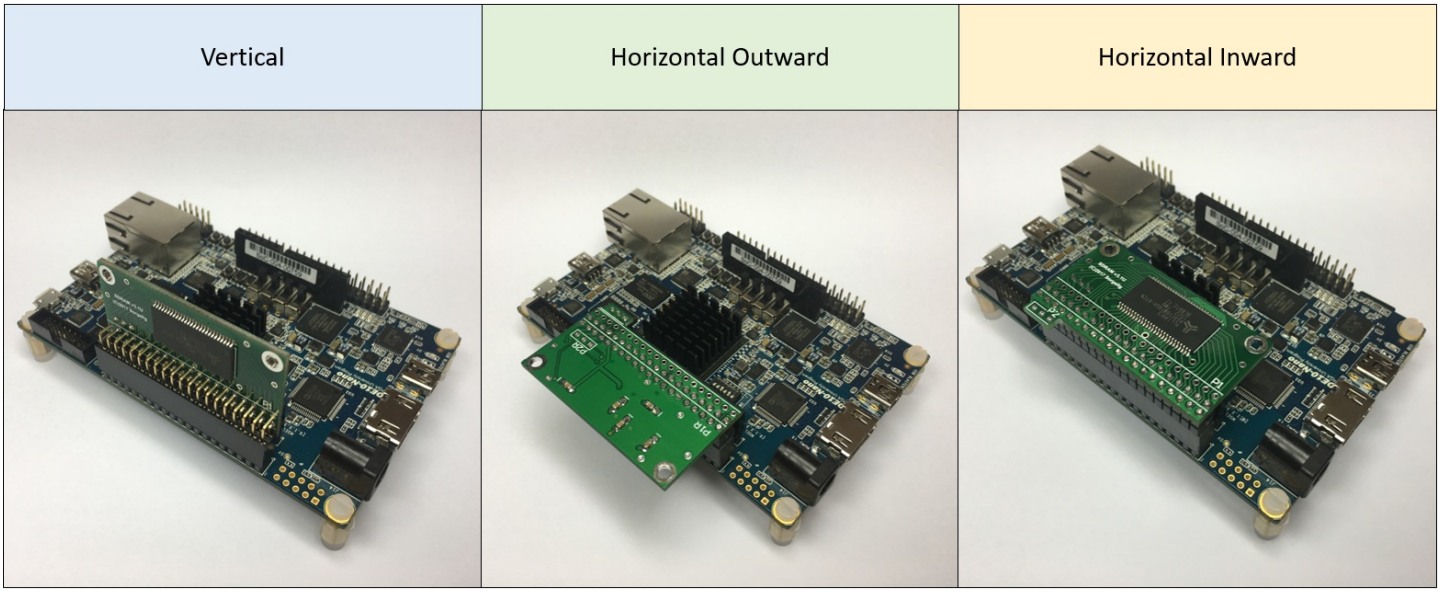
SDRAM Board

Actual SDRAM Board Revisions: **XS v2.2** and **XSD v2.5**

The MiSTer SDRAM Board, although optional, is considered an essential expansion board for the DE10-Nano FPGA board. The SDRAM is required by most cores of the MiSTer platform.

SDRAM Board Types

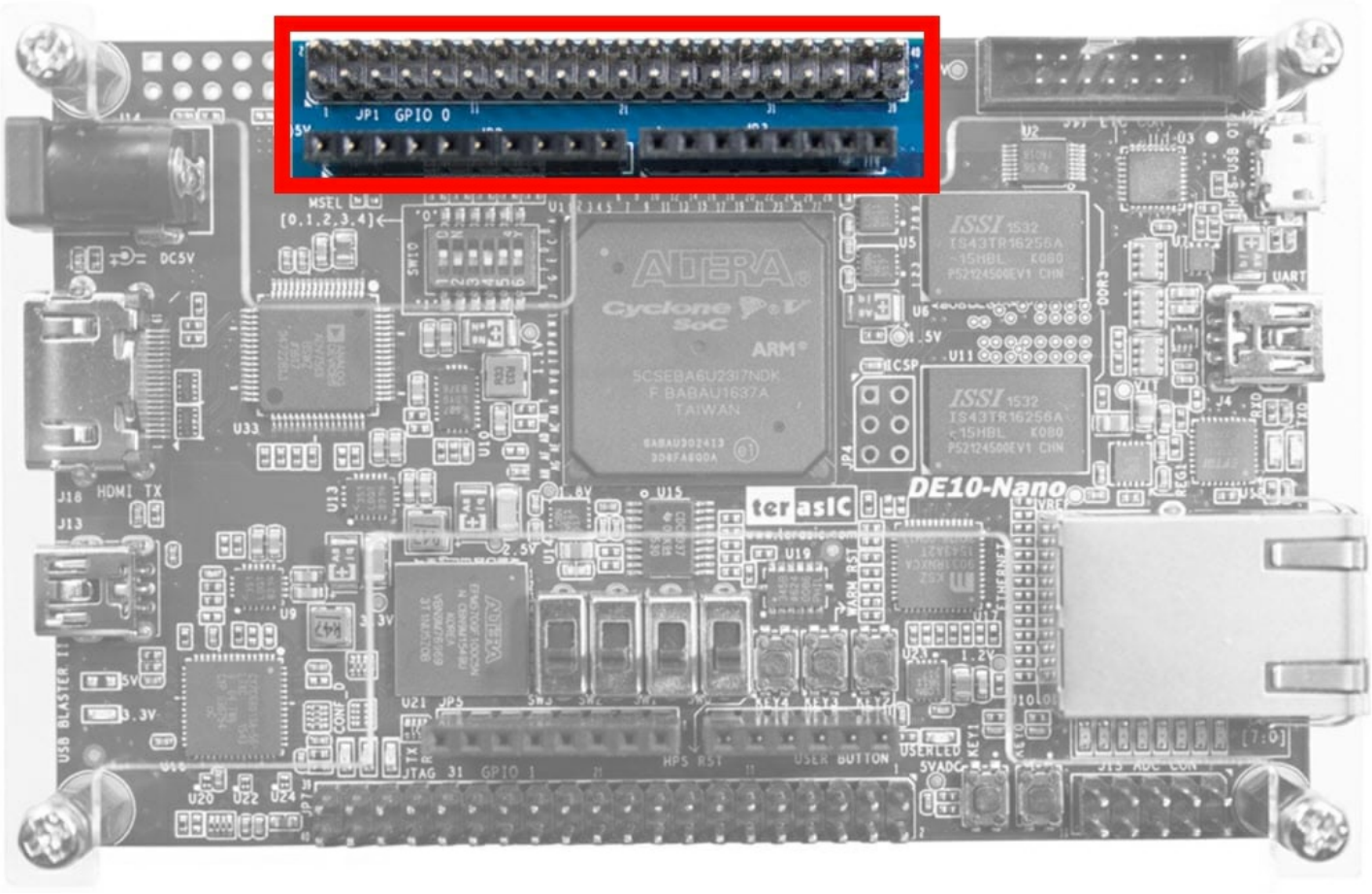
SDRAM Board Universal:



	Advantages	Disadvantages
Vertical	Doesn't increase horizontal dimensions. Doesn't cover and allows better cooling for FPGA chip. Doesn't block Arduino GPIO - compatible with future or custom expansions. Easy to attach/detach.	Slightly less maximum working frequency.
Horizontal Outward	Doesn't cover and allows better cooling for FPGA chip. Doesn't block Arduino GPIO - compatible with future or custom expansions. Higher maximum working frequency.	Increases horizontal dimensions.
Horizontal Inward	Doesn't increase neither horizontal nor vertical dimensions. Higher maximum working frequency.	Blocks Arduino GPIOs - incompatible with future or custom expansions (PCB v3.2 solves this issue). Covers FPGA chip and makes cooling harder. Temperature condition is quite bad. **Not compatible with I/O board v4 and newer**.

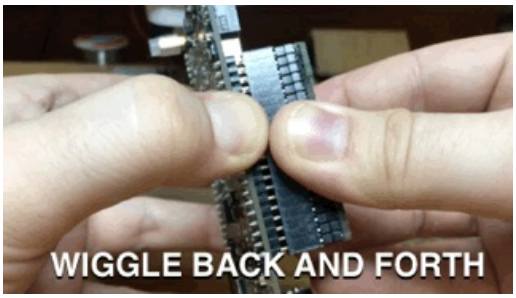
Install the SDRAM Board

The SDRAM Board will be inserted into the GPIO 0 Header of the DE10-Nano Board. Three additional pins will be inserted into the Arduino header JP3



When Plugging in the SDRAM Board, make sure to support the DE10-Nano from beneath with your thumbs. Forcing in the SDRAM Board without support can bend the DE10-Nano board and permanently damage it.

Removing the SDRAM Board can be tricky for some GPIO connectors and just pulling won't do it sometimes. For very tight connectors, it is recommended to wiggle the SDRAM Board back and forth to remove the connectors slowly, bit-by-bit. Just pulling with force will often bend the GPIO header.



Cores that use SDRAM

Here is a list of cores that may take advantage of or require an SDRAM Add-On Board in order to function:

Name	SDRAM Speed (MHz)	Comments
Boot Menu	100	Doesn't require the SDRAM board but will use it if found. It erases the whole SDRAM (and DDR) memory while running
Amiga	57	
Amstrad-PCW		
Amstrad	64	
Arcade Astrocade	57	Only for samples in Seawolf II and Gorf (with speech)
Arcade Cave		
Arcade Cosmic		
Arcade Jackal		
Arcade Moon Patrol		
Arcade Scramble		
Arcade Space Invaders		
Arcade Tecmo		
Arcade VBall		
Arcade Zaxxon		
Archie	126	
Atari 800	57	Only for memory config >320KB or Cartridge usage
Atari ST	96	
BK0011M	96	
ColecoVision	43	
Commodore 64	64	
Gameboy	66	
GBA	100	Optional, Requires 33.5MB for 32MB games; will automatically use the onboard DDR3 RAM instead if SDRAM is not large enough
Genesis	107	Optional
MacPlus	65	
Mega CD	107	
MSX	85.9	
NeoGeo	96	Requires up to 100MB of SDRAM depending on game
NES	85.9	
QL	84	

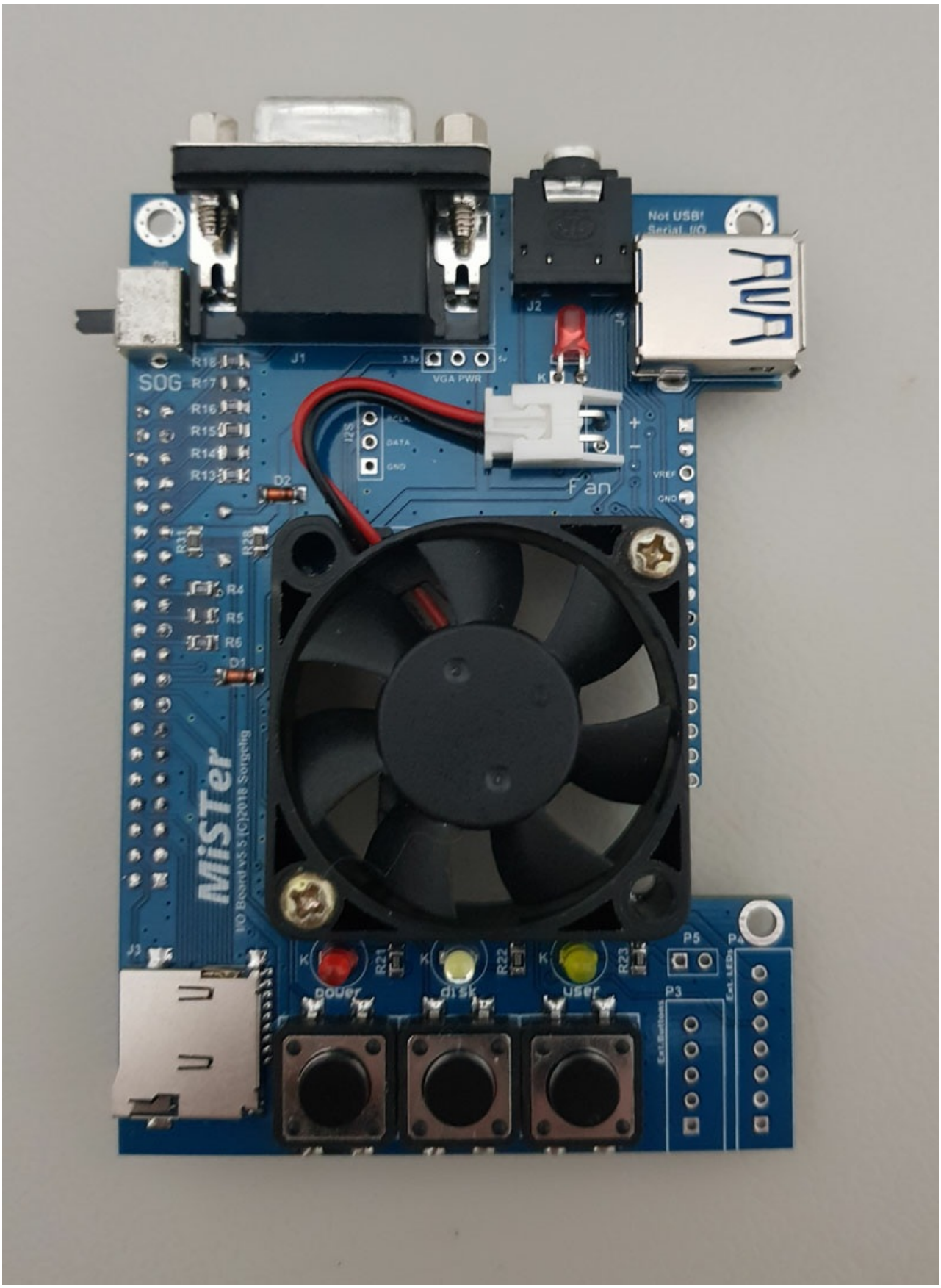
Name	SDRAM Speed (MHz)	Comments
SAM Coupe	96	
Sega Master System	96	
SNES	85	
TSConf	84	
TurboGrafx16	86	Optional
X68000	80	
ZX Spectrum	112	

IO Board

Current Board Revision: 6.1 and digital IO 1.2

The MiSTer IO Board is an **optional** expansion. It adds the following features to the MiSTer Platform:

- VGA Connector for Analog Video output. Capable of RGBs, YPbPr and standard VGA. Check Analog video output compatibility
- 3.5mm Audio Jack with TOSLink
- 3 Buttons
- 3 Status LEDs
- Secondary SD card (for some cores)
- FAN for cooling the FPGA
- Expansion connector (in the form of USB 3.0 connector, but it's not USB!)
- Additional connectors to integrate the MiSTer into cases.


Not USB!
Serial I/O
J2
J1
VGA PWR
SOG
I2S
BCLK
DATA
GND
Fan
VREF
GND
D2
D1
R4
R5
R6
MiSTer
I/O Board v3.5 (C)2018 Sorgelig
J3
power
disk
user
P5
P4
P3
Ext. Buttons
Ext. LEDs

Secondary SD card

Introduction

Some cores require low-level SD card access. The primary SD card cannot provide low-level access due to an incompatible layout. It is also locked by Linux system from direct access. Thus, I/O board v5.x provides the secondary SD card which is directly connected to FPGA. Linux system doesn't see this card. The layout of the secondary SD card fully depends on core requirements. Usually, it's simple FAT16 format, but some cores may require custom formatting.

Schematics

Secondary SD card is a simple direct connection between SD connector and DE10-nano GPIOs. It's possible to solder by yourself if whole I/O Board is not required (or older I/O Board is used). Check the schematic if you want to connect it by yourself.

Cores with Secondary SD card support

Following cores support (but not require, see VHD section) the secondary SD card:

- BBC Micro
- MSX
- Sinclair QL - SD support hasn't been confirmed yet - requires specific ROM. Current ROM doesn't use SD card.
- TSConf

Following cores require the secondary SD card:

- Sharp X68000

Use VHD files instead of secondary SD card

The cores where secondary SD card is optional require VHD image file to replace the SD card in order to work. VHD image is a simple dump of SD card. You can prepare a special SD card for specific core and then make a full backup using a utility such as **Win32 Disk Imager**. You even can create VHD file without SD card at all. In this case, you can create a smaller VHD file with suitable size in such application as **WinImage** - save it into IMA format and then rename to VHD.

Notes about VHD format

- Usually FAT16 is required, but some cores can support FAT32 as well.
- Content of VHD file should be exactly as it is originally required for Secondary SD card.
- Most (but not all) cores support both non-MBR and MBR types of images. **MSX** core is known to support only MBR images. Tools like WinImage create only non-MBR images but may open and edit MBR images created somewhere else (like in disk dumper tool from a real prepared SD card).

How to use VHD files

You can make the image which will be **mounted automatically** at the start of the core. So, it will look like you have SD card connected at the start. For automatic mounting, you need to name the VHD as **CoreID.VHD** and place together with RBF file or create the folder **CoreID** and put BOOT.VHD image in that folder. The second option is convenient if you have several VHD files for the core, while the first option is more convenient if only a single file is used. **CoreID** is the name you see in core's OSD. For example, for Atari 800 core, the auto mounted name will be Atari800.vhd

How to use Secondary SD card and VHD

VHD file has priority over a real secondary SD card if mounted. Unmount the VHD (press BACKSPACE while choosing VHD file in OSD) in order to use the Secondary SD card.

Direct Video

Direct Video

Since autumn of 2019 there is a method for outputting analog video called *Direct Video* that doesn't require you to install the IO Board in your MiSTer. This new method offers **same** minimal latency to the VGA port from the IO Board and even better color depth in some cores. To enjoy it, you just need to activate the feature in your mister.ini file and attach a DAC to your HDMI port.

You may use an HDMI-to-VGA converter similar to this one.

NOTE: as of 2021/06/27 it seems very hard to find devices which are compatible with VGA to SCART adapters. Devices which work with VGA are still easy to find. More info here: https://github.com/MiSTer-devel/Main_MiSTer/issues/410

Compatibility

Direct Video is compatible with most current cores and will be supported in all future cores coming to MiSTer.

RGBs, RGsB and YPbPr are supported, although YPbPr has less display compatibility in *Direct Video* mode compared to YPbPr from the IO Board.

Note that output resolution and format can vary from core to core. See analog video output compatibility for more information.

How to Use

First you need a simple DAC with VGA output. Those based on the chip AG6200/AG6201 (like the one in the picture) are proven to work fine and are fairly inexpensive and easy to find. Also many of them output analog audio too. The only thing we recommend is to not hotplug them or pull them out from your MiSTer while it is still **ON**, since you might damage your HDMI port in the process.

Then you need to add the following line to your mister.ini file:

```
direct_video=1
```

This activates the *Direct Video* feature when the system starts. Once it's enabled, it allows your HDMI port to output the raw and unprocessed digital audio/video signal from the loaded core, which is consumed by your DAC. MiSTer will not work with your HD TV or monitor while this mode is enabled, since they require a standard HDMI signal to work, and the zero-lag *Direct Video* signal **IS NOT** standard HDMI.

The resulting analog video signal from your DAC should be compatible with standard CRTs if the core loaded supports standard CRT refresh frequencies (see analog video compatibility for more details). But there is still a bit of additional configuration you need to do depending on which kind of analog video signal you want to use.

Setup for RGB signals

For analog RGB output, you'll want to enable composite sync on the HSync signal in your mister.ini (`composite_sync=1`). After that, you are good to plug in an RGB-compatible VGA-SCART cable to your TV.

For optimal results, Retro Access sells SCART and BNC cables specifically for use with MiSTer.

DISCLAIMER: Please be aware that many VGA DACs may output a sync signal at **3 volts *or more*** on pin 13 (HSYNC/CSYNC) even when being used in *Direct Video* mode! You will need a 470 ohm resistor on the corresponding SCART sync pin to attenuate this signal to levels that are safe for all video equipment. A variety of professional video equipment can have wide tolerances for higher voltage sync signals but other devices, especially external scalers like the OSSC or XRGB Mini Framemeister, can be worn out and reduced to a non-working state by repeated

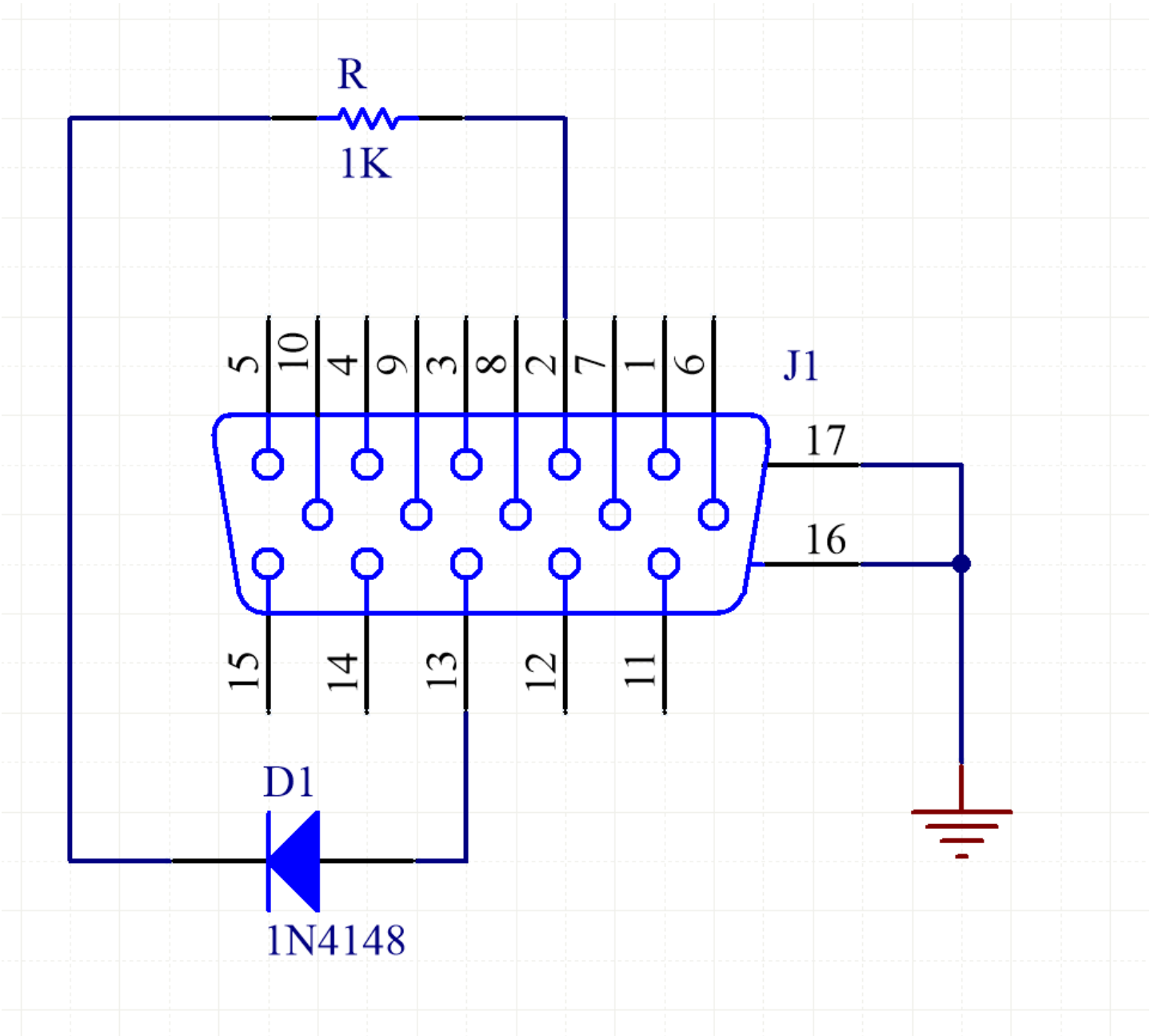
use with higher-voltage sync signals. Unless you are 100% certain your equipment can tolerate higher voltage signals, it is *strongly* recommended that you either purchase a cable that has a 470 ohm resistor inline or add one yourself. Retro Access MiSTer SCART cables, for example, come with a resistor inline by default.

Setup for YPbPr signals

YPbPr - also known as Component Video - is available in *Direct Video* mode but has limited compatibility. This is due to limitations of HDMI-to-VGA DACs, which were not meant to produce signals in the YPbPr color space in the first place, resulting in signals that are slightly out-of-spec. It is possible that your display will accept the *Direct Video* YPbPr signal with no issue, but it may also appear bright pink due to the way many displays process such signals. For higher YPbPr compatibility you may prefer RGB mode with an external RGB-to-YPbPr transcoder instead, or YPbPr via the IO Board.

To use YPbPr in *Direct Video* mode, you'll need to enable `composite_sync` and `ypbpr` in your mister.ini file.

You'll also need to add a sync-on-green circuit on your VGA connection. Sync-on-green circuits can be very simple; you just need a diode (1N4148) and a 1k resistor.



Connection: from HSync -> anode of diode, cathode of diode -> resistor, resistor to Green signal.

Doubling frequency

This is a handy way to use those modern monitors that have VGA input but are typically not compatible with 15KHz signals, and require 31KHz analog video. In case you need it, you just have to set `forced_scandoubler=1` in your mister.ini to turn a 15KHz signal into a 31Khz one (See which cores output 15KHz video here).

Color Depth

Direct Video produces a 24bit color signal, which is superior to the 18bit signal coming from the IO Board. However, few cores use 24bit color, so the improved color depth may not make much of an impact depending on the cores you run.

Also, most cheap DACs, like the ones based on the chip AG6200, don't produce Full Range RGB. They instead produce Limited Range RGB (16-235) or much more commonly a nonspec Limited Range variant (16-255). In order to compensate for this, you may want to configure the `hdmi_limited` option in your mister.ini to adjust the signal being sent to your DAC.

- Full Range (0-255): `hdmi_limited=0`.
- Limited Range (16-235): `hdmi_limited=1`.
- Limited Range common DAC variant (16-255): `hdmi_limited=2`.

Analog video output compatibility

Note

The VGA-style HD DB15 port on the IO board is intended for analog video output and **does not explicitly output VGA or any one standard.**

This page is titled to reflect this, as individual cores and configuration can vary. The default configuration is to output non-scaled video in an RGBHV mode. MiSTer can be configured to enable scaling, output RGBs, RGsB or component as well over this port. Native resolutions here denote video signals generated by the core by default and may not match original hardware. Display output frequencies and resolutions tend to change over a core's development. Some cores output multiple resolutions and formats, which may require a more general classification.

The primary display output of the DE10 nano is HDMI, so HDMI is often the initial target for video output in most cores.

Table of cores and resolutions supported over the RGB/VGA output

Consoles

Core	Version	Native Resolution	Frequency (Horizontal, Vertical)	Notes
Atari2600	20181214	240p	15.40kHz, 59.4Hz	A startup rom must be placed in the core
Astrocade	20181224	240p	15.70kHz, 59.9Hz	
Atari5200	20180819	240p	15.62kHz, 59.9hz	
Colecovision	20181130	240p	15.49kHz, 59.7Hz	
Gameboy	20200331	240p	15.77kHz, 59.7Hz	Enable "Stabilize video" to prevent sync issues on blank screens
GBA	20200409	240p	15.77kHz, 59.7Hz	
Genesis	20200502	240p	15.72kHz, 59.9Hz	
NeoGeo	20200325	240p	15.63kHz, 59.2Hz	
NES	20200308	240p	15.75kHz, 60.1Hz	
Odyssey2	20181221	240p	15.61kHz, 60.1Hz	
SMS	20200309	240p	15.7kHz, 59.9Hz	
SNES	20200306	240p	15.75kHz, 60.1Hz	
TurboGrafx16 (PC Engine)	20181231	240p	15.56kHz, 59.7Hz	
Vectrex	20180616	VGA	44.96kHz, 60.0Hz	

Computers

Core	Version	Native Resolution	Frequency (Horizontal, Vertical)	Notes
Acorn Archimedes	20180519	240p	16.22kHz, 52.2Hz	
Altair 8800	20181113	VGA	45.03kHz, 60.0Hz	
Amiga	20181216	240p	15.68kHz, 50.4Hz	
Amstrad CPC 6128	2018084	240p	15.63kHz, 50.25Hz	boot rom needed
486	20181215	VGA	44.95kHz, 60.0Hz	
Apogee	20180305	240p	15.44kHz, 50.0Hz	
Apple II+	20180308	VGA	30.62kHz, 58.5Hz	

Apple Macintosh Plus	20180305	Mac	22.06kHz, 60.1Hz	
Aquarius	20180429	240p	15.49kHz, 59.6Hz	
Atari 800	20180812	240p	15.63kHz, 59.9Hz	
Atari ST	1	VGA	31.13kHz, 49.9Hz	
BBC Micro	20180521	?	62.5kHz, 62.5kHz	
BK0011M	20180901	240p	15.82kHz, 48.82Hz	
Commodore 64	20180831	240p	15.31kHz, 58.4Hz	
Commodore 16	20180902	240p	15.50kHz, 49.8Hz	
Commodore PET	20180305	240p	15.49kHz, 59.6Hz	
Commodore VIC-20	20180926	240p	15.54kHz, 59.8Hz	
PDP-1	20190101	1280x1024	63.80kHz, 59.9Hz	
MSX	20180520	240p	15.62kHz, 59.9Hz	
MultiComp	20180629	VGA	30.99kHz, 59.6Hz	
Sharp MZ Series	20180927	240p	15.01kHz, 58.0Hz	no HDMI output
Sinclair QL	20180305	240p	15.74kHz, 60.3Hz	
Specialist/MX	20180305	yes	15.61kHz.50.0Hz	
Tandy CoCo3	091918a	VGA	31.28kHz, 59.9Hz	
TI-99/4A	20180527	240p	15.38kHz, 59.88Hz	A startup rom must be placed in the core
TSConf	20180901	240p	15.51kHz, 50.0Hz	
Vector 06C	20180304	240p	15.51kHz, 50.0Hz	
X68000	20171103	NA	NA	
ZX Spectrum	20181231	240p	15.51kHz, 48.8Hz	
ZX81	20180903	240p	15.62kHz, 59.9Hz	

Arcade

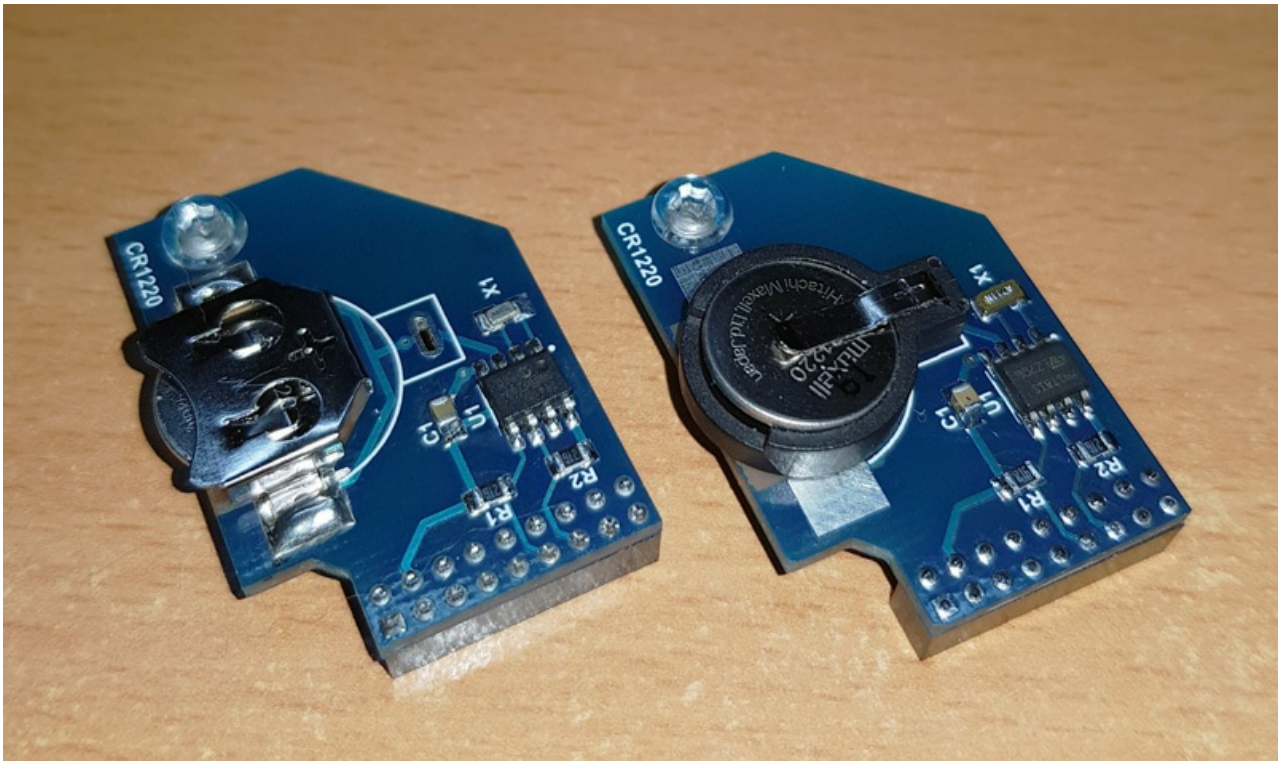
Core	Version	Native Resolution	Frequency (Horizontal, Vertical)	Notes
1942	20180313	VGA	37.77kHz, 60.3Hz	TATE, image squished over HDMI
Alibaba	20180313	240p	15.49kHz, 59.1Hz	
Amidar	20180313	240p	15.86kHz, 60.6Hz	
AzurianAttack	20180313	240p	15.86kHz, 60.6Hz	
Bagman	20180313	240p	15.86kHz, 60.6Hz	
BlackHole	20180313	240p	15.86kHz, 60.6Hz	
BombJack	20180313	240p	15.49kHz, 59.1Hz	
Bubbles	20181217	240p	15.49kHz, 59.1Hz	
BurgerTime	20180313	240p	15.49kHz, 59.1Hz	
BurningRubber	20180313	240p	15.49kHz, 59.1Hz	

Catacomb	20180313	240p	15.49kHz, 59.1Hz	
Colony7	20181218	240p	15.49kHz, 59.1Hz	
ComputerSpace	20180313	240p	15.49kHz, 59.1Hz	menu not available over VGA
CosmicAvenger	20180313	240p	15.89kHz, 61.1Hz	
CrazyClimber	20180607	240p	15.49kHz, 59.1Hz	
CrazyKong	20180313	240p	15.49kHz, 59.1Hz	
CrushRoller	20180313	240p	15.49kHz, 59.1Hz	
Defender	20180313	240p	15.49kHz, 59.1Hz	
DonkeyKong	20180418	240p	15.86kHz, 60.6Hz	
Dorodon	20180313	240p	15.86kHz, 60.6Hz	
DreamShopper	20180313	240p	15.86kHz, 60.6Hz	
Eeekk	20180313	240p	15.86kHz, 60.6Hz	
Eyes	20180313	240p	15.86kHz, 60.6Hz	
Frogger	20180313	240p	15.86kHz, 60.6Hz	
Galaga	20180313	240p	15.86kHz, 60.6Hz	
Galaxian	20180313	240p	15.86kHz, 60.6Hz	
GalaxyWars	20171115	VGA	37.8kHz, 60.3Hz	
Gorkans	20180313	240p	15.86kHz, 60.6Hz	
Invaders	20181010	VGA	31.20kHz, 59.9Hz	
Joust	20181216	240p	15.49kHz, 60.0Hz	
LadyBug	20180313	240p	15.49kHz, 60.0Hz	
LizardWizard	20180313	240p	15.49kHz, 60.0Hz	
LodeRunner	20171115	VGA	37.8kHz, 60.3Hz	
LunarRescue	20171115	VGA	37.8kHz, 60.3Hz	
Mayday	20181218	240p	15.49kHz, 60.0Hz	
MoonCresta	20180313	240p	15.49kHz, 60.0Hz	
MoonPatrol	20180315	240p	15.51kHz, 50.0Hz	
MrDoNightmare	20180313	240p	15.49kHz, 59.1Hz	
MrTNT	20180313	240p	15.49kHz, 59.1Hz	
MsPacman	20180313	240p	15.49kHz, 59.1Hz	
Omega	20180313	240p	15.49kHz, 59.1Hz	
Orbitron	20180313	240p	15.49kHz, 59.1Hz	
Pacman	20180313	240p	15.49kHz, 59.1Hz	
PacmanClub	20180313	240p	15.49kHz, 59.1Hz	
PacmanPlus	20180313	240p	15.49kHz, 59.1Hz	
PacmanicMiner	20180313	240p	15.49kHz, 59.1Hz	
Pengo	20180313	240p	15.49kHz, 59.1Hz	

Phoenix	20180313	240p	15.49kHz, 59.1Hz	
Pisces	20180313	240p	15.49kHz, 59.1Hz	
Ponpoko	20180313	240p	15.49kHz, 59.1Hz	
Pooyan	20180313	240p	15.49kHz, 59.1Hz	
Robotron	20181215	240p	15.49kHz, 59.1Hz	
Scramble	20180929	240p	15.86kHz, 60.6Hz	
Sinistar	20181218	240p	15.86kHz, 60.6Hz	
SnapJack	20180315	240p	15.86kHz, 60.6Hz	
Space Attack II	20171115	VGA	37.8kHz, 60.3Hz	
Space Invaders	20171115	VGA	37.8kHz, 60.3Hz	
Space Invaders Part II	20171115	VGA	37.8kHz, 60.3Hz	
Space Laser	20171115	VGA	37.8kHz, 60.3Hz	
Splat	20181217	240p	15.86kHz, 60.6Hz	
Stargate	20181216	240p	15.86kHz, 60.6Hz	
Super Earth Invasion	20171115	VGA	37.8kHz, 60.3Hz	
SuperGlob	20180313	240p	15.86kHz, 60.6Hz	
TheEnd	20180313	240p	15.86kHz, 60.6Hz	
TimePilot	20180313	240p	15.86kHz, 60.6Hz	
VanVanCar	20180313	240p	15.86kHz, 60.6Hz	
WarOfTheBugs	20180103	240p	15.86kHz, 60.6Hz	
Woodpecker	20180313	240p	15.86kHz, 60.6Hz	
Xevious	20180313	240p	15.86kHz, 60.6Hz	
ZigZag	20181219	240p	15.86kHz, 60.6Hz	

TO-DO:
Declare specific resolution
Specify if PAL/NTSC alternate is available
Note is core expecting horizontal or vertical orientation

RTC board



Some cores (ao486, Minimig) use clock, so MiSTer provides the real time for such cores. MiSTer can take the time from internet if active connection is present.

This board is pretty simple and provide real time offline. It supports 2 types of battery holders and several types of RTC ICs. It's plugged into LTC connector.

Assembling notes

- If through hole battery holder is used, you need to cut all its plastic pins and make sure the holder is fully seated on board without gaps. Otherwise it may touch the I/O board and make short circuit.
- For through hole holder, make sure you cut the pins as short as possible so it won't protrude under the board which can make short circuit.
- The board has the hole on opposite side of connector. Use a plastic(or other non-conductive) screw to make a leg and prevent the board bend and touch the main board.
- SMD holder (left picture) is preferable as it's more slim.

Usage notes

To get real time saved, simply connect MiSTer to internet and let it run for around 15 minutes. or from the console:

By default the time zone is UTC(GMT). If you want to get the time of your zone, you need to do following:

- connect to MiSTer by FTP/SFTP
- navigate to /usr/share/zoneinfo/posix folder and find there the name of your place or time zone.
- copy that file to your computer under name timezone
- copy it back to MiSTer to folder /media/fat/linux with name timezone

From the console you can also force the hardware clock to update and read the time from it.

- *store datetime to RTC* : **hwclock -wu**
- *read RTC* : **hwclock -u**

Core support

If you have internet connection or RTC board installed, these cores will show correct time and date and/or be utilised by the core.

RTC is supported in following cores and how to test:

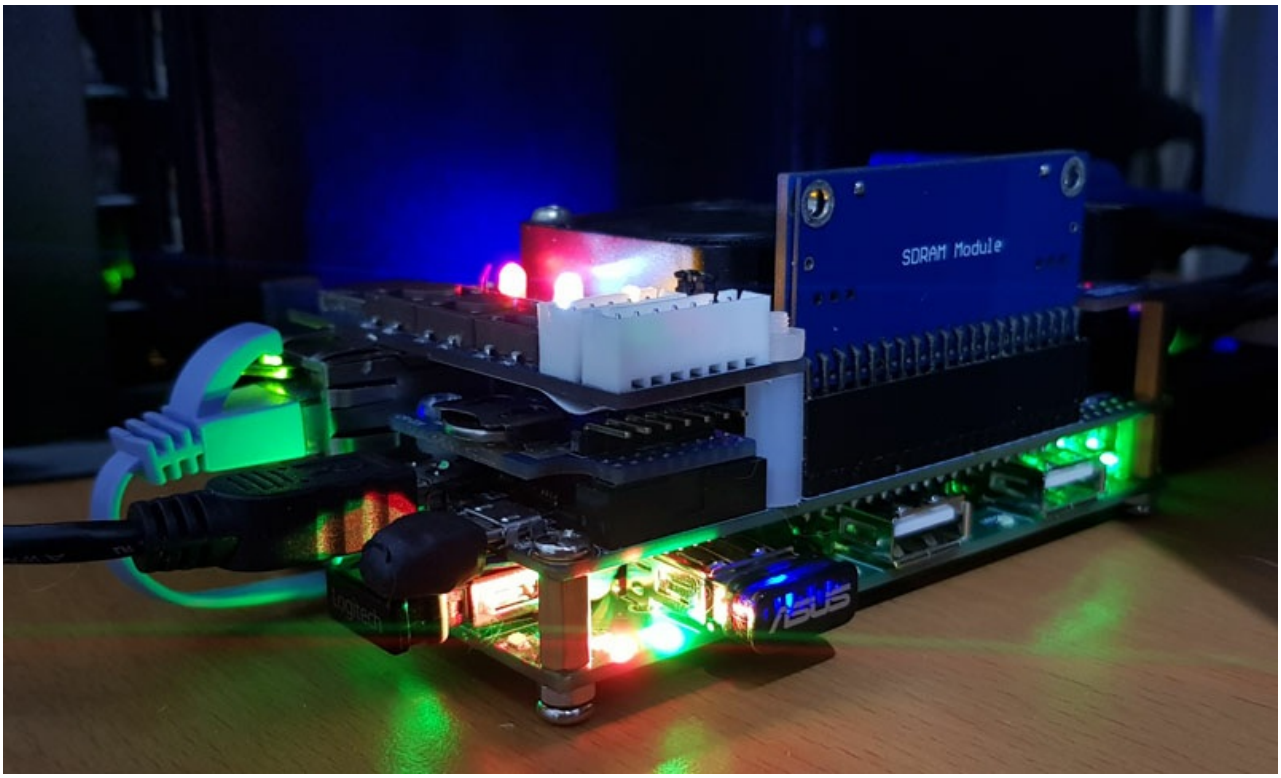
- ao486 (PC) (DATE on MS-DOS prompt)
- Archie (Acorn Archimedes) (Apps > Alarm)
- GBA (checked with Pokémon Sapphire, wall clock in first room)
- Gameboy (GB and GBC) (Pokémon Gold: set clock, go one room right, set day of week, save, restart; clock is in the menu screen)
- MSX ("MSX3") (DATE on MSX-DOS prompt)
- BBCMicro (BBC Micro Master 128K) (PRINT TIME$ in BBC Master mode)
- Menu (on top of the panels)
- Minimig (Amiga) (Utilities > Clock)
- NeoGeo ([DIP] Settings: ON > BOOK KEEPING > CABINET/COIN, check date of log entries)
- PC8801mk2SR (PC-88) (PRINT DATE$, shows year 85 but month and day are OK, and PRINT TIME$ shows correct time)
- QL (Sinclair QL) (PRINT DATE$)
- SNES (Daikaijuu Monogatari II, Tengai Makyou Zero - once set up, you need to reload the core for the clock to take the real time)
- TSConf (top of Wild Commander screen, also MaxiClock plugin)
- X68000 (DATE on HUMAN68K prompt)
- Zet98 (PC-98) (DATE on MS-DOS prompt)
- ZXNext (ZX Spectrum Next) (main menu in Next mode)

USB Hub

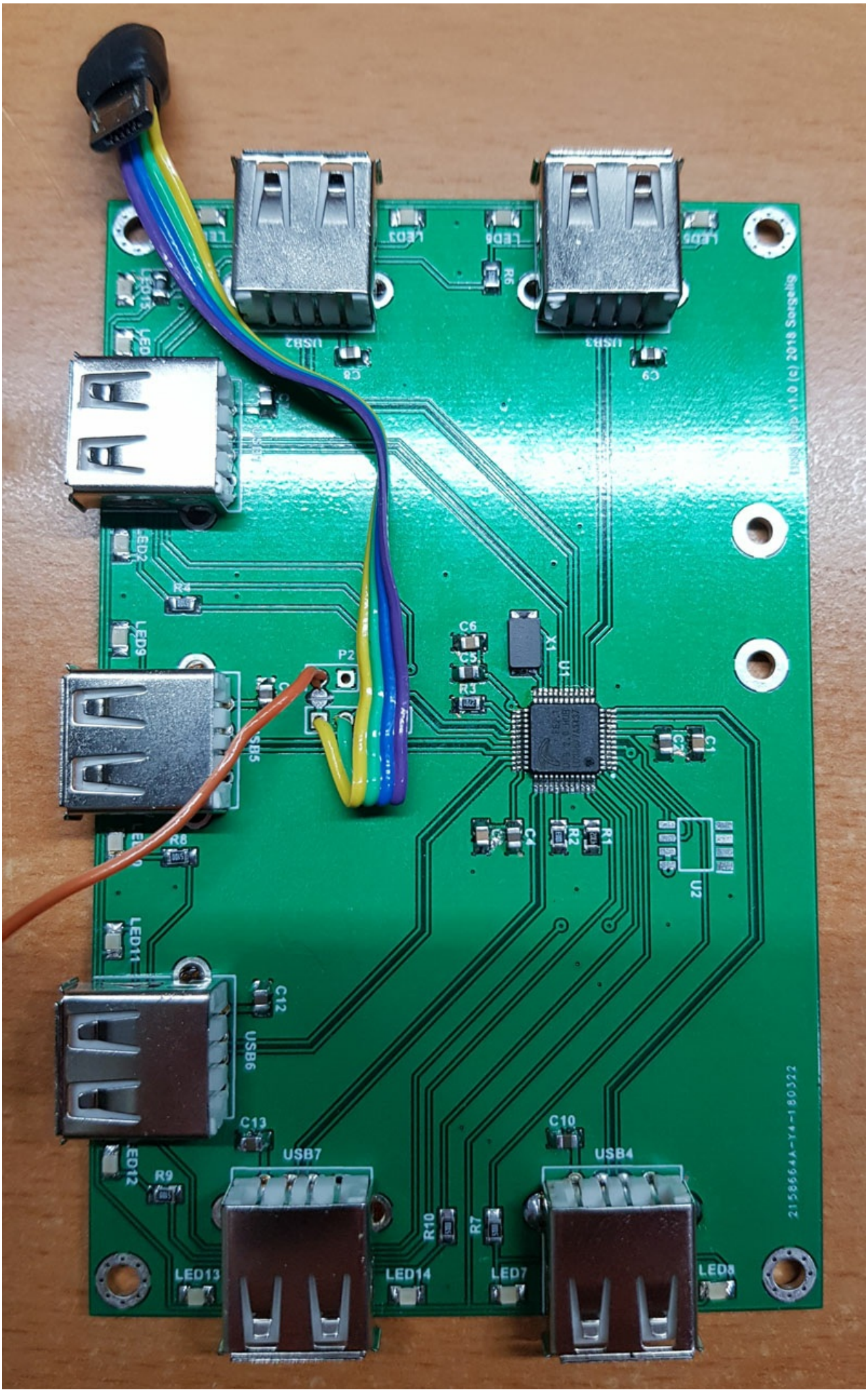
This is an optional add-on usb hub pcb made for convenient USB device connections. With this board the MiSTer has an extra seven USB ports that fit snugly under the existing chassis.

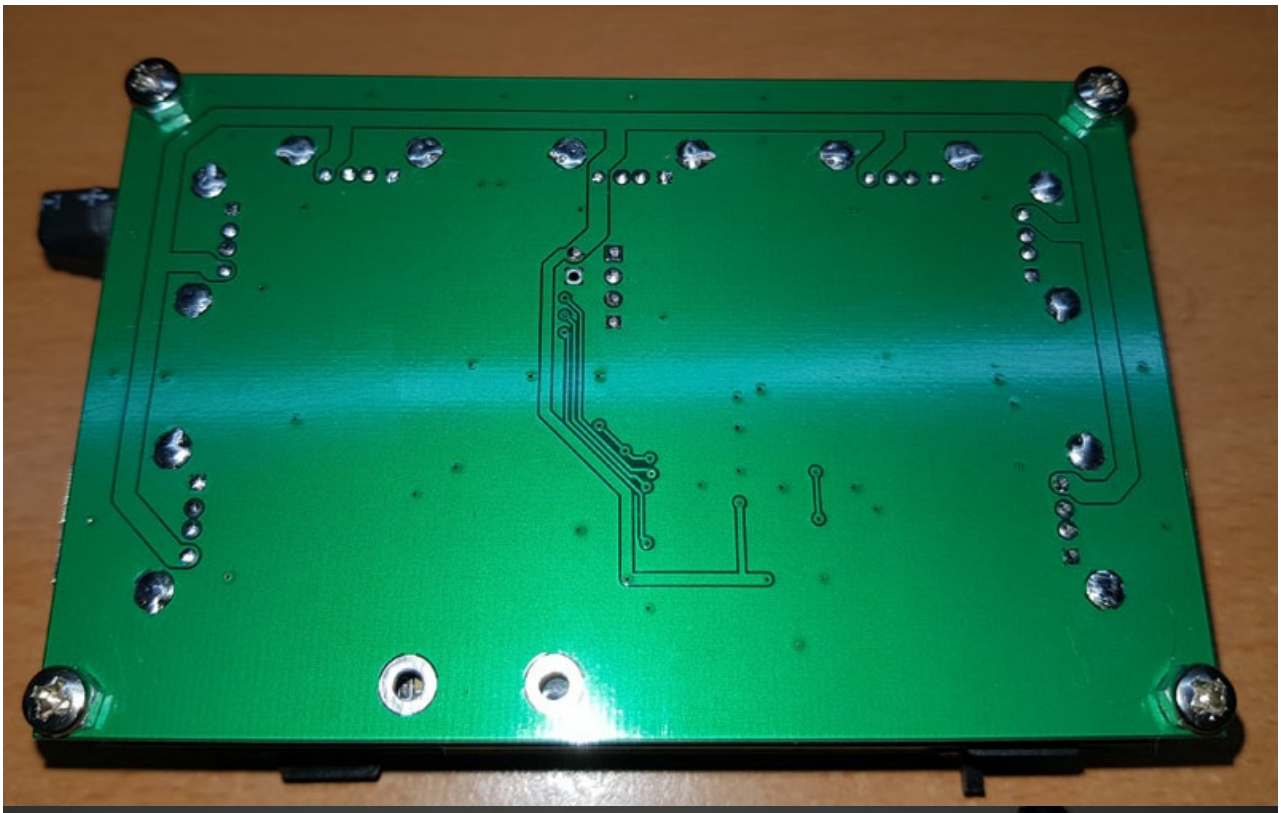
Order v1.2 PCB on PCBWay

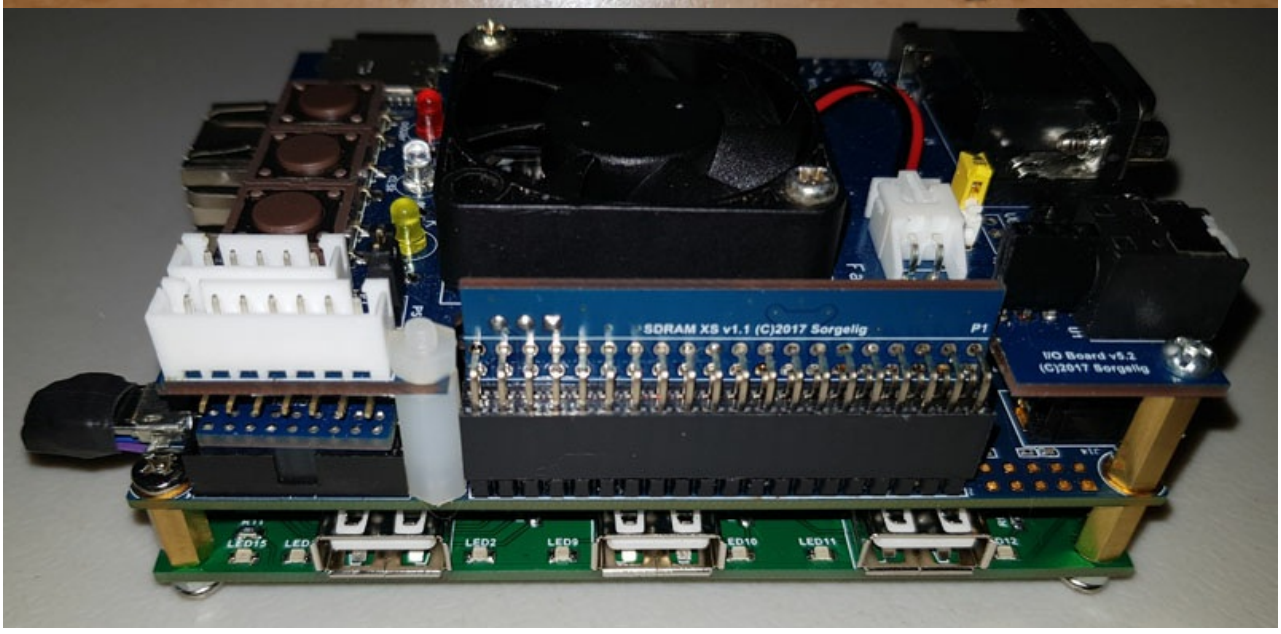
Order USB Bridge board PCB on PCBWay

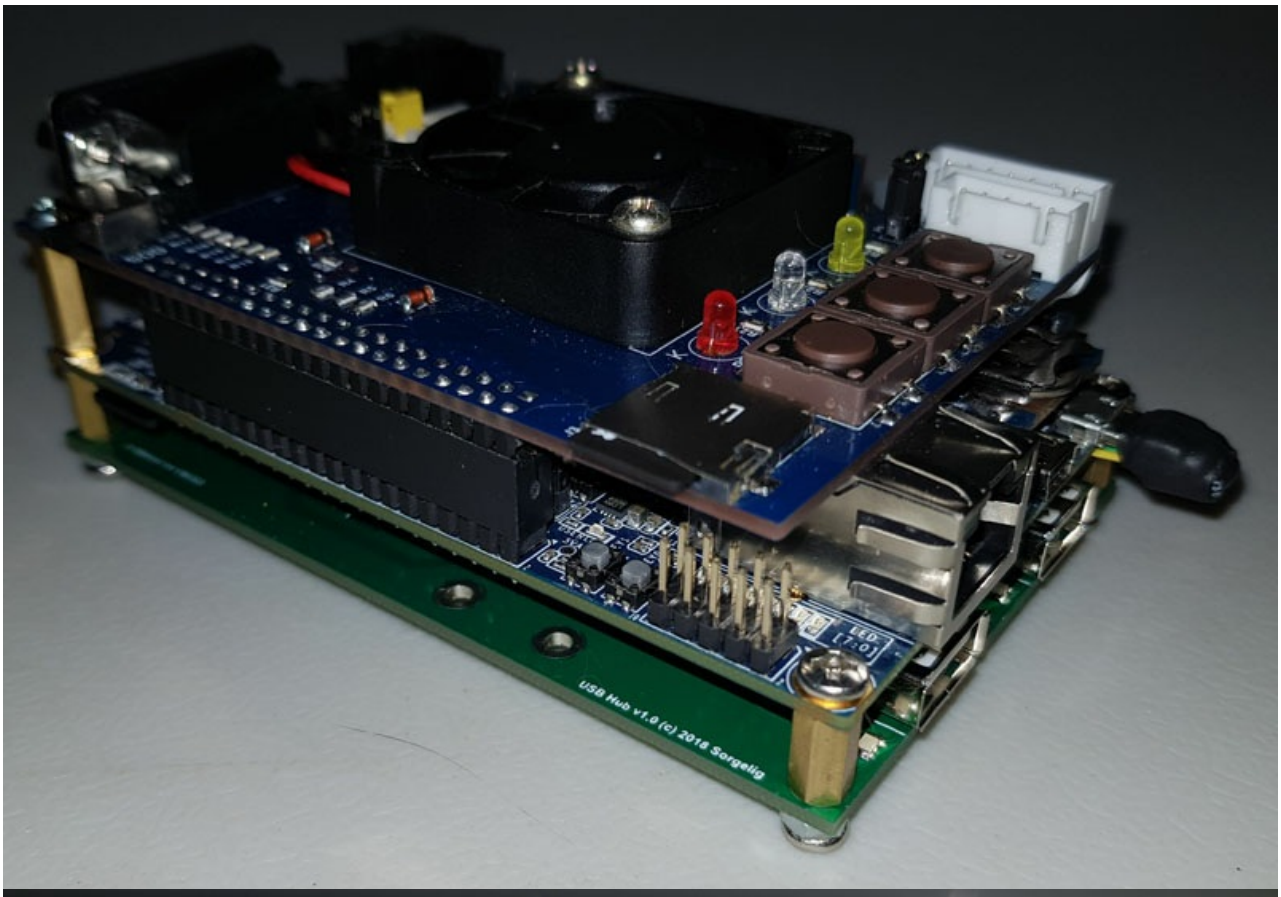


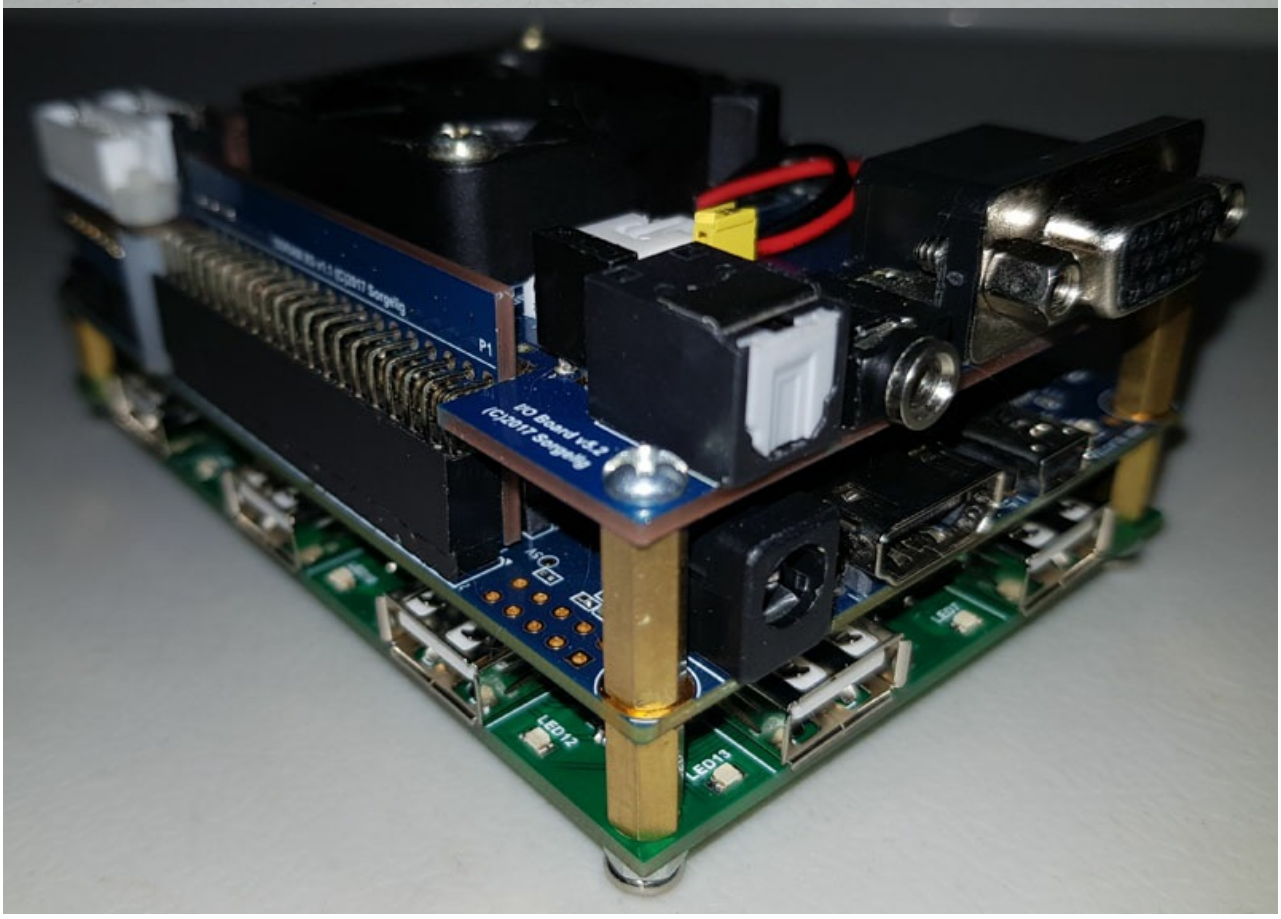
The Hub is based on the FE2.1 chip, which requires only a few external components. It uses 3 sides for USB sockets. Since the input Micro-USB connector has a flexible wired connection to the board, the user can mount it in either direction to choose the side providing the USB connections.


USB2
USB3
USB5
USB6
USB7
USB4
P2
X1
U1
U2
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C12
C13
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
LED2
LED3
LED5
LED6
LED7
LED8
LED9
LED11
LED12
LED13
LED14
2158664A-Y4-180322

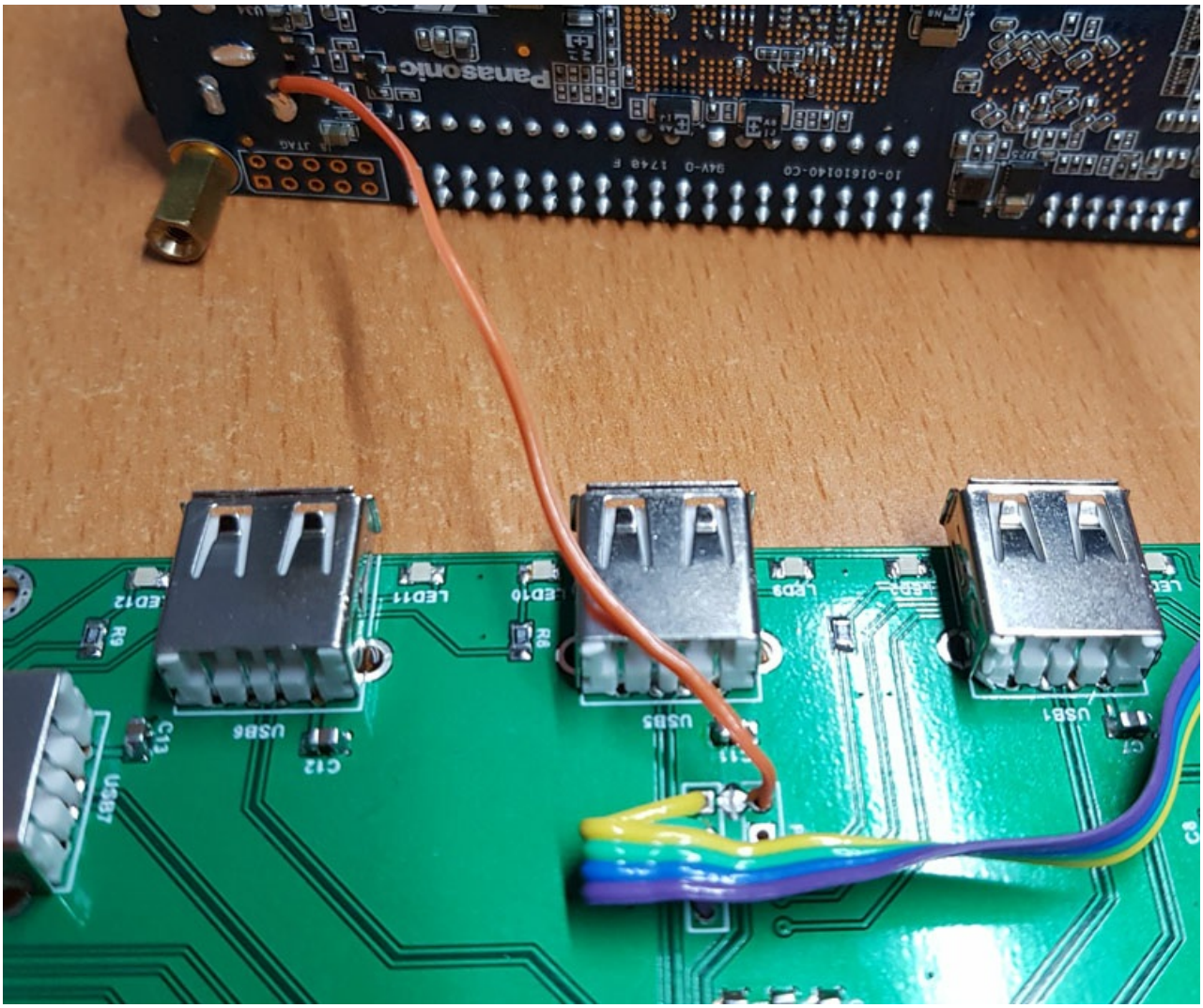


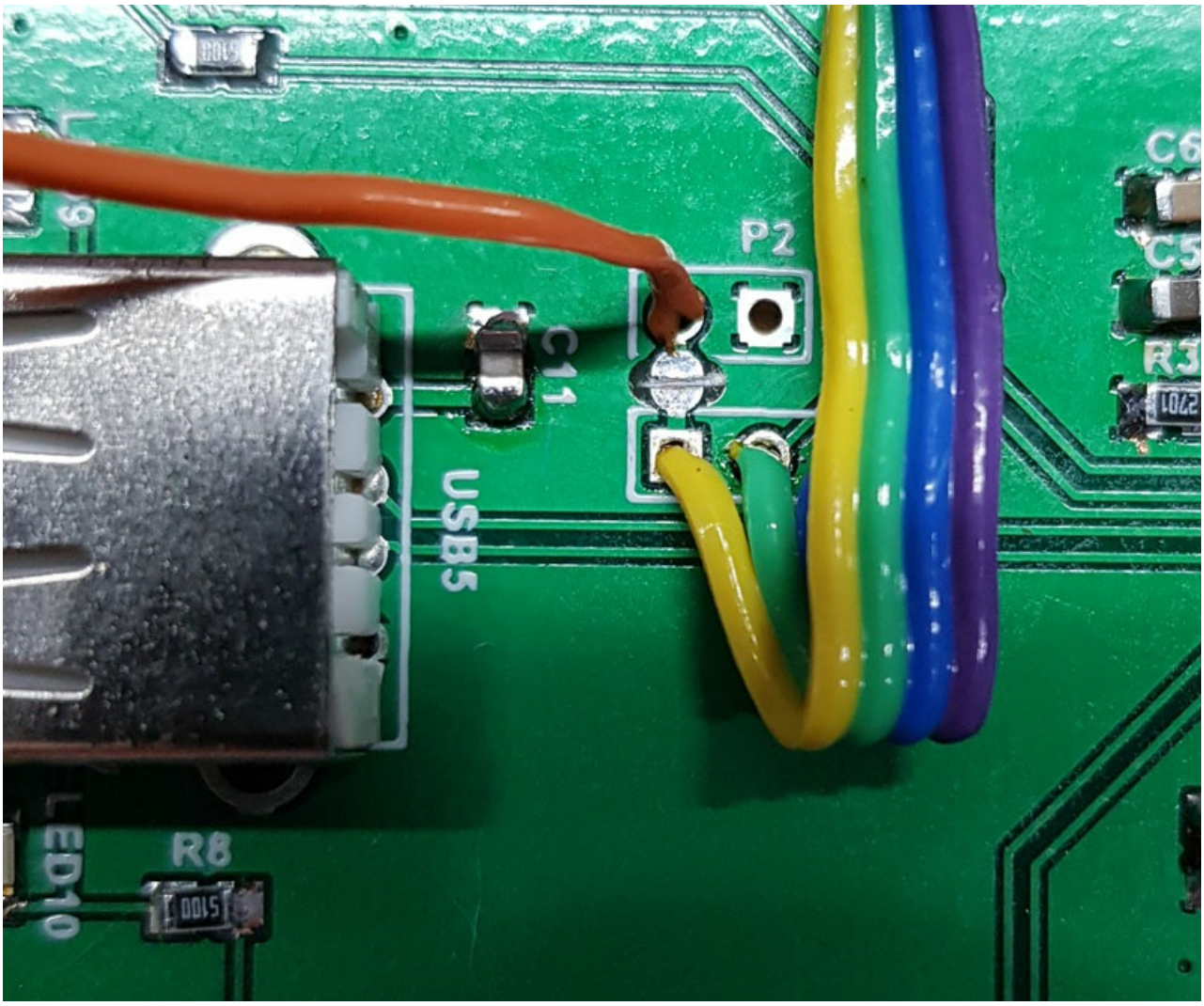

SDRAM XS v1.1 (C)2017 Sorgelig
P1
I/O Board v5.2
(C)2017 Sorgelig
LED15
LED2
LED9
LED10
LED11





The Hub add-on board allows you to use power from Micro-USB or external power. External power is preferred as this will prevent USB over-current triggering. In case external power is used, a small cut point (shown on picture) should be cut (or you can leave the Micro-USB +5V wire unsoldered)!


Panasonic
LED12
LED11
LED10
USB6
USB5
USB1
C13
C12


P2
C11
USB5
C6
C5
R3
R8
LED10

Core support

ADC is supported in following cores:

- Acorn Atom
- ADCTest
- AliceMC10
- Amstrad
- Apple-II
- Atari 7800 (used only to support the Atari 2600 Starpath Supercharger)
- Commodore 16
- Commodore 64
- Commodore VIC-20
- CoCo2 / Dragon 32
- Lynx 48 and 96
- Ondra SPO 186
- Oric
- SVI-328
- TI-99/4A
- Tesla PMD 85-1
- ZX-Spectrum
- ZX-Spectrum Next

The following are cores that are not officially released for MiSTer but do have ADC support:

- Acorn Electron
- MSX1
- Spectrum ZX48

ADCTest will help visualize and set volume levels and verify tape quality (and operation of the ADC itself).

3D-printed (DIY)

A custom case for your MiSTer can be 3D printed using these files from Thingiverse. There are separate models which you can use depending on the add-ons that you have.

Universal Case for the MiSTer to now support the I/O-Board 5.2 and all SDRAM configurations

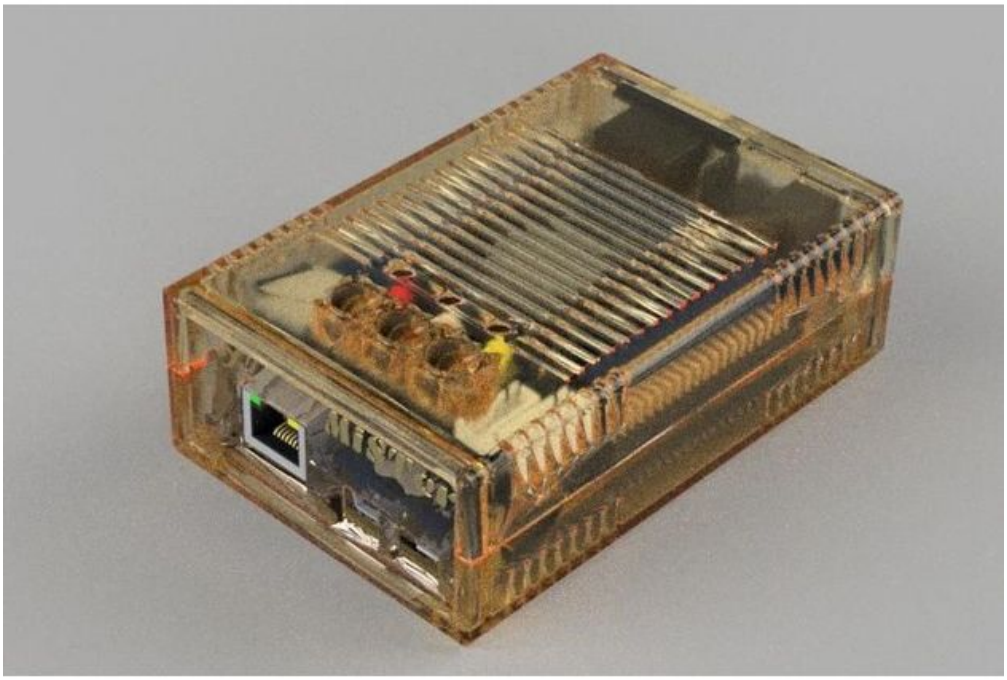
- MiSTer - Case Universal v5.2

MiSTer Sidecar USB case add-on usable with MiSTer USB Hub v1.1 daughter board Sidecar will with MiSTer - Case Universal v5.2 and v1.2

- MiSTer Sidecar USB case add-on

MiSTer XS Case v1 - v5.2 XS for SDRAM XS - extra slim - v1.1

- MiSTer XS Case v1 - v5.2 XS



MiSTer XS Case v2 - v5.2 XS for SDRAM XS - extra slim - v1.1 (the case is 4 mm higher than v1)

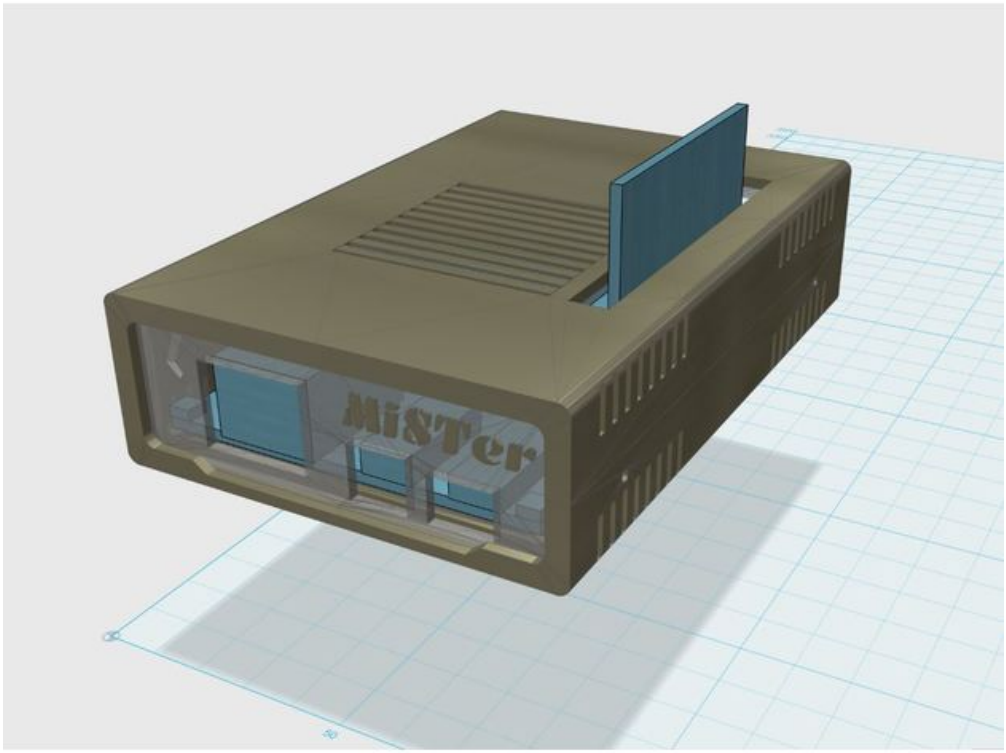
- MiSTer XS Case v2 - v5.2 XS

Case with fan holder which fits vertical and horizontal SDRAM boards (Case Universal)

- MiSTer - Case Universal v1.2

Slimmed down case that only fits vertical SDRAM board

- MiSTer - Case Slim v1.0



MiSTer - Case v1.0 *Updated*

Original case that only fits horizontal SDRAM board

- MiSTer - Case v1.0

MiSTer board

3D model (STEP 3D) for case development: MiSTer with I/O Board v5.2 and SDRAM XS v1.1

Pi-Top (v1)

This is an attempt to put MiSTer into Pi-Top v1 (original) case and make a MiSTer notebook.

Main note:

v1 case is discontinued by manufacturer. It just disappeared from the site. This case is still available from distributors such as RS-Online, Sparkfun, but won't last long. So, hurry up if you want to put MiSTer into this case! You may buy it even if you are not sure. You still can use this case for RPi/TB in case if you will change your mind.

This is not a budget case.

Case is pretty expensive but the only solution on the market. High price of the case doesn't mean good quality, alas. Plastic used on the case is so-so. Connections between plastic details aren't good either. Be careful as some parts are easy to brake (although pi-top site is full of pictures with children). Overall it looks like cheap case, but as has been said, there is no other option on the market to make your own notebook.

You can think about adaptation of some (old) notebook case. But it will be more hard and possibly even more expensive. Notebook displays usually have either LVDS or eDP interface while Pi-Top has HDMI. So you will need to find/make a conversion board for notebook display which is not always possible. Keyboard and touchpad in traditional notebook have "naked" interfaces providing low-level connection while Pi-Top provides USB connection for both keyboard and touchpad. So you will need to make another conversion board for keyboard and touchpad on traditional notebook. Battery management on traditional notebook is on main board, so another board is required while Pi-Top provides integrated HUB board to manage the battery and display power control. So, adaptation of traditional notebook will require pretty large board to convert internal interfaces to those compatible with MiSTer(DE10-nano). Different notebooks have different internals, so it's impossible to make a one board for all notebook cases. And usually even in old thick notebooks there is no much space to fit DE10-nano - especially in height dimension.

So, after judge all these difficulties, the only viable option is Pi-Top case.

There are 2 versions of Pi-Top cases.

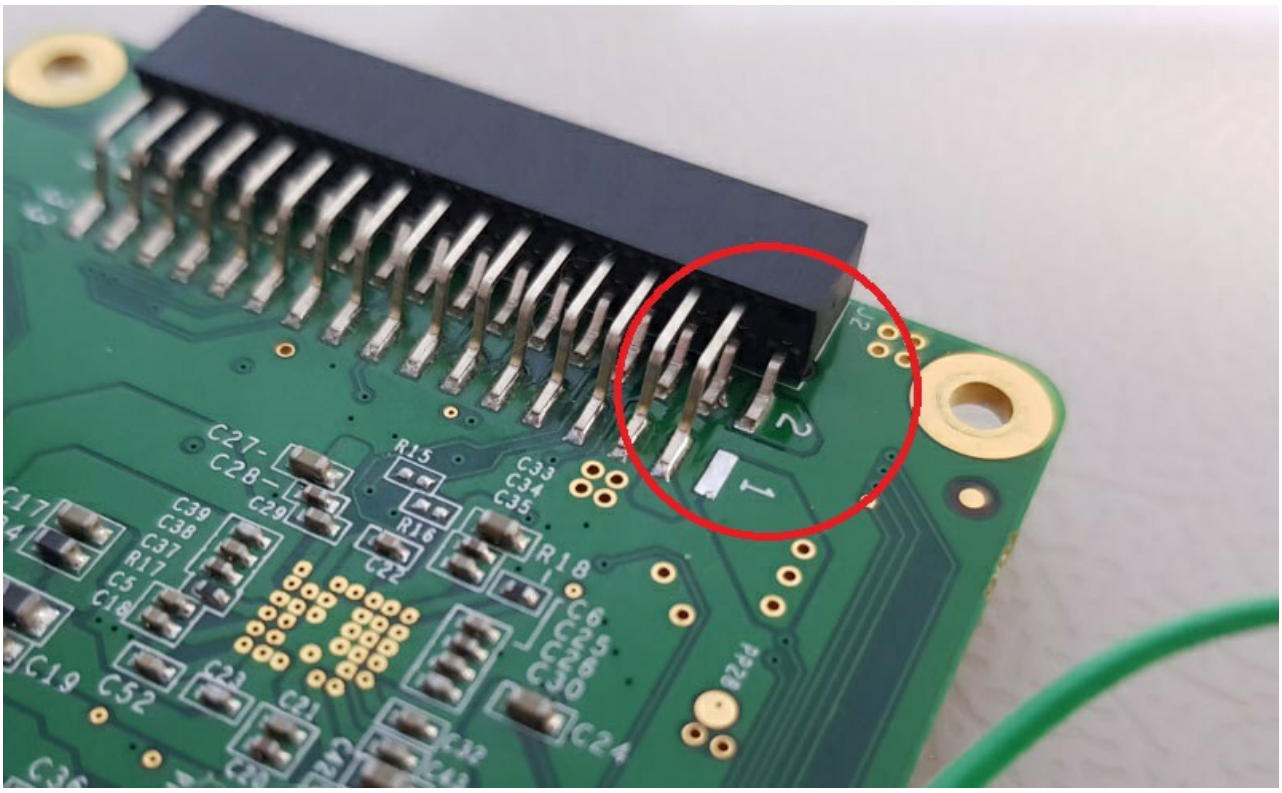
This project is targeted to v1 (called as original on Pi-Top site) case only. This version has more space inside and has no specific to RPi holes and places. So, we have pretty relaxed placement of DE10-nano inside. Semi-transparent cover gives good view to all DE10-nano LEDs and doesn't require to drill the holes. v1 uses railed holes to fix the boards, while v2 case uses magnet rails not suitable for this project.

Probably it's possible to use v2 case as well, but it will require completely different add-on boards.

v2 is available only in childish acid green color which is a main distraction point.

You need to be very careful while messing with internals.

- There is high voltage on HUB board and its connector. It's highly advised to de-solder and remove pin 1 from PiTopHUB connector.



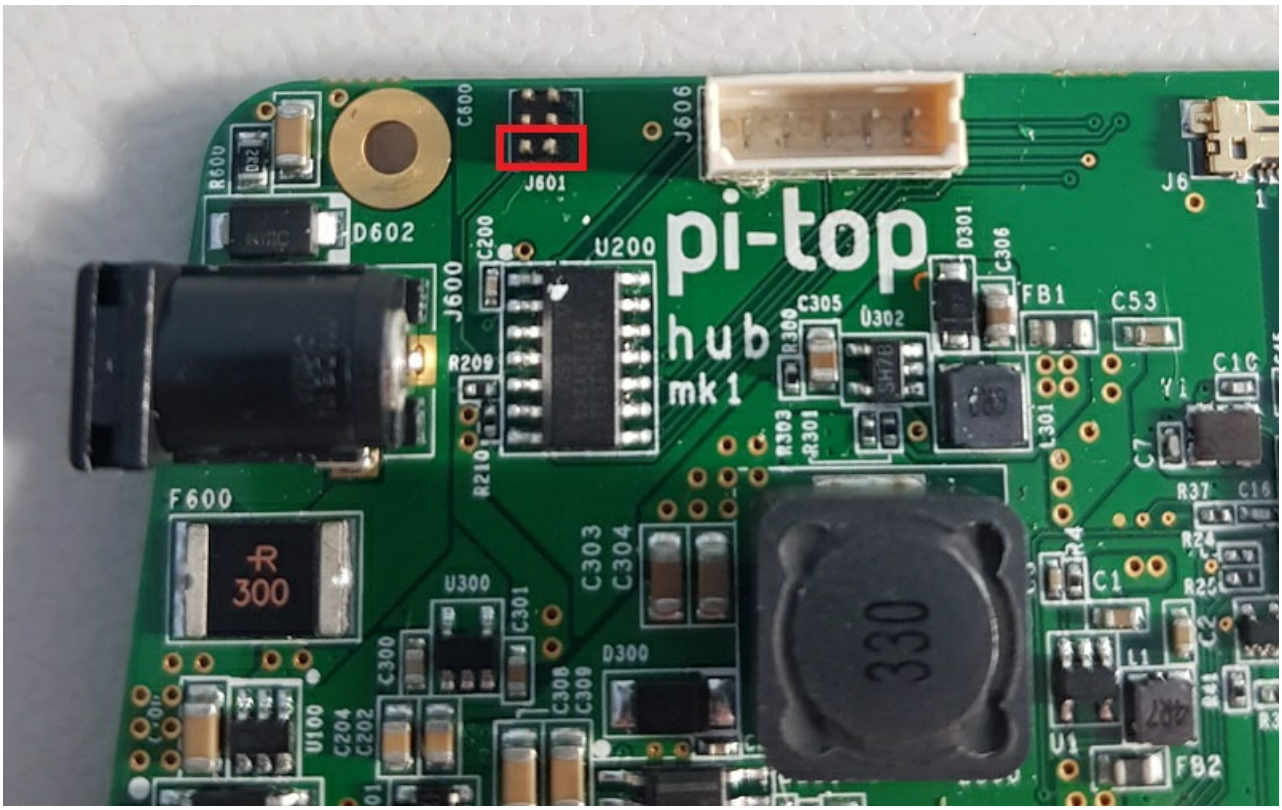
It delivers 16.5V and very dangerous for DE10-nano. There is no usage of 16.5V so it's better to remove this pin. Connector on PiTopHUB isn't shrouded and it's very easy to shift it while connecting and fry everything!

- Many parts on PiTopHUB are powered even when main power of Pi-Top is turned off. It's result of bad Pi-Top design. So, you need to be careful. Don't drop any metallic parts on the board. Avoid to touch the board.

Buggy MCU firmware.

While developing the boards for Pi-Top i found it's easy to make MCU on PiTopHUB to stick in non-responsible state. And since MCU is always powered (even in power off state) power cycle isn't enough. There are 2 ways to reset the MCU.

1. Remove PiTopHUB and effectively remove the power from MCU.
2. Carefully short the two lower pins on tiny 6-pin connector - this is reset of MCU.



New releases required.

New releases of Linux, MiSTer binary and Cores are required. Releases after March 02 2018 will be compatible with Pi-Top. Display of Pi-Top supports only one resolution - 1920x1080. So every core will require to support this resolution.

Butter-smooth video

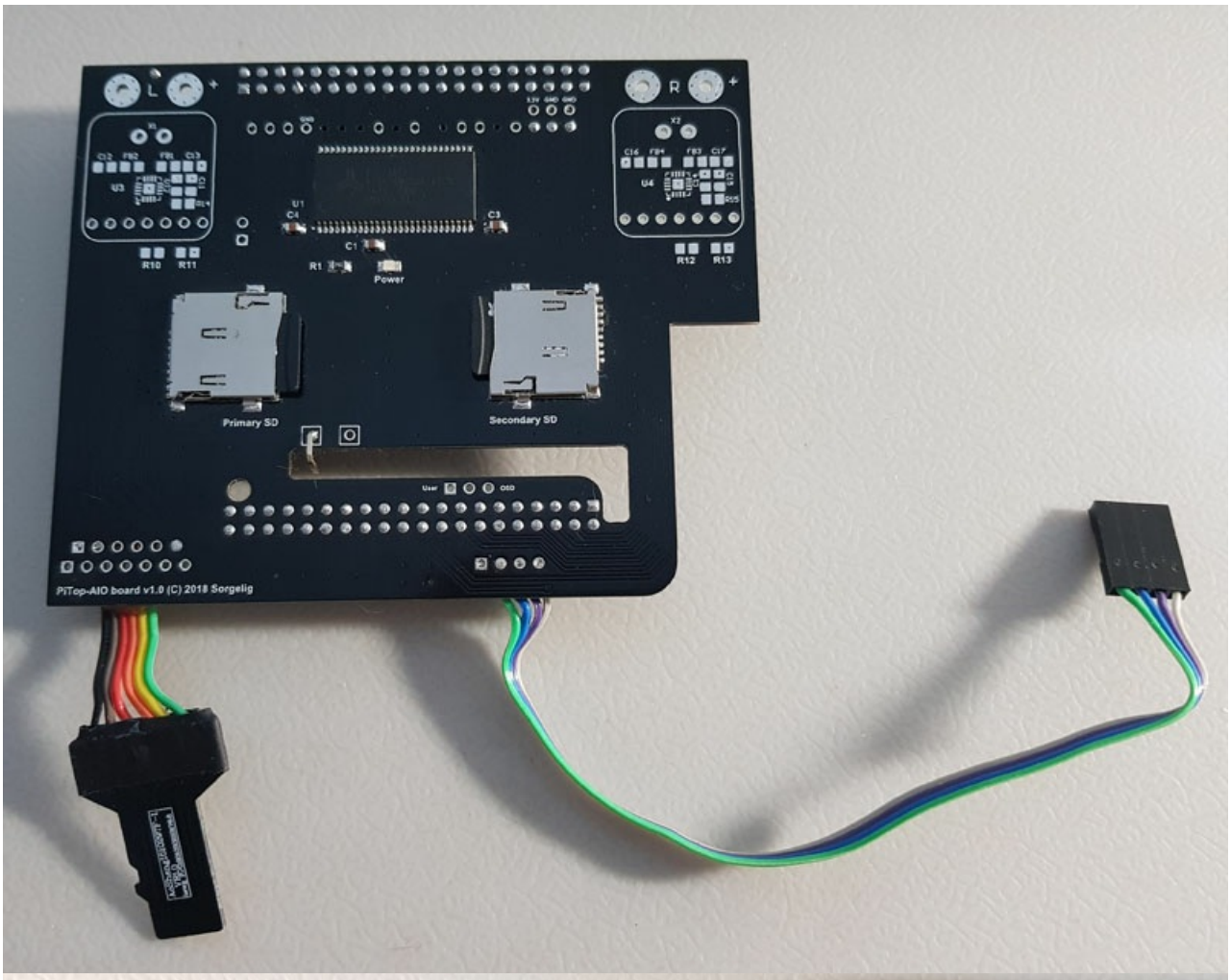
Although Pi-Top display supports only single resolution, it accepts any (reasonable) pixel clock and refresh rate. So, with this display and automatic VSync adjust option video may have original retro system refresh rate. All scrollers will be very smooth.

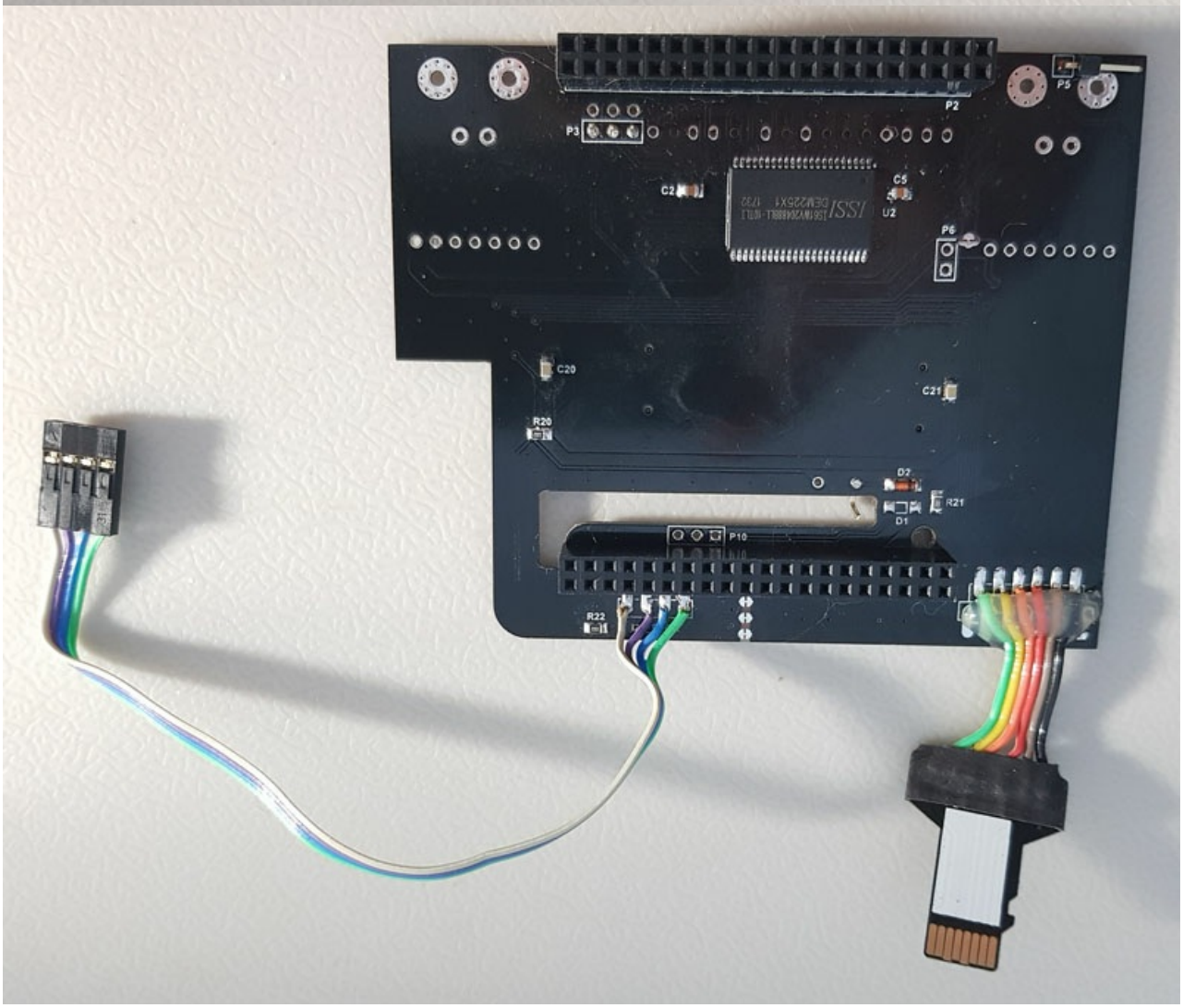
Required Boards

Gerber and PDF files for all boards can be found here

Main Board (Order on PCBWay)

This board has all important parts such as memory, both SD cards and Audio.


Primary SD
Secondary SD
Power
PiTop-AIO board v1.0 (C) 2018 Sorgelig

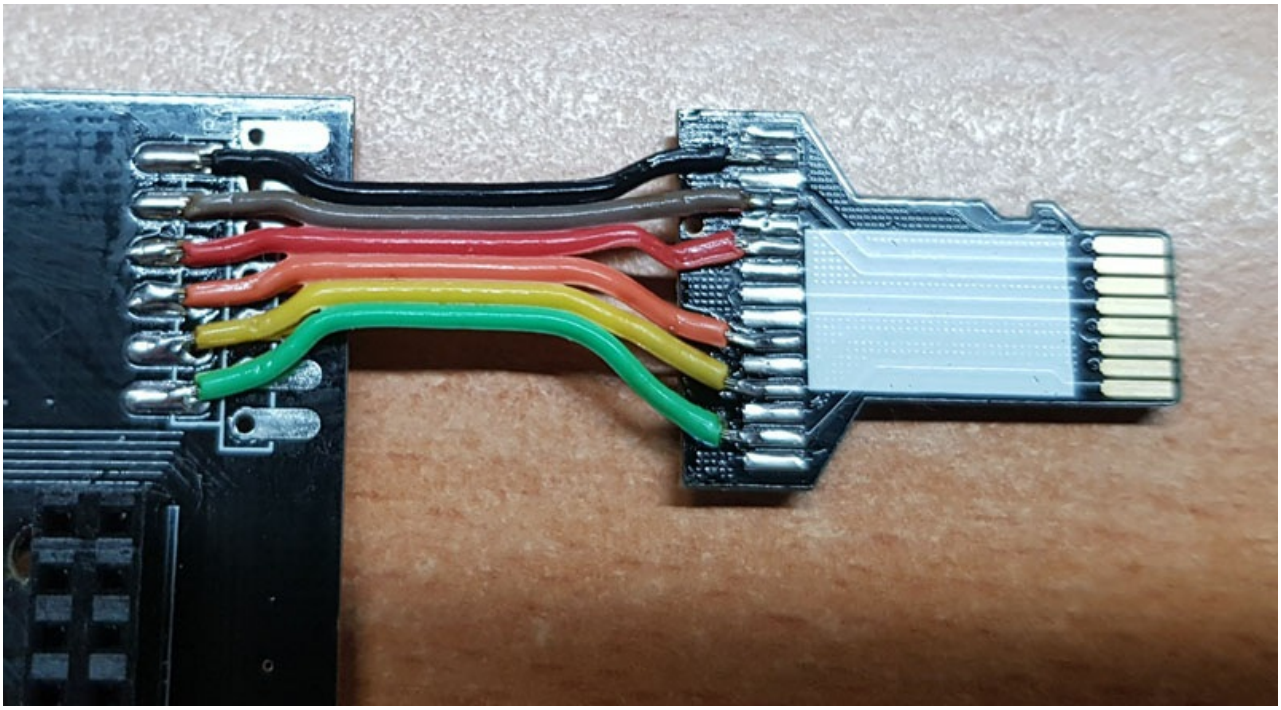

P2
P3
P5
P6
C2
C5
U2
C20
C21
R20
R21
R22
D1
D2
P10

Memory

Both SDRAM (32MB) and SRAM (2MB) are on this board, although SRAM is not supported by any core. There are no plans to use SRAM in foreseen future, so it's advised not to solder SRAM and save the cost.

SD Cards

Board has the same secondary SD card as on original MiSTer board. Another SD card is original DE10-nano card through micro-SD extender providing convenient access to card as original is in hard to reach place.



Audio

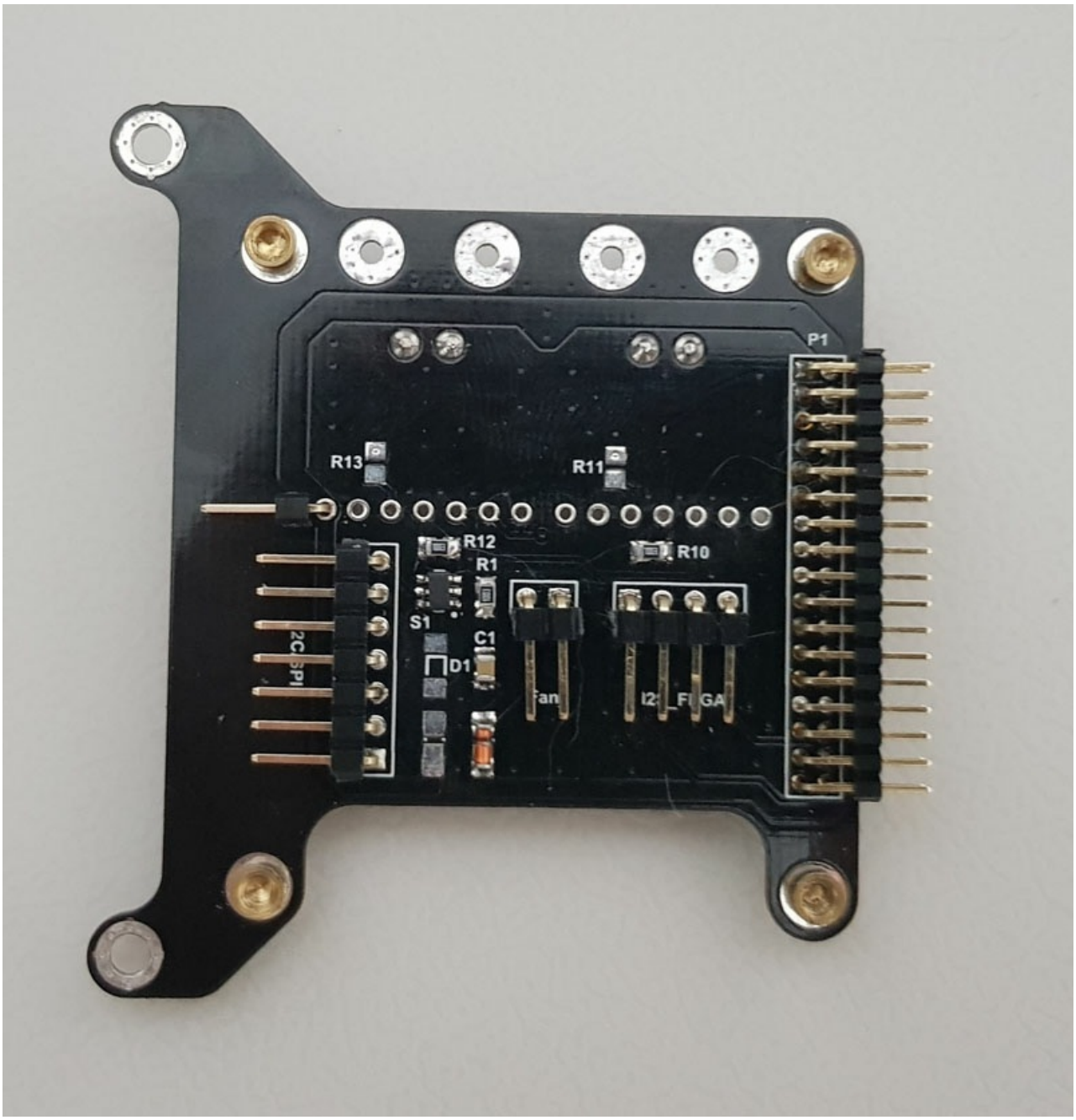
Audio amplifiers may be soldered directly on board by components, or using Adafruit audio breakout boards by soldering connectors points. MAX98357A has very tiny pads - if you are not sure in your soldering capability, use the breakout boards. The price is about the same as set of components. Although board has places for audio amplifier, it's advised not to solder it. Audio amplifiers gives up to 3.2W per channel and may draw up to 1.2A from 5V source. Due to audio nature, the draw will not be constant and will hammer the DE10-nano power circuit. So, it's advised to solder audio amplifiers on separate Audio board. By default 5V power of audio is connected to 5V of DE10-nano. There is an option to use external 5V supply for audio by cutting tiny cut point near P6 connector and connect external 5V (take from Audio board) source to unload internal DE10-nano power circuit.

Notes

Due to height limit in Pi-Top case, this board is mounted at some angle. P1 uses standard profile connector while P2 uses high profile connector (same as on standard MiSTer I/O Board). Thus one side of Arduino connector will slightly protrude from this board's cutout while other side will be covered by the board. You need to cut pins on both connectors at the board level before soldering to make sure it won't prevent closing of sliding cover. (pictures are coming) P4(primary card extender) and I2S_FPGA (if you use audio board) are better to be soldered as SMD using exposed pads on the bottom of the board for the same reason.(pictures are coming)

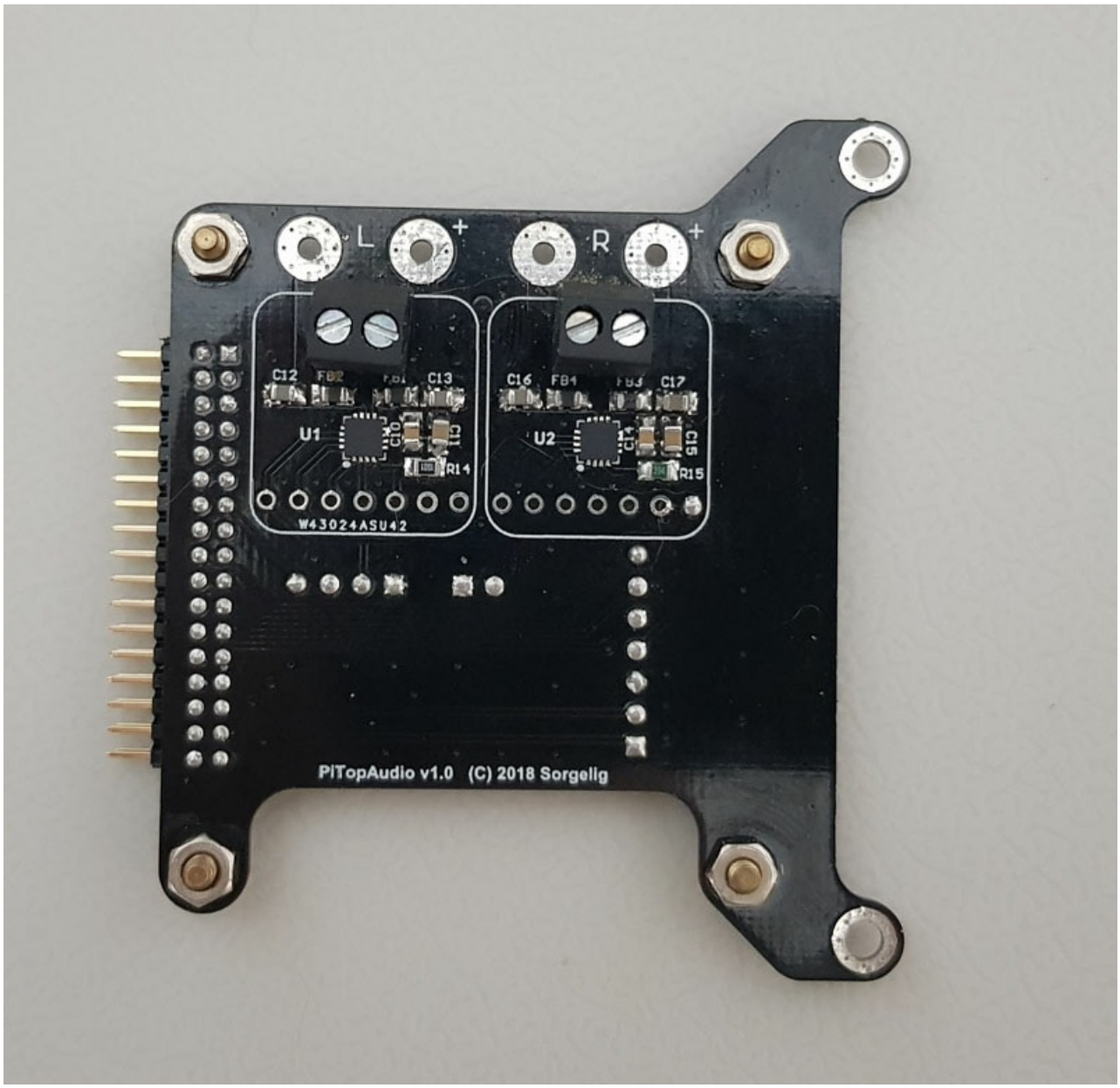
Recommended board thickness is **1.2mm**. This board thickness is included to total height of the whole construction.

Audio Board (Order on PCBWay)

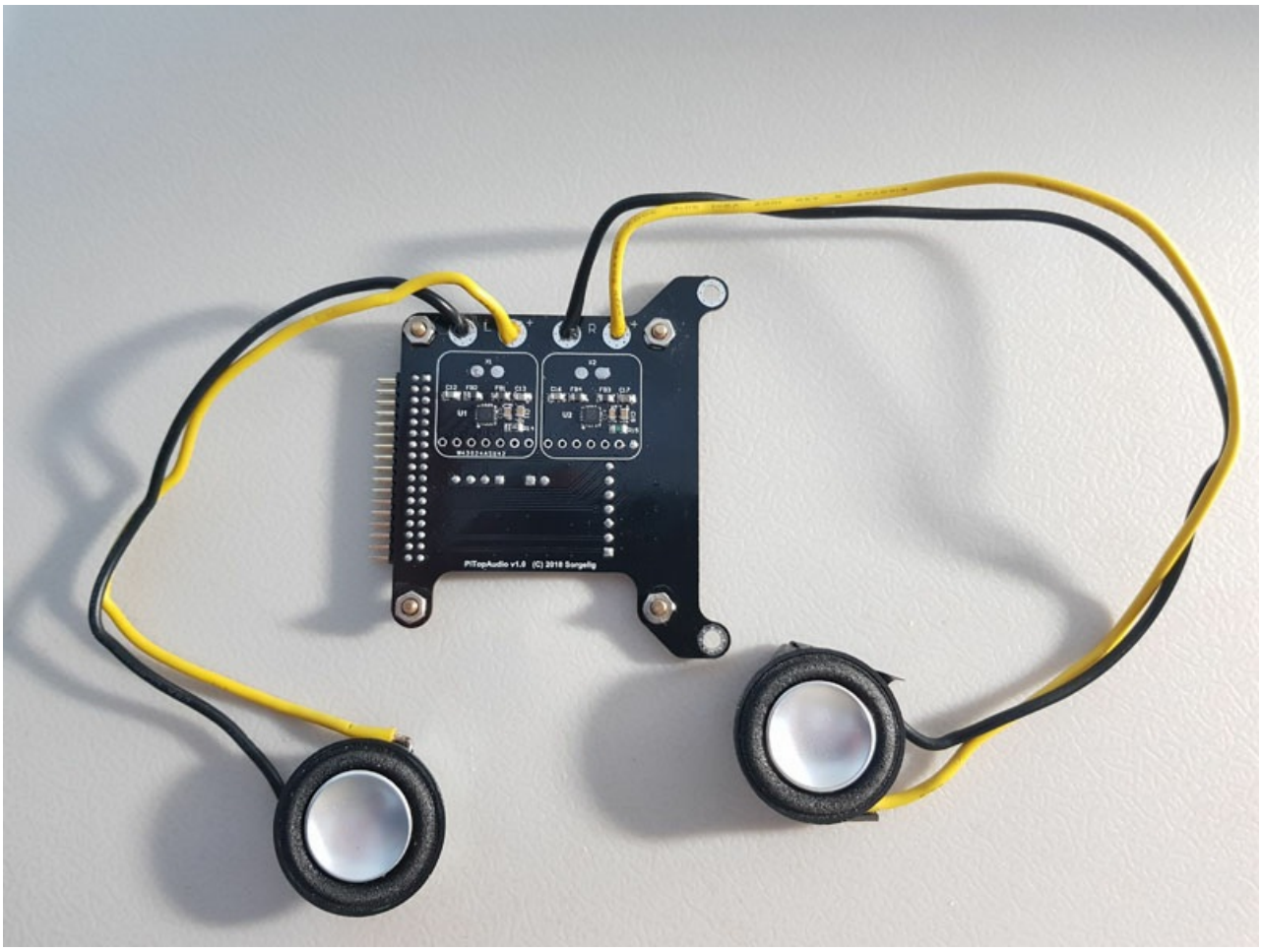


Using sockets

for speakers:

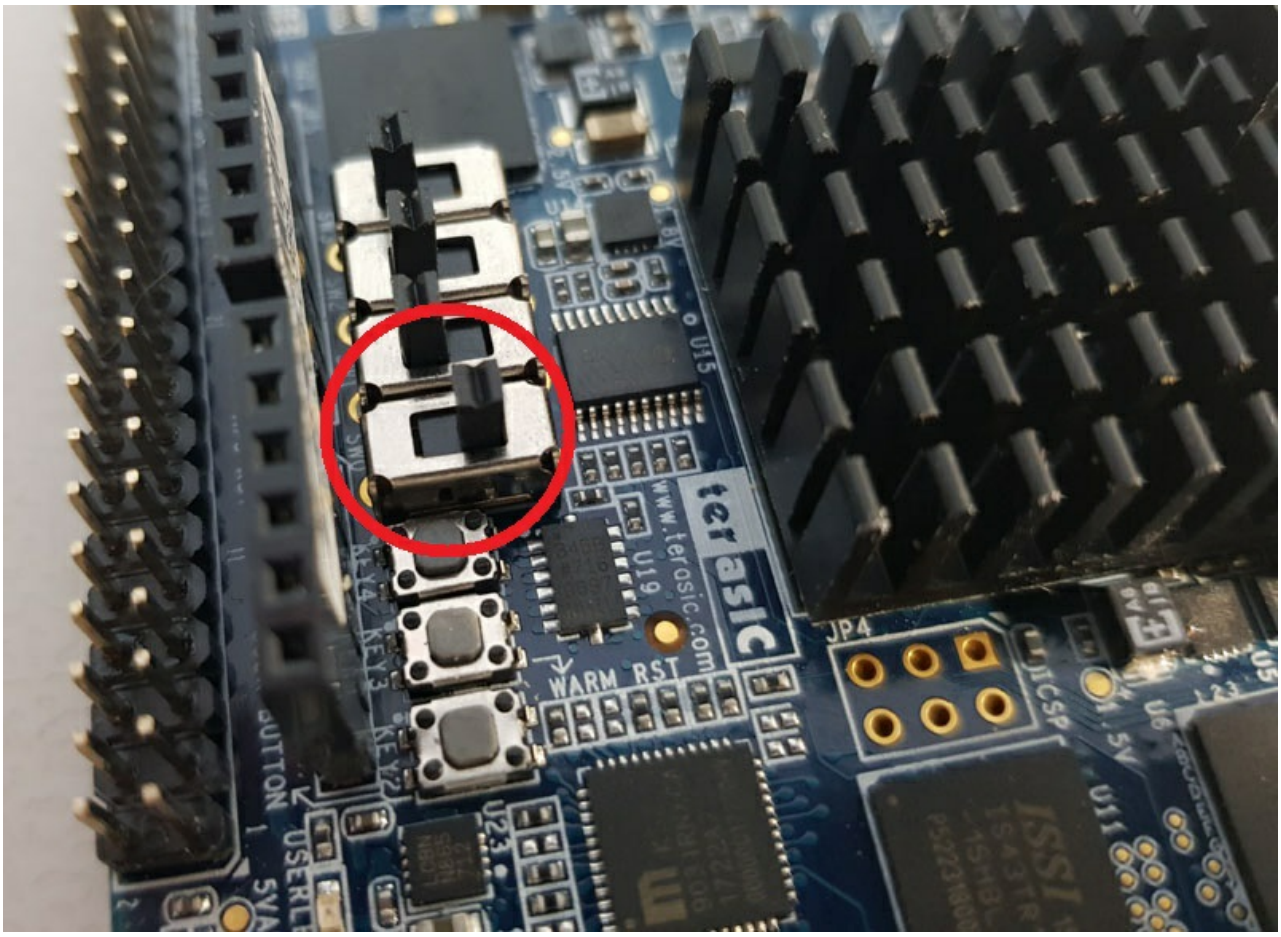


Directly wired:



Audio amplifiers can be soldered directly on board by components, or using Adafruit audio breakout boards by soldering connectors points. MAX98357A has very tiny pads - if you are not sure in your soldering capability, use the breakout boards. The price is about the same as set of components.

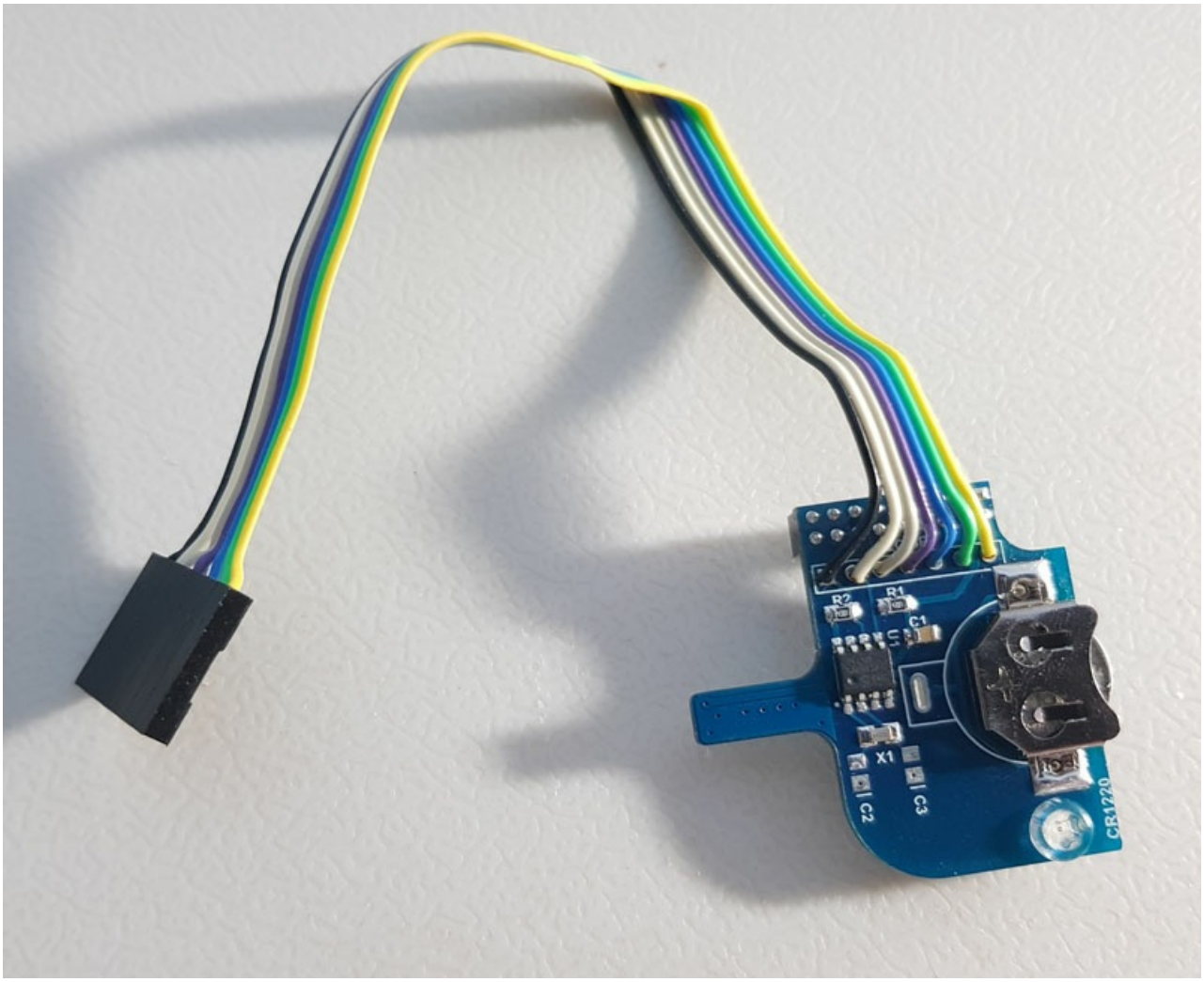
Set on-board DE10-nano switches according to picture to enable I2S output to audio board:



Besides audio amplifiers, the boards are providing I2C and SPI connections to DE10-nano (through RTC board). I2C is used for battery monitoring. SPI is used for display brightness control. It's also providing power for fan. Fan power circuit has flexible way to adjust the power using diodes and resistors (see schematics).

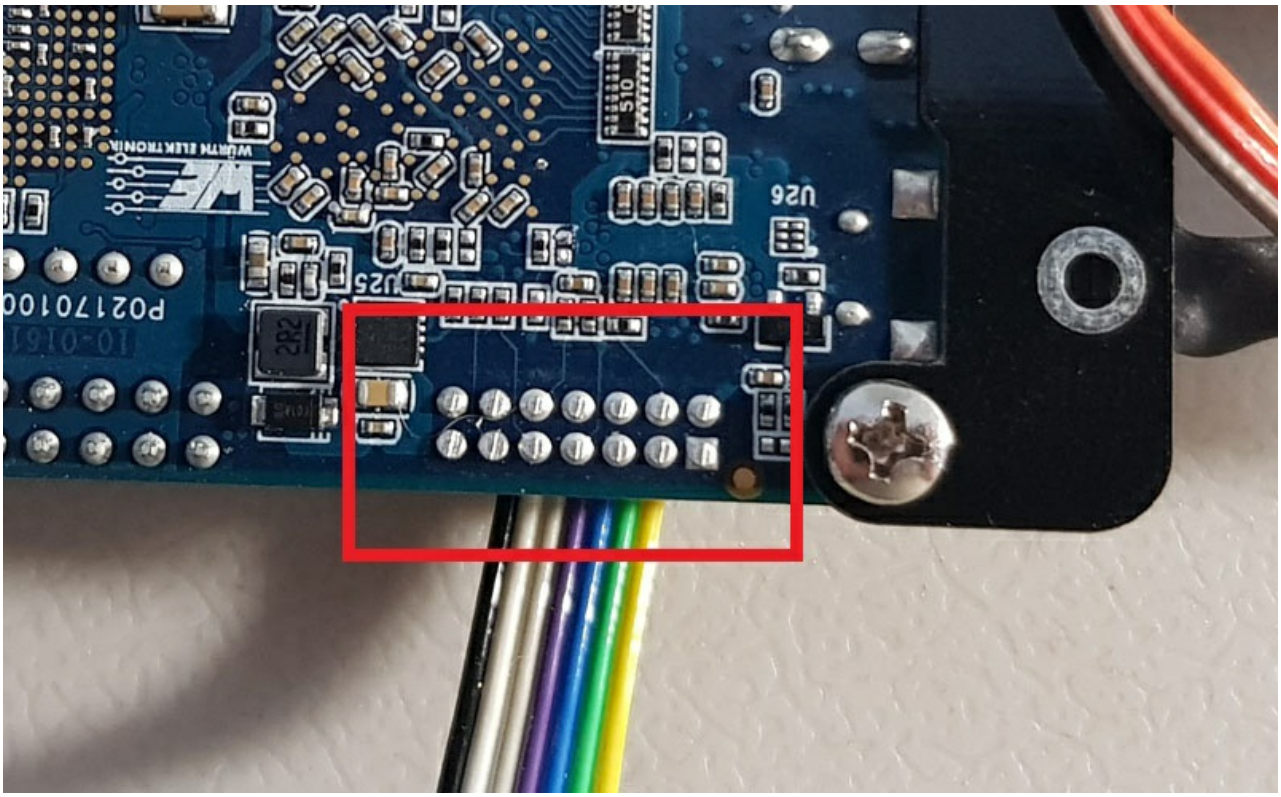
Recommended board thickness is **1.0mm**. This board thickness is included to total height of the whole construction.

RTC Board (Order on PCBWay)



RTC v1.3(or higher) is required to provide I2C and SPI connections. You don't need to populate the board if RTC is not required. You may connect I2C/SPI directly to DE10-nano without RTC board using 14-pin cable connector - just follow the RTC schematics.

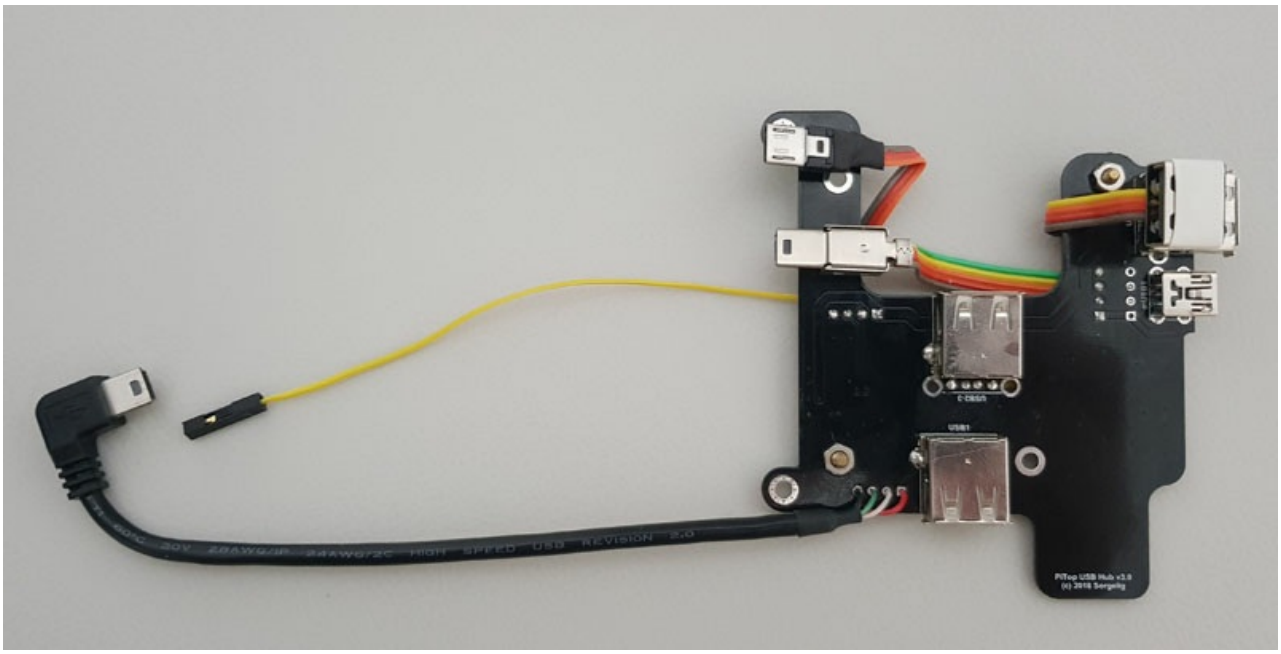
It's better to trim connector pins on the bottom of DE10-nano board to prevent the cable puncture:

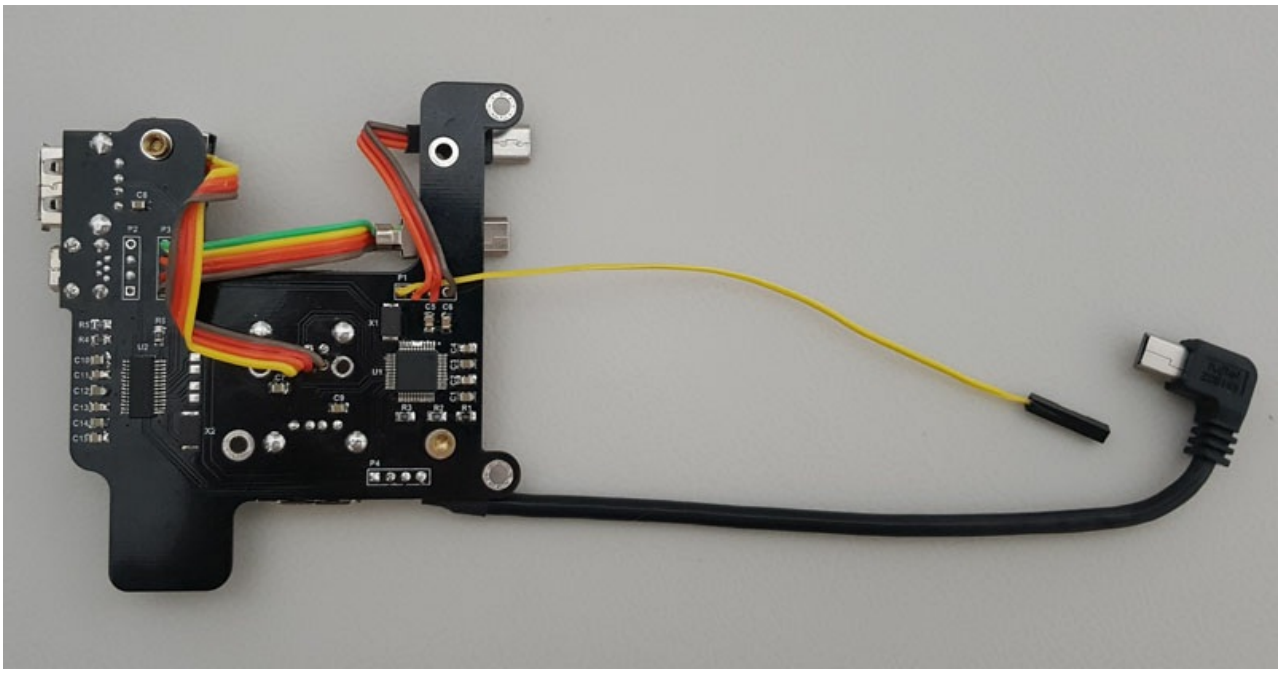


Recommended board thickness is **1.6mm**

USB HUB Board (Order v2 on PCBWay) (Order v3 on PCBWay)

You may solder dual USB2-3 socket directly to the board, or (as shown on pictures) solder single socket with other socked hanging on wires to fit some larger USB device (such as 8BitDo receiver) inside.





This board provides convenient USB device connections - both internal(3xUSB) and external(1xUSB). Board also provides mini-USB pass-through for console (or USB Blaster if required). This is optional board. You may use any tiny USB HUB to connect devices as you like.

There are 2 versions:

- v2 - based on 2 FE1.1S chips. This version is enough for most users. SSOP28 chips are relatively easy to align and solder.
- v3 - based on FE1.1 (periphery hub) and FE1.1S (Console+Blaster). FE1.1 is MTT USB hub providing high performance for mixed usage when both slow devices like keyboard/gamepad and high performance devices like WiFi or USB storage connected at the same time. FE1.1 uses LQFP48 package which is harder to align and solder.

Both versions have 2 Hubs on board. If you are not actively developing FPGA cores, then most likely you don't need to solder the hub on U2 chip. Only solder the hub based on U1 chip and then mUSB1 and the miniUSB "tail" to P2.

Recommended board thickness is **1.0mm**. This board thickness is included to total height of the whole construction.

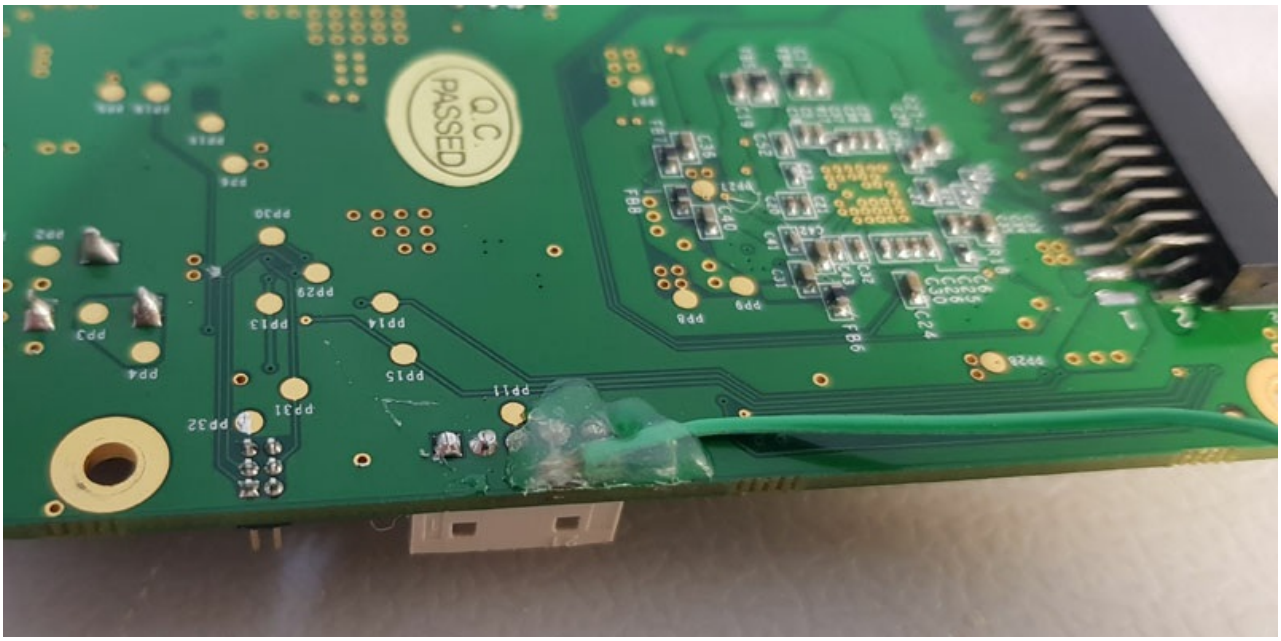
LEDs

Pi-Top v1 case has large semi-transparent door, so any lights will be visible through. Thus LEDs on DE10-nano are used for indications. Main Board provides the power LED.

Buttons

Use keyboard to simulate OSD(F12), User(LCtrl-LAlt-RAlt) and Reset(LShift-LCtrl-LAlt-RAlt) buttons.

It's possible to use hardware reset button by short press of power button but it requires hardware modifications of DE10-nano.

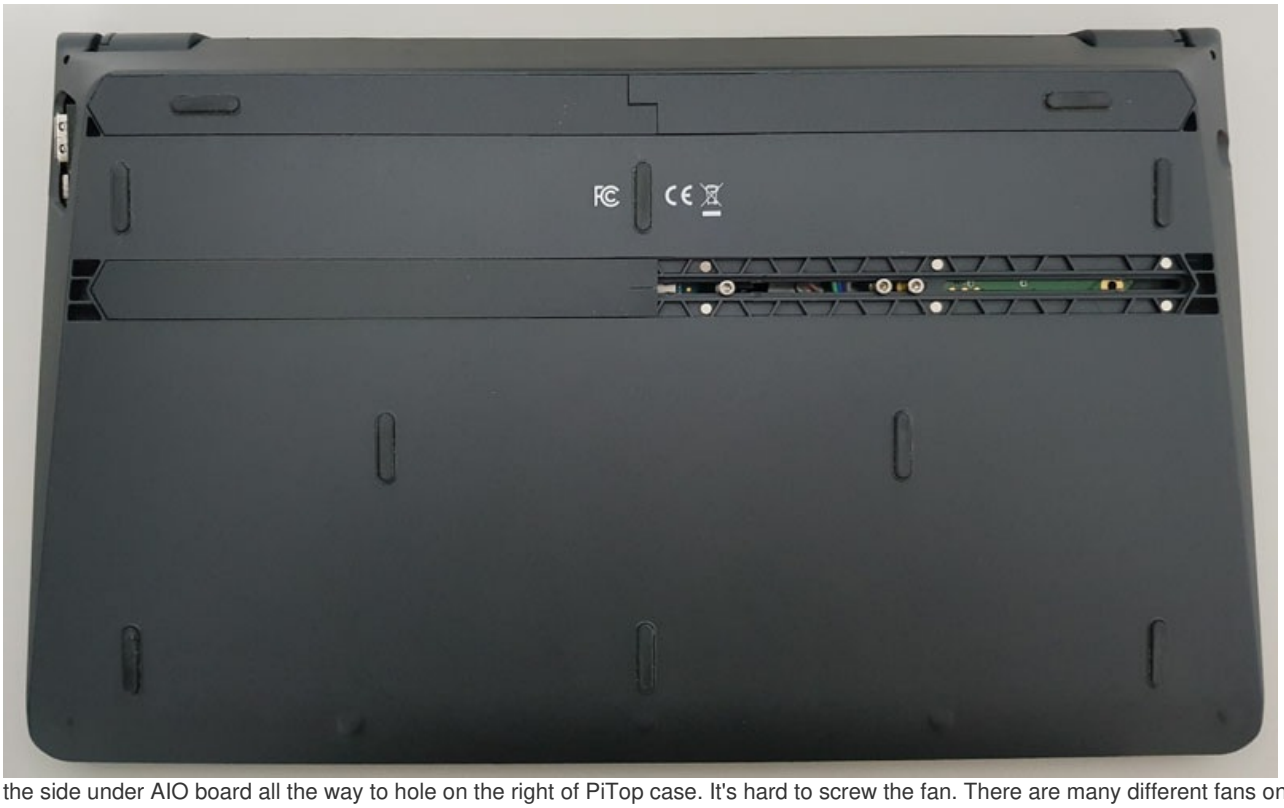


Power button.

Pi-Top power button requires about 2 seconds hold to turn the power on/off. If brightness has been changed since power on, then power button requires around 3-4 seconds hold to turn the power off.

Cooling

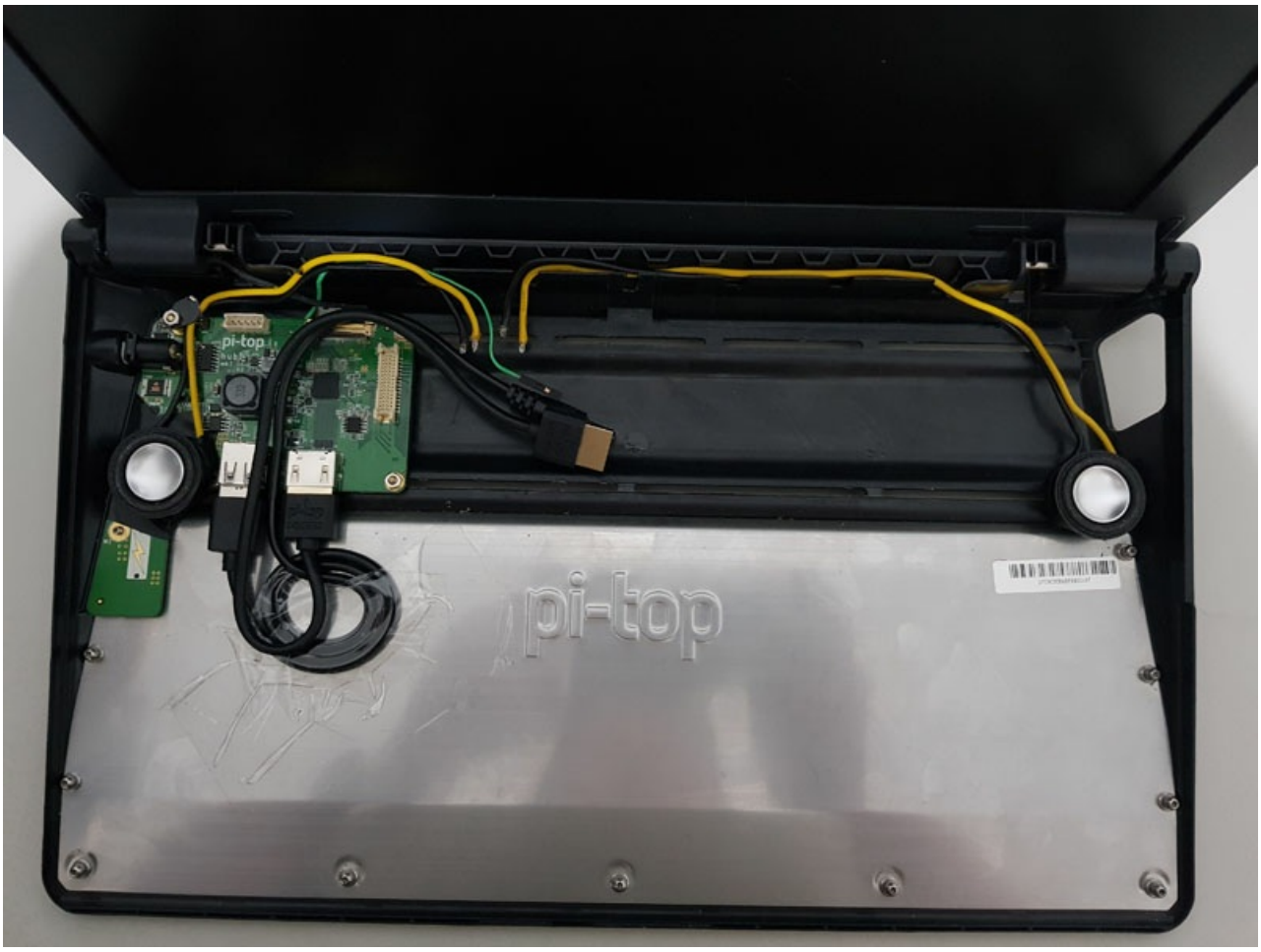
Cooling is necessary as the space around the main chip is to small for passive cooling! The construction uses the turbine fan. It takes air from bottom of the case (remove one magnetic plate on the bottom for air flow)



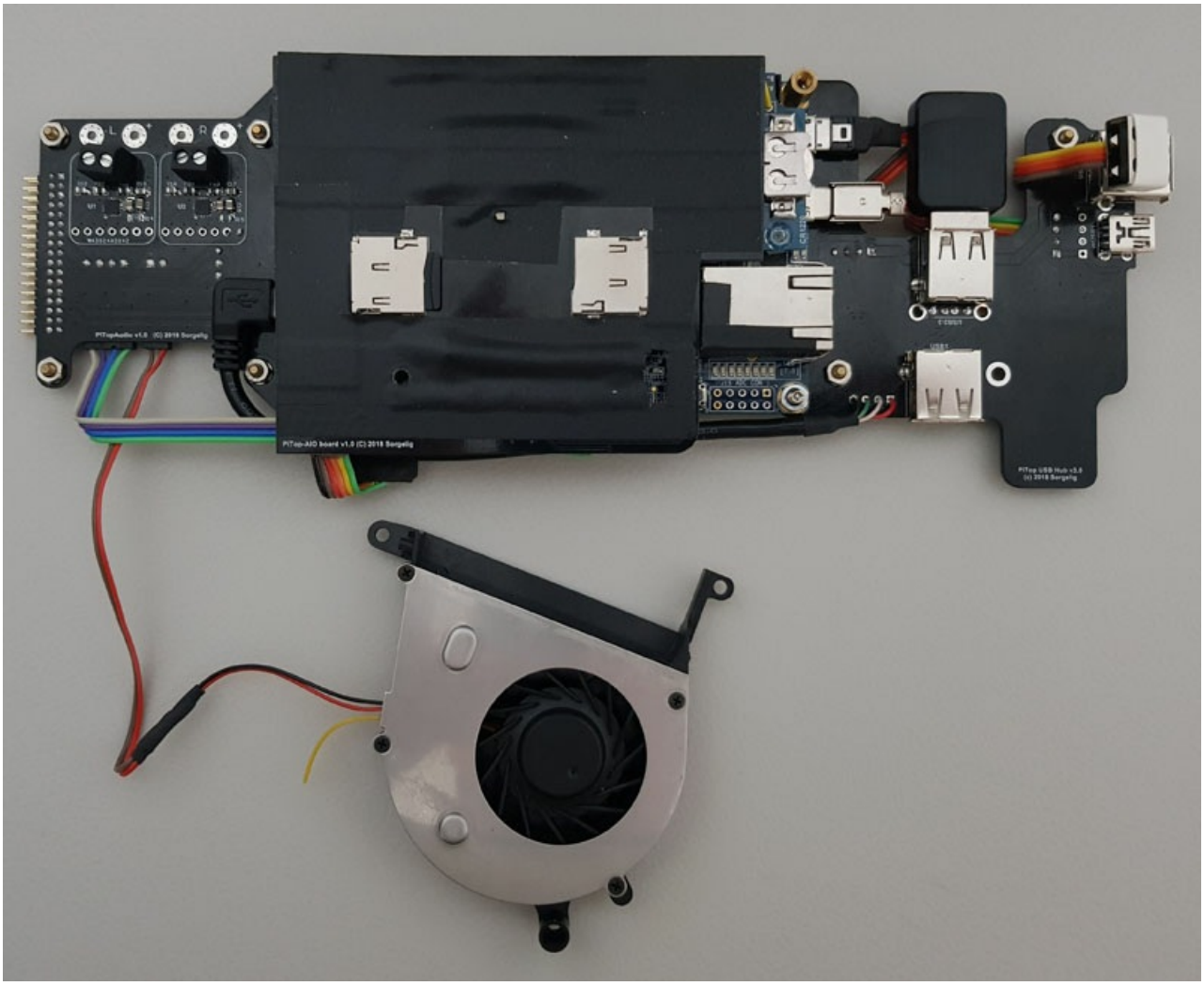
and blows to the side under AIO board all the way to hole on the right of PiTop case. It's hard to screw the fan. There are many different fans on the market. Currently used fan just lays unattached - surrounded parts just keep the fan in place.

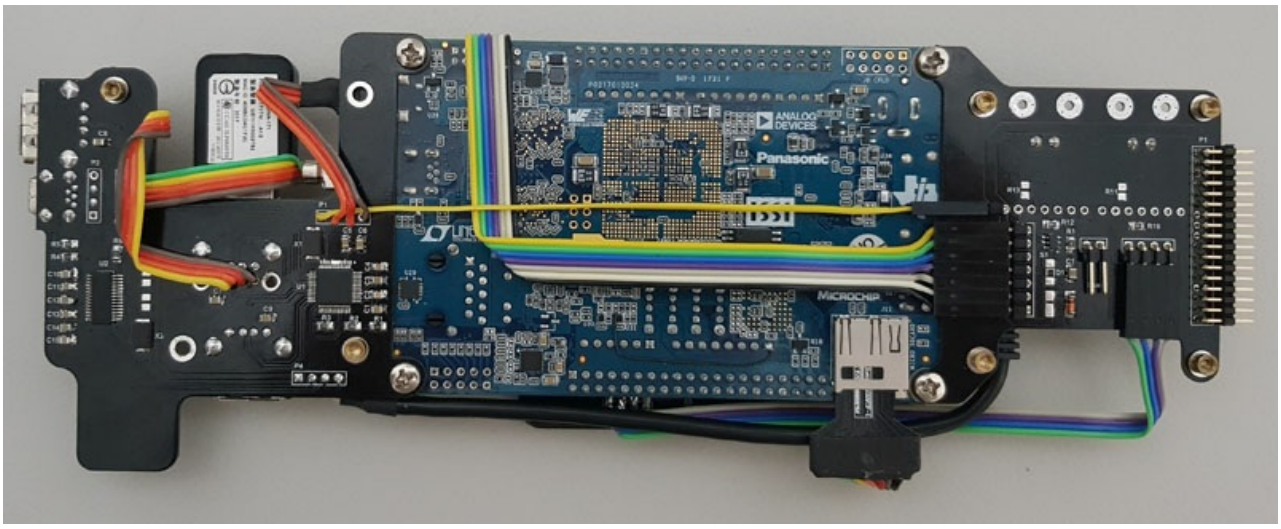
More pictures

You need to buy USB-to-barrel short cable to connect the power to DE10-nano board.

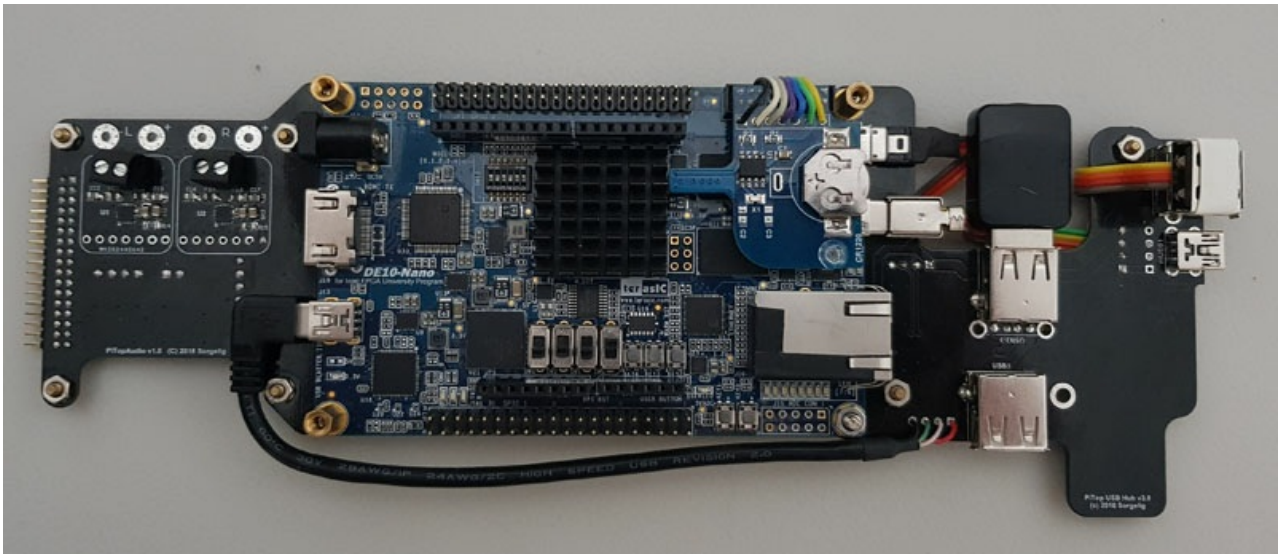


Optionally AIO board can be covered by black tape for additional protection.

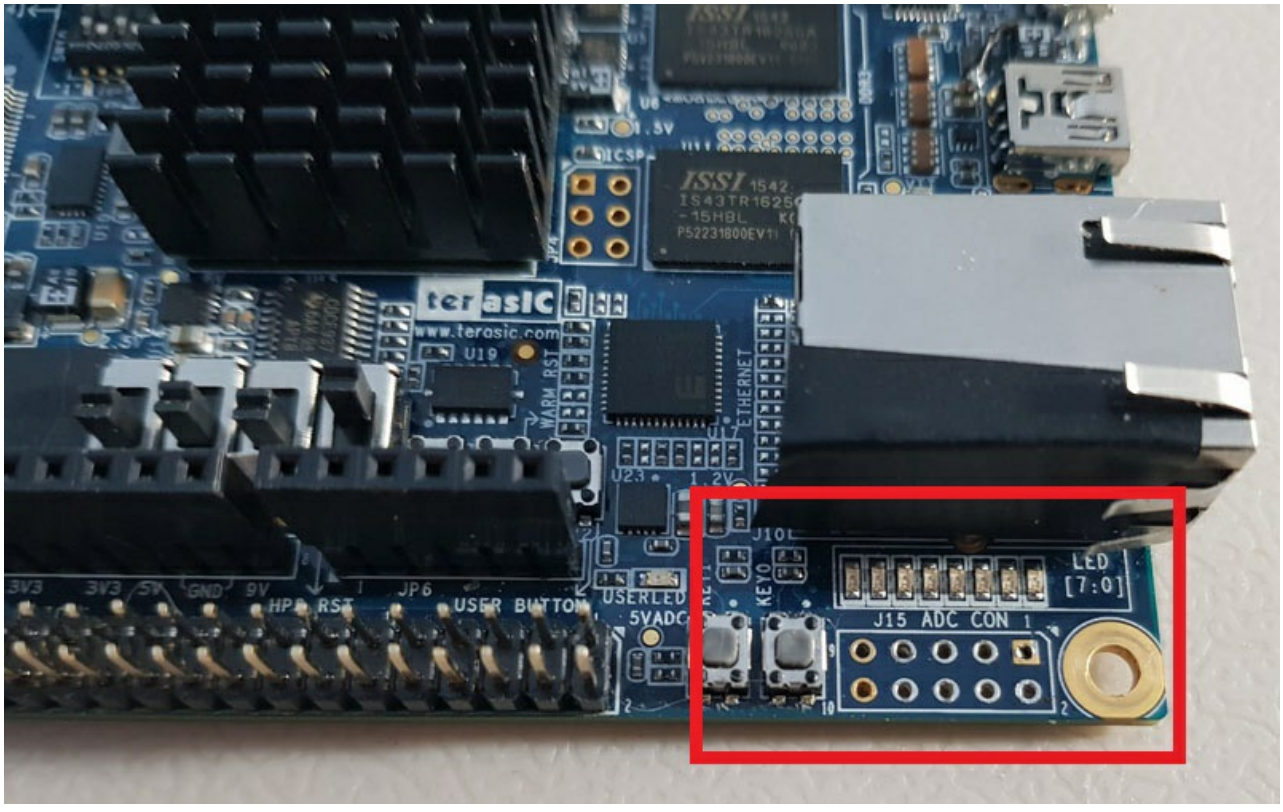




AIO board detached:



Unused ADC header has been removed from DE10-nano to improve LEDs view. Black tape attached to Ethernet connector is to reduce glow effect.

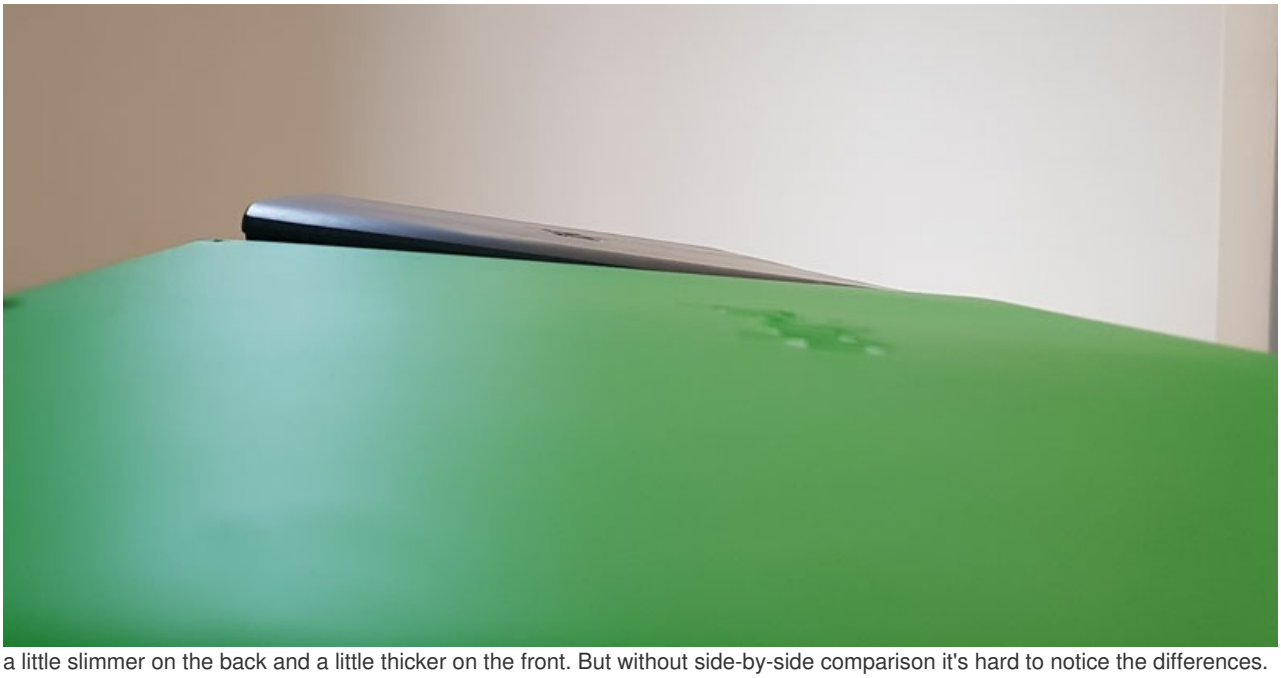


Bonus: v2 consideration

This is how bare DE10-nano fits inside v2 case:

Cover cannot be closed. Board doesn't fit. Picture shows the maximum closed position of the cover. Probably, after removing the metallic rail road and cutting the plastic ribs, DE10-nano will be able to fit. So, irreversible modification of the case will be required with drilling the holes. Most likely it's possible to use the same AIO and Audio boards. It will require other USB hub board to accommodate RPi holes on the back of the case.

Height comparison v1 vs v2:



Basically, v2 is a little slimmer on the back and a little thicker on the front. But without side-by-side comparison it's hard to notice the differences.

Overall comparison of v2 to v1:

- (-) Only green color available.
- (+) Slightly better tactile feeling of the case. It's not so slippery as v1.
- (+) Wider keyboard and bigger touchpad.

- (-) Cannot fit DE10-nano without butchering the case.
- (-) Non transparent cover. Cannot see any indication from the board.
- (-) Almost no holes (v1 has big holes around hinges), everything is tightly coupled -> needs to drill the holes for speakers.

Conclusion: use v2 for Raspberry Pi or Asus Tinker Board ;)

Arcade cores

- Arkanoid
- Asteroids
- Asteroids Deluxe
- Atari Tetris
- Bagman / Le Bagnard Inc: Botanic, Pickin', Super Bagman, Squash
- Bally Midway Astrocade / Bally Midway Arcade Inc. The Adventures of Robby Roto!, Extra Bases, Gorf, Sea Wolf II, Space Zap, Wizard of Wor
- Bally Midway MCR-1 Inc: Kick-Man, Kick, Solar Fox
- Bally Midway MCR-2 Inc: Domino Man, Kozmik Krooz'r, Satan's Hollow, Tron, Two Tigers, Wacko
- Bally Midway MCR-3 Inc: Discs of Tron, Journey, Tapper, Timber
- Bally Midway MCR-Monoboard Inc: Demolition Derby, Max RPM, Power Drive, Rampage, Sarge, Star Guards
- Bally Midway MCR-Scroll Inc: Crater Rider, Spy Hunter, Turbo Tag
- Berzerk
- Black Widow Inc: Gravitar, Lunar Battle
- Blockade Inc: CoMotion, Hustle, Blasto
- Bomb Jack
- Breakout
- Burger Time
- Burnin' Rubber / Car Action
- Canyon Bomber
- Cave: Inc: DoDonPachi
- Centipede
- Computer Space
- Crazy Balloon
- Crazy Climber
- Crazy Kong Inc: Crazy Kong Part II
- Defender Inc: Colony 7, Jin, Mayday
- Dig Dug
- Dominos
- Donkey Kong Inc: Radar Scope, Pest Place, Donkey Kong 3 (bootleg)
- Donkey Kong 3
- Donkey Kong Junior / Donkey Kong Jr.
- Dottori-Kun Inc: Dottori-Man Jr
- Finalizer - Super Transformation

- Food Fight
- Frenzy
- Galaga
- Galaxian Inc: Azurian Attack, Black Hole, Catacomb, Clean Sweep, Devil Fish, King And Balloon, Lucky Today, Moon Cresta, Mr. Do's Nightmare, Omega, Orbitron, Pisces, UniWar S, Victory, War of the Bugs
- Gaplus / Galaga 3
- Gauntlet Inc: Gauntlet, Gauntlet II, Vindicators II
- Gyruss
- Irem M62 Inc: The Battle-Road, Horizon, Kid Niki: Radical Ninja / Kaiketsu Yanchamaru, Kung-Fu Master / Spartan X, Lode Runner, Lode Runner II: The Bungling Strikes Back, Lode Runner III: Golden Labyrinth / Majin No Fukkatsu, Lode Runner IV: Teikoku Karano Dasshutsu, Lot Lot, Spelunker, Spelunker II, Youjyuden
- Iron Horse
- Jackal
- Jailbreak
- Joust 2: Survival of the Fittest
- Lady Bug Inc : Lady Bug, Snap Jack, Dorodon, Cosmic Avenger
- Lunar Lander
- Mario Bros
- Moon Patrol
- Mr. Do!
- Mystic Marathon
- Ninja-Kun: Majō no Bōken
- Pac-Man / Puck Man Inc: Alibaba and 40 Thieves, Birdiy, Crush Roller, Dream Shopper, Eeekk!, Eggor, Eyes, Gorkans, Jump Shot, Lizard Wizard, Mr. TNT, Ms. Pac-Man, Pac-Man Club, Pac-Man Plus, Pac Manic Miner Man, Ponpoko, Super Glob, Van-Van Car, Woodpecker
- Pengo
- Phoenix
- Pleiads
- PolyPlay
- Pong
- Pooyan
- Popeye Inc: Sky Skipper
- QBert
- River Patrol
- Rally-X Inc: New Rally-X
- Robotron 2084 Inc: Alien Arena, Bubbles, Joust, Playball, Sinistar, Splat, Stargate
- Rush'n Attack / Green Beret Inc: Mr. Goemon
- Scooter Shooter

- Scramble Inc: Amidar, Anteater, Armored Car, Battle of Atlantis, Calipso, Dark Planet, The End, Frogger, Lost Tomb, Mars, Mighty Monkey, Minefield, Moon War, Rescue, Speed Coin, Strategy X, Super Cobra, Tazz-Mania, Turtles
- Sega System 1 Inc: 4-D Warriors, Block Gal, Bullfight, Flicky, I'm Sorry, Mister Viking, My Hero, Pitfall II, Rafflesia, Regulus, Sega Ninja, Spatter, Star Jacker, Swat, TeddyBoy Blues, Up'n Down, Water Match, Wonder Boy
- Sega System E Hang-On Jr, Slap Shooter, Transformer, Pythgoras no Nazo, Opa Opa, Fantasy Zone II - The Tears of Opa-Opa, Tetris
- Silver Land
- Solomon's Key / Solomon no Kagi
- Space Invaders Inc: 280Z ZZAp, Amazing Maze, Attack Force, Balloon Bomber, Blue Shark, Boot Hill, Clowns, Cosmo, Galaxy Wars, Gun Fight, Laguna Racer, Lunar Rescue, Lupin III, Sea Wolf, Space Encounters, Space Invaders II, Space Invaders Part II, Vortex
- Space Race
- Sprint 1
- Sprint 2
- Subs
- Super Breakout
- Tecmo: Inc: Arugosu no Senshi: Legendary Warrior / Rygar: Legendary Warrior, Gemini Wing, Silkworm
- TIA-MC1: Inc: Billiard, Gorodki, Konek Gorbunok, Kot-Rybolov, Snezhnaja Koroleva, SOS
- Time Pilot
- Time Pilot '84
- The Tower of Druaga Inc: Dig Dug II, Mappy, Motos, Super Pacman, Pac & Pal, Pac-Man & Chomp Chomp, Grobda
- Traverse USA / MotoRace USA / Zippy Race / Mototour: Inc: Shot Rider
- Ultra Tank
- Universal Cosmic games Inc. Space Panic, Cosmic Alien, Magical Spot
- VBall
- Xevious
- Zaxxon Inc: Super Zaxxon, Future Spy
- Zig Zag

MRA format description

- Arcade game and systems hardware information - System 16: The Arcade Museum
- Archive of original arcade manuals - The Arcade Manual Archive

Core porting notes

Introduction

MiSTer uses quite complex hardware, but thanks to open source, developers with different levels of hardware/software knowledge can develop for this platform. Many cores for MiSTer use common almost identical part of code simplifying access to hardware and firmware called **Framework**. We will use ZX Spectrum core in this guide.

Framework

Most of the sources of the Framework are located in sys folder. Usually, for a new project, you need to take following files/folders

- sys folder
- jtag.cdf
- jtag_lite.cdf
- zxspectrum.qpf (project file. Keep it read-only to prevent it from automatic changes whenever you switch between fill and lite versions and spam your change history)
- zxspectrum.qsf
- zxspectrum.srf (some warnings ignores)
- zxspectrum-lite.qsf
- zxspectrum-lite.srf (some warnings ignores)

You need to make some changes:

- Rename zxspectrum.* files according to a new project.
- Inside files find the "zxspectrum" word and replace it with the name of your project.
- in *.qsf files at the end you will find the list of project files. Remove files not related to your project.

The initial set of files for the new project is ready. Now you can open the project in Quartus 17.0 or higher. Assuming you are porting some existing core to MiSTer, the top module entity should be renamed to emu. See zxspectrum.sv and its input/output signals:

```
module emu
(
   //Master input clock
   input         CLK_50M,

   //Async reset from top-level module.
   //Can be used as initial reset.
   input         RESET,

   //Must be passed to hps_io module
   inout  [45:0] HPS_BUS,

   //Base video clock. Usually equals to CLK_SYS.
   output        CLK_VIDEO,

   //Multiple resolutions are supported using different CE_PIXEL rates.
   //Must be based on CLK_VIDEO
   output        CE_PIXEL,

   //Video aspect ratio for HDMI. Most retro systems have ratio 4:3.
   //if VIDEO_ARX[12] or VIDEO_ARY[12] is set then [11:0] contains scaled size instead of aspect ratio.
   output [12:0] VIDEO_ARX,
   output [12:0] VIDEO_ARY,

   output  [7:0] VGA_R,
   output  [7:0] VGA_G,
   output  [7:0] VGA_B,
   output        VGA_HS,
   output        VGA_VS,
   output        VGA_DE,    // = ~(VBlank | HBlank)

   output        LED_USER,  // 1 - ON, 0 - OFF.

   // b[1]: 0 - LED status is system status ORed with b[0]
   //       1 - LED status is controled solely by b[0]
   // hint: supply 2'b00 to let the system control the LED.
   output  [1:0] LED_POWER,
   output  [1:0] LED_DISK,

   output [15:0] AUDIO_L,
   output [15:0] AUDIO_R,
   output        AUDIO_S, // 1 - signed audio samples, 0 - unsigned


   //High latency DDR3 RAM interface
   //Use for non-critical time purposes
   output        DDRAM_CLK,
   input         DDRAM_BUSY,
   output  [7:0] DDRAM_BURSTCNT,
   output [28:0] DDRAM_ADDR,
   input  [63:0] DDRAM_DOUT,
   input         DDRAM_DOUT_READY,
   output        DDRAM_RD,
   output [63:0] DDRAM_DIN,
   output  [7:0] DDRAM_BE,
   output        DDRAM_WE,

   //SDRAM interface with lower latency
   output        SDRAM_CLK,
   output        SDRAM_CKE,
   output [12:0] SDRAM_A,
   output  [1:0] SDRAM_BA,
   inout  [15:0] SDRAM_DQ,
   output        SDRAM_DQML,
   output        SDRAM_DQMH,
   output        SDRAM_nCS,
   output        SDRAM_nCAS,
   output        SDRAM_nRAS,
   output        SDRAM_nWE

      // there are more needed signals - see the Template_MiSTer
);
```

These signals are main connections to MiSTer. You need to modify your core according to these signals.

API

Most top-level signals should be self-descriptive and more or less familiar to core developers, so I won't go too deep into it. I will describe some important parts where you need to pay attention to make your core work.

```
input   RESET
```

This signal is asserted while ARM part is under initial preparation. It can be used as an initial reset. It won't be asserted anymore during the whole work of core except when the user chooses another core from OSD. Then ARM will assert RESET in the existing core to let it stop any possible activity before switching to another core (cores loaded through USB Blaster won't get "bye-bye" RESET). If core uses DDR3 memory (Scaler isn't counted) then initial and bye-bye RESET is crucial for correct work. You will get a hard hang if DDR3 is accessed during RESET.

```
inout  [45:0] HPS_BUS
```

Pass it as-is to hps_io module.

```
output        CLK_VIDEO,
output        CE_PIXEL,
output  [12:0] VIDEO_ARX,
output  [12:0] VIDEO_ARY,
output        VGA_DE,
```

Besides other well-known VGA_* signals, these signals are important in MiSTer in order to get a proper display. CLK_VIDEO and CE_PIXEL are used to sample the pixels. If pixel rate equals to CLK_VIDEO, then set CE_PIXEL=1. Many cores have common system clock where pixel frequency is the product of the division of system clock to some number. In this case, set CLK)VIDEO to the system clock and assert CE_PIXEL at clock cycles where a new pixel is produced. Look in zxspectrum.sv to see how it's done. without CLK_VIDEO, you can still have VGA output, but OSD won't work.

VIDEO_ARX and **VIDEO_ARY** define aspect ratio on HDMI output. It doesn't affect VGA output. Usually VIDEO_ARX=4, VIDEO_ARY=3 or VIDEO_ARX=16, VIDEO_ARY=9. They can have other values but with extreme values, you may have video problems.

VGA_DE is another crucial part for MiSTer. Basically, it's opposite to blank signals: `~(VBlank | HBlank)` but with some notes. HDMI video scaler detects the horizontal resolution by first active video line. So, you need to be sure the first line is not cut due to unaligned VBlank and HBlank signals to each other. Otherwise, you will get a messed HDMI video.

VGA_HS and **VGA_VS** should have positive pulse polarity.

```
output        AUDIO_S
```

Make sure you set correct mode signed/unsigned, otherwise audio will be distorted.

DDRAM_ are signals of DDR3 memory. If the core can use this memory instead of SDRAM, then it won't require SDRAM board.

SDRAM_ are signals of SDR SDRAM memory. The core will require SDRAM board if it uses these signals.

HPS_IO

There is a supplementary module **hps_io.v** which is also required to use in the same entity (see zxspectrum.sv). It provides in/out control from ARM side. Most signals are same or similar to MiST (user_io+data_io or mist_io modules) signals. So, if the core is ported from MiST, it in most cases it's 1:1 signal connections.

MiSTer has some changes/improvements over original MiST io modules:

- Supports up to 4 images mount at the same time (MiST has only 1). Set module parameter VDNUM to 2..4 if more than 1 mounted image is required. If VDNUM=1 then related signals are same as in MiST.
- Due to SD card on MiSTer uses multiple partitions and holds other vital to MiSTer parts, access to whole SD card from cores is not available in MiSTer. Only the access to image files is possible. Cores requiring direct access to whole SD card should be redesigned to access to images only.
- Main MiSTer file system on SD card is exFAT which supports files bigger than 4GB.
- OSD supports up to 15 lines (7 lines in MiST) which is handy for many cores.
- ARM<->FPGA communication is done through the parallel bus which speeds up the communication. It supports 16bit I/O.

There are some other under the hood improvements in firmware like Keyboard/Mouse/Joystick setup.

Core configuration string

MiSTer provides an on-screen display (OSD) that can be toggled on and off by pressing F12 on your keyboard (or OSD button). For each core that is loaded, this menu can be configured to add specific options for that core.

The top-level module for the cores is `emu`. This does **NOT** mean `emu` is the top-level module for the project, but rather it is the top-level module for our purposes. The `emu` module is typically found in the SystemVerilog file (`*.sv` extension) with the same name as the project. As an example, the Arcade-Galaga project has it's top-level module at `Arcade-Galaga.sv`.

The configuration string is stored in the variable `CONF_STR` of the `emu` module. This variable is passed to the `hps_io` module that handles sending it to the processor to be read when necessary.

Each line of the configuration string is delimited with a semicolon.

The first line is the core name followed by 2 semicolons. The core name is also used e.g. as the location inside the `games` folder.

- (Optional) `UART[{speed...}]` - Enables UART menu.
    - (Optional) `{speed}` - a list of numeric interface speed values separated by a colon `:`. Defaults are `110, 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 31250, 38400, 57600, 115200`.
- (Optional) `MIDI[{speed}]` - Enables MIDI menu.
    - (Optional) `{speed}` - a list of numeric interface speed values separated by a colon `:`. Default is `31250`.
- (Optional) `SS{base}:{size}` - Savestate support for memory area designated by `{base}` address and `{size}`.

Valid options for the menu (`[]` - means optional parameter):

- `C[,{Text}]` - Enables a cheat menu entry with the label `{Text}`.
- `CHEAT` - like DIP, for arcades with a cheat system
- `D{Index}` - Prefix which disables the option if `menumask[Index]` is set.
- `d{Index}` - Same as 'D', but disables the option if `menumask[Index]` is NOT set.
- `DIP` - in arcade cores, this will display the DIP menu from the MRA
- `F[S][#],{Ext}[,{Text}][,{Address}]` - Load file button.
    - `{Ext}` - a concatenated list of 3 character file extensions. For example, `BINGEN` would be `BIN` and `GEN` extensions.
    - Optional `{Text}` string is the text that is displayed before the extensions like "Load RAM". If `{Text}` is not specified, then default is "Load *".
    - Optional `[S]` - core supports save files, load a file, and mount a save for reading or writing
    - `#` is explicit index (or index is generated from line number if index not given).
    - Optional `{Address}` - load file directly into DDRAM at this address
    - ioctl_index from hps_io will be: ioctl_index[5:0] = index(explicit or auto), ioctl_index[7:6] = extension index
- `FC[#],{Ext}[,{Text}][,{Address}]` - Open file and remember it, useful for remembering an alternative rom, config, or other type of file. See F for how the options work.
- `H{Index}` - Prefix which hides the option if `menumask[Index]` is set.
- `h{Index}` - Same as `H`, but hides the option if `menumask[Index]` is NOT set.
- `O{Index1}[{Index2}],{Name},{Options...}` - Option button that allows you to select between various choices.
    - `{Index1}` and `{Index2}` are values from 0-9 and A-V (like Hex but it extends from A-V instead of A-F). This represents all 31 bits. First and second index are the range of bits that will be set in the status register.
    - `{Name}` is what is shown to describe the option.

    - `{Options...}` - a list of comma separated options.
- `P{#},{Title}` - Creates sub-page for options with `{Title}`.
- `P{#}` - Prefix to place the option into specific `{#}` page. This is added before `O#` but after something like `d#`. (e.g. `"d5P1o2,Vertical Crop,Disabled,216p(5x);"` is correct and `"P1d5o2,Vertical Crop,Disabled,216p(5x);"` is incorrect and the menu options will not work.)
- `R{Index},{Name}` - Same as T option but closes the OSD after selecting. Convenient for Reset option.
- `S{Slot},{Ext}[,{Text}]` - Mount SD card button.
    - `{Slot}` - a value from 0-3. Up to four images can be mounted at the same time.
    - `{Ext}` - same as in `F` option.
    - (Optional) `{Text}` - The text that is displayed before the extensions like "Load RAM". If `{Text}` is not specified, then default is "Mount *".
- `T{Index},{Name}` - Trigger button. This is a simple button that will pulse HIGH of specified `{Index}` bit in status register. A perfect example of this is for a reset button.
    - `{Name}` is the text that describes the button function.
- `-[,TEXT]` - empty (or with `TEXT`) line.
- Lower case options `o`, `t`, `r` equal their upper case variants with adding 32 to status bit indexes.

For example:

```
// Status Bit Map:
//             Upper                    Lower
// 0       1       2       3       4       5       6
// 01234567890123456789012345678901 23456789012345678901234567890123
// 0123456789ABCDEFGHIJKLMNOPQRSTUV 0123456789ABCDEFGHIJKLMNOPQRSTUV
//
```

If you add a status option like `O89` this is going to use the first group of status bits on the left since the "O" for option is capitalized (marked Upper). Then it's going to use 8 and 9, this is from the bottom row (which is alphanumeric). So you would update the index after using these like so:

```
// Status Bit Map:
//             Upper                    Lower
// 0       1       2       3       4       5       6
// 01234567890123456789012345678901 23456789012345678901234567890123
// 0123456789ABCDEFGHIJKLMNOPQRSTUV 0123456789ABCDEFGHIJKLMNOPQRSTUV
//       XX
```

Non-OSD options (must be placed at bottom of configuration string):

- `J[1],{Button1}[,{Button2},...]` - J1 means lock keyboard to joystick emulation mode. Useful for keyboard-less systems such as consoles. {Button1}, {Button2},... is list of joystick buttons used in the core. Up to 12 buttons can be listed. Analog axis are not defined here. The user just needs to map them through the Menu core.
- `jn,{SNES Button Name1},[,{SNES Button2},...]` - this sets the default mapping of the buttons. ie: jn,A would map joystick bit 4 to the A button on a SNES style controller automatically
- `jp` - same as `jn` but used when `gamepad_defaults=1` in `MiSTer.INI`. Typically refers to positional mapping relative to a SNES controller
- `V,{Version String}` - Version string. {Version String} is the version string. Takes the core name and appends version string for name to display.
- `I,INFO1,INFO2,...,INFO255` - `INFO1-INFO255` lines to display as OSD info (top left corner of screen).
- `DEFMRA,{mra name}` - default MRA (ie: Puckman.mra) to be used when core is uploaded by USB blaster (debug)

jn vs. jp and mapping conventions

- jn mapping is the default.
- jp mapping is used when the INI file has gamepad_defaults=1

- the only difference in the two mappings is the "philosophy" of how they are expected to work

The difference is one of convention when mapping buttons:

- jn is name-base mapping
- jp is position-based mapping

What does this mean?

Consider how the internal gamepad is defined on MiSTer:

1. Internally, controllers are mapped in the menu core to a SNES-style gamepad
2. The button order on the internal MiSTer gamepad is ABXYLR + Start + Select
3. That is: button 1=A, button 2=B, and so on.

With "jn" mapping:

- the convention is to map button A of the core to the internal "SNES A" button.
- If the core has no button "A", this would be the first button.
- Second button would be "SNES B", and so forth.
- This is because the name or order of the button matches the original definition (ABYXLR etc)

With "jp" mapping:

- the convention is to consider the physical location of each button
- For a Genesis 3-button controller that has A, B, C as buttons, mapping would be "SNES Y, SNES B, SNES A"
- This is because YBA on a SENS gamepad are the lower 3 buttons of the controller

emu Top Level of a MiSTer core

Introduction

On MiSTER the top level is inside sys/sys_top.v When writing or porting a MiSTer core, instead of telling quartus to use your own top, quartus uses sys_top, and it will call a module called emu that you need to provide. This will document the parameters that your top level will need to implement. The MiSTer sys_top will handle a lot of useful things like audio and HDMI video.

Instantiating your core

your core should be inside an emu block, see the Template Top Level for an example.

Master input clock

```
//Master input clock
input        CLK_50M,
```

Most cores will use the PLL which is instantiated from the sys folder, but the pll needs to live in the rtl folder off of the top level (at the same level as sys). This choice was made so that you can update the sys folder without losing the PLL configuration.

NOTE: For full compatibility with the existing sys architecture, keep the pll named 'pll' and the instance 'pll' as shown below. You can use the megawizard to edit the pll to modify the required frequencies etc... (including adding more outputs). If you don't have a pll create a new one, or use the one from the template. The sys_top.sdc requires the name for the pll and the instance name be correct.

```
wire clk_sys;
pll pll
(
    .refclk(CLK_50M),
    .rst(0),
    .outclk_0(clk_sys)
);
```

Reset

```
//Async reset from top-level module.
//Can be used as initial reset.
input        RESET,
```

Normally the RESET line is used in conjunction with the button on the I/O board, as well as a status flag. Most cores use status bit 0, and a CONF_STR that includes an option like: `"R0,Reset;"`

```
wire reset = RESET | status[0] | buttons[1];
```

HPS Bus

The HPS Bus is used to communicate with the ARM processor. It is passed into the hps_io module which will do a bunch of work for the core, and provide a simpler interface to use. Include the hps_io module in the emu module. See hps_io

```
//Must be passed to hps_io module
inout  [45:0] HPS_BUS,
```

Video Signals

The top level module that calls emu includes the high quality HDMI scaler, and provides a fairly simple interface to output video on MiSTer. There are a number of other helper modules that can be included to provide proper video output.

At the simplest level provide a CLK_VIDEO, and/or CE_PIXEL (CE_PIXEL can be set to 1 if your video clk is correct). The video clock needs to be greater than 40MHZ for all of the features to work.

Generally the video signals want to be whatever the native resolution of the machine that is being emulated. ie: an old 8bit computer will generally output 15khz video. MiSTer has a scandoubler built in that will create VGA compatible output if the user wants that on their VGA port. To include this scandoubler in the core you will usually use the video_mixer or arcade_video modules. The forced_scandoubler signal comes from the hps_io

module. Some users want to use a 15khz monitor with 15khz cores, so the core itself shouldn't have a scandoubler built in. Some original devices may have had higher resolution output, if that native output was greater than 15khz, it is ok to output it natively through the VGA signals. The scaler will use the VGA_DE (usually based on HBLANK and VBLANK `assign VGA_DE = ~(HBlank | VBlank);`) to scale it up to HDMI based on the settings in the ini file.

Video Mixer

Use the Video Mixer Module to add mister framework standard features like gamma support, scaling, and scandoubling.

Video Freak

Use the Video Freak Module to add cropping and scaling

Arcade Video

Use the Arcade Video Module for arcade cores to support rotation, scandoubling, and most of the features that are in video_mixer. This file also includes **screen_rotate** which is used for vertical arcade games to allow rotation over HDMI.

```
    //Base video clock. Usually equals to CLK_SYS.
    output        CLK_VIDEO,

    //Multiple resolutions are supported using different CE_PIXEL rates.
    //Must be based on CLK_VIDEO
    output        CE_PIXEL,

    //Video aspect ratio for HDMI. Most retro systems have ratio 4:3.
    //if VIDEO_ARX[12] or VIDEO_ARY[12] is set then [11:0] contains scaled size instead of aspect ratio.
    output [12:0] VIDEO_ARX,
    output [12:0] VIDEO_ARY,

    output  [7:0] VGA_R,
    output  [7:0] VGA_G,
    output  [7:0] VGA_B,
    output        VGA_HS,
    output        VGA_VS,
    output        VGA_DE,    // = ~(VBlank | HBlank)
    output        VGA_F1,
    output [1:0]  VGA_SL,
    output        VGA_SCALER, // Force VGA scaler

    input  [11:0] HDMI_WIDTH,
    input  [11:0] HDMI_HEIGHT,
    output        HDMI_FREEZE,

`ifdef MISTER_FB
    // Use framebuffer in DDRAM (USE_FB=1 in qsf)
    // FB_FORMAT:
    //    [2:0] : 011=8bpp(palette) 100=16bpp 101=24bpp 110=32bpp
    //    [3]   : 0=16bits 565 1=16bits 1555
    //    [4]   : 0=RGB  1=BGR (for 16/24/32 modes)
    //
    // FB_STRIDE either 0 (rounded to 256 bytes) or multiple of pixel size (in bytes)
    output        FB_EN,
    output  [4:0] FB_FORMAT,
    output [11:0] FB_WIDTH,
    output [11:0] FB_HEIGHT,
    output [31:0] FB_BASE,
    output [13:0] FB_STRIDE,
    input         FB_VBL,
    input         FB_LL,
    output        FB_FORCE_BLANK,

`ifdef MISTER_FB_PALETTE
    // Palette control for 8bit modes.
    // Ignored for other video modes.
    output        FB_PAL_CLK,
    output  [7:0] FB_PAL_ADDR,
    output [23:0] FB_PAL_DOUT,
    input  [23:0] FB_PAL_DIN,
    output        FB_PAL_WR,
`endif
`endif
```

LEDs on the IO board

```
output      LED_USER,  // 1 - ON, 0 - OFF.

// b[1]: 0 - LED status is system status OR'd with b[0]
//       1 - LED status is controled solely by b[0]
// hint: supply 2'b00 to let the system control the LED.
output  [1:0] LED_POWER,
output  [1:0] LED_DISK,
```

Buttons on IO board

```
// I/O board button press simulation (active high)
// b[1]: user button
// b[0]: osd button
output  [1:0] BUTTONS,
```

Audio

```
input        CLK_AUDIO, // 24.576 MHz
output [15:0] AUDIO_L,
output [15:0] AUDIO_R,
output       AUDIO_S,   // 1 - signed audio samples, 0 - unsigned
output  [1:0] AUDIO_MIX, // 0 - no mix, 1 - 25%, 2 - 50%, 3 - 100% (mono)
```

A to D converter

Used for tape input and other things. An extra module on MiSTer that provides an analog audio jack, and conversion to digial signals.

```
//ADC
inout   [3:0] ADC_BUS,
```

Signals for the second SD card ?

```
//SD-SPI
output       SD_SCK,
output       SD_MOSI,
input        SD_MISO,
output       SD_CS,
input        SD_CD,
```

DDR3 Memory Subsystem

```
//High latency DDR3 RAM interface
//Use for non-critical time purposes
output       DDRAM_CLK,        // any clock, no restrictions. Typically main core clock
input        DDRAM_BUSY,       // every read and write request is only accepted in a cycle where busy is low
output  [7:0] DDRAM_BURSTCNT,  // amount of words to be written/read. Maximum is 128
output [28:0] DDRAM_ADDR,      // starting address for read/write. In case of burst, the addresses will internally count up
input  [63:0] DDRAM_DOUT,      // data coming from (burst) read
input        DDRAM_DOUT_READY, // high for 1 clock cycle for every 64 bit dataword requested via (burst) read request
output       DDRAM_RD,         // request read at DDRAM_ADDR and DDRAM_BURSTCNT length
output [63:0] DDRAM_DIN,       // data word to be written
output  [7:0] DDRAM_BE,        // byte enable for each of the 8 bytes in DDRAM_DIN, only used for writing. (1=write, 0=ignore)
output       DDRAM_WE,         // request write at DDRAM_ADDR with DDRAM_DIN data and DDRAM_BE mask
```

Writing

The internal DDR3 controller handles writes very efficiently, so burst writes are typically not required. To write, DDRAM_WE should be high for 1 clock cycle whenever DDRAM_BUSY is low. It will write the data in DDRAM_DIN to DDRAM_ADDR with respect to DDRAM_BE. For a single write, DDRAM_BURSTCNT should be 1. Multiple writes can also be issued without pausing when DDRAM_BURSTCNT = 1.

Reading

To read one or multiple 64 bit words, DDRAM_RD must be high for 1 clock cycle, while DDRAM_BUSY is low. It will read DDRAM_BURSTCNT 64 bit words from DDRAM_ADDR(counting up for bursts) and provide the results, typically one each clock cycle, at DDRAM_DOUT with DDRAM_DOUT_READY = 1, when the read is valid. Every read request will have a latency of multiple cycles. Something like 20 cycles @ 100Mhz is a typical value, but it can be way longer. DDR3 read should therefore be used with higher DDRAM_BURSTCNT to make use of the high bandwidth, whenever possible.

Busy

DDRAM_BUSY acts like an ignore for the request ports. So whenever this signal is high, no request can be issued in this clock cycle. However, having a request pending doesn't lead to any problem, so it is uncritical to e.g. set DDRAM_RD = 1 in a clocked process and only clear it back to 0 when DDRAM_BUSY = 0. This way, the request signals can be clocked instead of being combinatorial, leading to higher possible clock speed and less problems with timing closure.

Single and Dual SDRAM interface

For full understanding of the SDRAM interface you will need to look at the specifications for the SDRAM chip. Also, the hardware may be useful, here is the Schematic. It is useful to look through some code that already runs on MiSTer:

Look through the data sheet for the 32MB module. The 64MB module data sheet isn't as detailed.

Some example SDRAM modules:

- single port direct usage: Gameboy
- multi port request/response: Gameboy Advance
- complex bank machine: JT Frame

```
    //SDRAM interface with lower latency
    output        SDRAM_CLK,
    output        SDRAM_CKE,
    output [12:0] SDRAM_A,
    output  [1:0] SDRAM_BA,
    inout  [15:0] SDRAM_DQ,
    output        SDRAM_DQML,
    output        SDRAM_DQMH,
    output        SDRAM_nCS,
    output        SDRAM_nCAS,
    output        SDRAM_nRAS,
    output        SDRAM_nWE,

`ifdef MISTER_DUAL_SDRAM
    //Secondary SDRAM
    //Set all output SDRAM_* signals to Z ASAP if SDRAM2_EN is 0
    input         SDRAM2_EN,
    output        SDRAM2_CLK,
    output [12:0] SDRAM2_A,
    output  [1:0] SDRAM2_BA,
    inout  [15:0] SDRAM2_DQ,
    output        SDRAM2_nCS,
    output        SDRAM2_nCAS,
    output        SDRAM2_nRAS,
    output        SDRAM2_nWE,
`endif
```

Serial Support

Serial is passed to the linux arm side of the MiSTer. On the arm side, software decides what to do with the data. ie: send it to shell, ppp, MIDI, etc.

Different serial speeds and optons are set using options in the CONF_STR.

```
    input         UART_CTS,
    output        UART_RTS,
    input         UART_RXD,
    output        UART_TXD,
    output        UART_DTR,
    input         UART_DSR,
```

User Port - extra USB 3.1A style connector on MiSTer

USB	P7	Name	PIN	Mister	emu wire
1	+5V	+5V			
2	2	TX	SDA	AH9	USER_IO[1]

USB	P7	Name	PIN	Mister	emu wire
3	1	RX	SCL	AG11	USER_IO[0]
4	GND	GND			
5	8	DSR	IO10	AF15	USER_IO[5]
6	7	DTR	IO11	AG16	USER_IO[4]
7	6	CTS	IO12	AH11	USER_IO[3]
8	5	RTS	IO13	AH12	USER_IO[2]
9	10	IO6	IO8	AF17	USER_IO[6]
10	Shield	Shield			

```
// Open-drain User port.
// 0 - D+/RX
// 1 - D-/TX
// 2..6 - USR2..USR6
// Set USER_OUT to 1 to read from USER_IN.
input  [6:0] USER_IN,
output [6:0] USER_OUT,
```

OSD Status

This is set to 1 when the OSD is open. It can be used to pause the core when the OSD is open, and/or for autosave.

```
input      OSD_STATUS
```

sys - hps_io

Introduction

HPS stands for Hard Processor System, or the ARM part of the MiSTer FPGA. The hps_io.sv file contains the hps_io module that talks to the MiSTer binary on the arm side. It provides convenient signals to use in each core that abstract the different types of I/O that the core might need to support.

Instantiating hps_io

The module has a few parameters that are used to set it up.

- CONF_STR - This is how the HPS reads the OSD configuration string
- PS2DIV - This divides the clk_sys to create a slower clock for the legacy ps2 signals: ps2_kbd_clk_out etc
- WIDE - This makes the ioctl data and sd_buff signals 8 bit (WIDE=0) or 16 bit (WIDE=1) if it is wide, ioctl_addr is incremented by 2
- VDNUM - Virtual Device Number - This can be 1 to 4, and will create extra virtual block devices
- BLKSZ - Block Size - set the block size of the block device - 0 = 128, 1 = 256, 2 = 512(default), .. 7 = 16384
- PS2WE - PS2 Write Enable - option to use 2 way PS/2 communications. See ao486 core.

clk_sys is the system clock. Make sure to use the same clock when reading from these signals. HPS_BUS should be passed through from the top level emu.

```
//
// Use buffer to access SD card. It's time-critical part.
//
// WIDE=1 for 16 bit file I/O
// VDNUM 1..4
// BLKSZ 0..7: 0 = 128, 1 = 256, 2 = 512(default), .. 7 = 16384
//
module hps_io #(parameter CONF_STR, CONF_STR_BRAM=1, PS2DIV=0, WIDE=0, VDNUM=1, BLKSZ=2, PS2WE=0)
(
   input         clk_sys,
   inout  [45:0] HPS_BUS,
```

Joystick input

MiSTer supports up to 6 players on separate joysticks. joystick_0..5 are digital joysticks. They contain the directions in the first 4 bits, and then they have buttons that can be mapped using the CONF_STR J.

- joystick_x[0] - right
- joystick_x[1] - left
- joystick_x[2] - down
- joystick_x[3] - up

Analog joysticks are supported by connecting joystick_l_analog_x. The values are analog -127..+127, Y: [15:8], X: [7:0] . Depending on how you want to use the values, you may need to renormalize them. -128 is omitted so it is easier to invert the direction. l is the left joystick, r is the right joystick on a playstation style controller.

Paddles - paddle_x - are input devices that have a range from 0 to 255. They do not spin fully.

Spinners - spinner_x - are a paddle looking device, but there is no stop in the hardware, they spin freely. Therefore, these use an extra bit 8 that toggles with each update. the value is in 0:7, -128..+127 - they are relative, and usually the value will be between -8 and 8.

```
// buttons up to 32
output reg [31:0] joystick_0,
output reg [31:0] joystick_1,
output reg [31:0] joystick_2,
output reg [31:0] joystick_3,
output reg [31:0] joystick_4,
output reg [31:0] joystick_5,

// analog -127..+127, Y: [15:8], X: [7:0]
output reg [15:0] joystick_l_analog_0,
output reg [15:0] joystick_l_analog_1,
output reg [15:0] joystick_l_analog_2,
output reg [15:0] joystick_l_analog_3,
output reg [15:0] joystick_l_analog_4,
output reg [15:0] joystick_l_analog_5,

output reg [15:0] joystick_r_analog_0,
output reg [15:0] joystick_r_analog_1,
output reg [15:0] joystick_r_analog_2,
output reg [15:0] joystick_r_analog_3,
output reg [15:0] joystick_r_analog_4,
output reg [15:0] joystick_r_analog_5,

// paddle 0..255
output reg  [7:0] paddle_0,
output reg  [7:0] paddle_1,
output reg  [7:0] paddle_2,
output reg  [7:0] paddle_3,
output reg  [7:0] paddle_4,
output reg  [7:0] paddle_5,

// spinner [7:0] -128..+127, [8] - toggle with every update
output reg  [8:0] spinner_0,
output reg  [8:0] spinner_1,
output reg  [8:0] spinner_2,
output reg  [8:0] spinner_3,
output reg  [8:0] spinner_4,
output reg  [8:0] spinner_5,
```

Buttons

Buttons from the top level emu are what the hardware sees. Buttons from the hps_io can include emulated button presses from the OSD, HPS, etc.

- buttons[1] is the user button
- buttons[0] is the OSD button

```
// I/O board button press simulation (active high)
// b[1]: user button
// b[0]: osd button
output      [1:0] buttons,
```

Forced Scandoubler

Forced Scandoubler is set to 1 when the user wants the scandoubler turned on.

```
output        forced_scandoubler,
```

Direct Video

Direct Video is set to 1 when the MiSTer is using the hdmi as a VGA port with a proper converter.

```
output        direct_video,
```

Status

The status bits are used by the OSD to provide the core with status that the user picks in the OSD. For example, the status string:

```
"O1,Colors,NTSC,PAL;",
```

would set status[1] to 0 for NTSC and 1 for PAL. See the documentation for the CONF_STR to understand all of the OSD options.

- status_menumask - this is used to tell the OSD to turn on and off options using the H or h option. Set bit 0 to effect H0.
- status_in, status_set - when status_set is 1, the hps_io will grab the status_in and use it to change the status bits

```
output reg [63:0] status,
input      [63:0] status_in,
input             status_set,
input      [15:0] status_menumask,
```

Info

- info - info string to display from CONF_STR
- info_req - ask HPS to display info string on OSD

```
input             info_req,
input      [7:0] info,
```

Force a new Video mode

- new_vmode - if the core needs to force the system to change video modes, it can set this flag

```
//toggle to force notify of video mode change
input             new_vmode,
```

Virtual Hard Drives and Block Devices

Hard drive images are loaded with the S option in the OSD. There can be up to 10 images mounted at once if VDUM is set to 10. 1-10 are valid values.

When an image is mounted a bit is set in img_mounted, and then img_readonly and img_size are valid for that img_mounted. Save these values into a register if you will need them later.

These are block devices, so the way to read or write to them is to first specify the blk, and then specify a rd or wr and it will start to stream addresses through the sd_buff_addr.

There is no wait line, because it is assumed that you will be writing one block of data into DPRAM and it will write immediately.

To read data, set the sb_lba to the block number you want to seek to. Then set the correct bit in sd_rd to 1, and the HPS will respond by raising the correct bit in sd_ack, and keeping it high for the duration of the read. The HPS will then use sd_buff_addr and sd_buff_dout and sd_buff_wr to count from 0 to BLKSZ and send a byte per clock. After reaching the last byte, it will clear the sd_ack.

To write data, set the sb_lba to the block number you want to seek to. Then set the correct bit in sd_wr to 1. Similar to a read, it will raise the sd_ack bit, and count the sd_buff_addr - the core will respond by setting each sd_buff_din to the data it wants to write. After the block is written (sd_buff_addr reaches BLKSZ) sd_ack will be cleared.

When the user unmounts the image from the OSD, img_mounted will go high, and the img_size will be set to zero. That is how the core knows to unmount the current image.

- sd_lba[VDNUM] - logical block address - this is the block we want to start accessing
- sd_blk_cnt - number of blocks
- sd_rd - read number of blocks starting at address
- sd_wr - write number of blocks starting at address
- sd_ack - data read on the sd_buff_out is valid OR that write data must be valid on sd_buff_in[VDNUM]
- sd_buff_addr - byte address
- sd_buff_dout - data from disk
- sd_buff_din[VDNUM] - data to disk

- sd_buff_wr - Typically the read data from the blk interface is written into a FIFO or other DPRAM construct. This is a supplied control signal for the write signal to that FIFO

```
// SD config
output reg [VD:0] img_mounted,  // signaling that new image has been mounted
output reg        img_readonly, // mounted as read only. valid only for active bit in img_mounted
output reg [63:0] img_size,     // size of image in bytes. valid only for active bit in img_mounted


// SD block level access
input      [31:0] sd_lba[VDNUM],
input       [5:0] sd_blk_cnt[VDNUM], // number of blocks-1, total size ((sd_blk_cnt+1)*(1<<(BLKSZ+7))) must be <= 16384!
input      [VD:0] sd_rd,
input      [VD:0] sd_wr,
output reg [VD:0] sd_ack,

// SD byte level access. Signals for 2-PORT altsyncram.
output reg [AW:0] sd_buff_addr,
output reg [DW:0] sd_buff_dout,
input      [DW:0] sd_buff_din[VDNUM],
output reg        sd_buff_wr,
```

ROM and File loading, NVRAM saving

Boot ROMS are automatically loaded using ioctl lines. You can also have ROMS loaded via the F in the OSD, or via an MRA.

- ioctl_download - 1 when data is being downloaded.
- ioctl_index - [15:6] which file type [5:0] file number
- ioctl_wr - high when each byte is valid.
- ioctl_addr - address of byte from / to file - counts by two if set to wide
- ioctl_dout - data going to core from HPS (ROM)
- ioctl_din - data going to HPS from core - ie: to save NVRAM
- ioctl_upload - indicate there is an active upload
- ioctl_upload_req - set to 1 to ask the HPS to initiate an NVRAM save, for autosave, HPS only reads this when the OSD is open
- ioctl_rd - data is valid to read
- ioctl_file_ext - this is the file extension as a string
- ioctl_wait - set this flag to 1 if core isn't ready to process another byte from the HPS (flow control)

```
// ARM -> FPGA download
output reg        ioctl_download = 0, // signal indicating an active download
output reg [15:0] ioctl_index,        // menu index used to upload the file
output reg        ioctl_wr,
output reg [26:0] ioctl_addr,         // in WIDE mode address will be incremented by 2
output reg [DW:0] ioctl_dout,
output reg        ioctl_upload = 0,   // signal indicating an active upload
input             ioctl_upload_req,
input      [DW:0] ioctl_din,
output reg        ioctl_rd,
output reg [31:0] ioctl_file_ext,
input             ioctl_wait,
```

Note: boot.rom is sent via ioctl_index == 0, boot1.rom is sent with [5:0] set to 0, and [15:6] set to 1. boot2.rom.. etc

SDRAM board size

```
// [15]: 0 - unset, 1 - set. [1:0]: 0 - none, 1 - 32MB, 2 - 64MB, 3 - 128MB
// [14]: debug mode: [8]: 1 - phase up, 0 - phase down. [7:0]: amount of shift.
output reg [15:0] sdram_sz,
```

RTC

The RTC will pass the core the time from the HPS. If the HPS doesn't have the optional RTC board, it will pick up the time from an NTP server.

NOTE: the RTC just sends the time once at the beginning of the core start. After that, the core is responsible for incrementing the seconds, or incrementing the seconds and the other fields of the RTC structure.

in BCD:

- RTC[7:0] - seconds
- RTC[15:8] - minutes
- RTC[23:16] - hour
- RTC[31:24] - month day
- RTC[39:32] - month
- RTC[47:40] - year
- RTC[55:48] - week day

```
// RTC MSM6242B layout
output reg [64:0] RTC,

// Seconds since 1970-01-01 00:00:00
output reg [32:0] TIMESTAMP,
```

UART Flags

```
// UART flags
output reg  [7:0] uart_mode,
output reg [31:0] uart_speed,
```

Keyboard emulation

The hps_io module has two ways of accessing the keyboard and mouse. It provides ps2 compatible signals which are useful for porting cores that are expecting a ps/2 keyboard. It also provides an easier to use more reliable interface.

To use the new interface, ps2_key[10] is toggled with each keypress. ps2_key[9] is whether the key is pressed or not. And the rest of the bits are pretty standard ps2 bits with bit 8 being the extended bit.

Setting PS2 Keyboard lights

The HPS will use numlock and scrl_lock to indicate the mouse/joystick1/2 emulation. If the core wants to set any of these three of these LEDs it needs to set the bit in ps2_kbd_led_use - to enable it, and then in ps2_kb_led_status to turn the LED on/off. The LEDs are in the order:

```
assign ps2_kbd_led_status = {scrl_lock,num_lock,caps_lock};
```

this control is quite slow, so the core shouldn't use it unless it's pseudo-static indication.

```
// ps2 keyboard emulation
output        ps2_kbd_clk_out,
output        ps2_kbd_data_out,
input         ps2_kbd_clk_in,
input         ps2_kbd_data_in,

input  [2:0] ps2_kbd_led_status,
input  [2:0] ps2_kbd_led_use,

output        ps2_mouse_clk_out,
output        ps2_mouse_data_out,
input         ps2_mouse_clk_in,
input         ps2_mouse_data_in,

// ps2 alternative interface.

// [8] - extended, [9] - pressed, [10] - toggles with every press/release
output reg [10:0] ps2_key = 0,

// [24] - toggles with every event
output reg [24:0] ps2_mouse = 0,
output reg [15:0] ps2_mouse_ext = 0, // 15:8 - reserved(additional buttons), 7:0 - wheel movements
```

Gamma

```
inout  [21:0] gamma_bus,
```

Extension Bus

This should be used sparingly. It is for the exceptional case where a CD is needed as in the TurboGrafx16. It requires code on the HPS in the MiSTer binary.

```
// for core-specific extensions
inout  [35:0] EXT_BUS
```

sys - video_freak

Introduction

Video Freak is used to crop and scale the image.

Video Freak will take in the original VGA_DE_IN and output a new VGA_DE and VIDEO_ARX/VIDEO_ARY using the cropping parameters.

```
module video_freak
(
   input          CLK_VIDEO,
   input          CE_PIXEL,
   input          VGA_VS,
   input    [11:0] HDMI_WIDTH,
   input    [11:0] HDMI_HEIGHT,
   output         VGA_DE,
   output reg [12:0] VIDEO_ARX,
   output reg [12:0] VIDEO_ARY,

   input          VGA_DE_IN,
   input    [11:0] ARX,
   input    [11:0] ARY,
   input    [11:0] CROP_SIZE,
   input     [4:0] CROP_OFF, // -16...+15
   input     [2:0] SCALE    //0 - normal, 1 - V-integer, 2 - HV-Integer-, 3 - HV-Integer+, 4 - HV-Integer
);
```

sys - video_mixer

Introduction

Video Mixer is used as a helper to provide a lot of the nice videos that the MiSTer framework supplies. This includes scandoubling, gamma, hq2x scaling, and freezing.

Instantiating video_mixer

The module has a few parameters that are used to set it up.

- LINE_LENGTH -
- HALF_DEPTH -
- GAMMA -

```
module video_mixer
#(
    parameter LINE_LENGTH  = 768,
    parameter HALF_DEPTH   = 0,
    parameter GAMMA        = 0
)
(
    input          CLK_VIDEO, // should be multiple by (ce_pix*4)
    output reg     CE_PIXEL,  // output pixel clock enable

    input          ce_pix,    // input pixel clock or clock_enable

    input          scandoubler,
    input          hq2x,      // high quality 2x scaling

    inout    [21:0] gamma_bus,

    // color
    input [DWIDTH:0] R,
    input [DWIDTH:0] G,
    input [DWIDTH:0] B,

    // Positive pulses.
    input          HSync,
    input          VSync,
    input          HBlank,
    input          VBlank,

    // Freeze engine
    // HDMI: displays last frame
    // VGA:  black screen with HSync and VSync
    input          HDMI_FREEZE,
    output          freeze_sync,

    // video output signals
    output reg [7:0] VGA_R,
    output reg [7:0] VGA_G,
    output reg [7:0] VGA_B,
    output reg     VGA_VS,
    output reg     VGA_HS,
    output reg     VGA_DE
);
```

sys - arcade_video

Introduction

Arcade video is used mostly in arcade cores. It has some useful helpers to optionaly turn on scandoubler (based on the ini settings), deals with rotation on HDMI, and can optionally handle GAMMA, and different color widths.

Instantiating arcade_video

The module has a few parameters that are used to set it up.

- WIDTH - the width of one scanline
- DW - width of pixels. most arcades don't use a full 24 bits
    - 6 : 2R 2G 2B
    - 8 : 3R 3G 2B
    - 9 : 3R 3G 3B
    - 12 : 4R 4G 4B
    - 24 : 8R 8G 8B
- GAMMA - allow gamma controls

The clk_video needs to be 4x the ce_pix and above 40MHZ.

```
    input        clk_video,
    input        ce_pix,
```

The FX parameter uses the output of status[5:3] in this example of a config string: `"O35,Scandoubler Fx,None,HQ2x,CRT 25%,CRT 50%,CRT 75%;",`

Rotating vertical games

Optionally, we add the screen_rotate and pass in whether the game needs to rotate ccw or cw.

Screen rotate uses DDRAM for the framebuffer for rotation. If your core needs DDRAM for something else, see MCR3 for an example that uses DDRAM for an audio file and for the screen_rotate. Use this ddram: ddram instead of screen_rotate.

screen_rotate screen_rotate (.*, .rotate_ccw(ccw));

```
module arcade_video #(parameter WIDTH=320, DW=8, GAMMA=1)
(
    input        clk_video,
    input        ce_pix,

    input[DW-1:0] RGB_in,
    input        HBlank,
    input        VBlank,
    input        HSync,
    input        VSync,

    output       CLK_VIDEO,
    output       CE_PIXEL,
    output [7:0] VGA_R,
    output [7:0] VGA_G,
    output [7:0] VGA_B,
    output       VGA_HS,
    output       VGA_VS,
    output       VGA_DE,
    output [1:0] VGA_SL,

    input  [2:0] fx,
    input        forced_scandoubler,
    inout  [21:0] gamma_bus
);
```

Compiling for ARM

Cross Compiling for ARM

The ARM core on the DE-10 Nano is an ARM Cortex-A9 dual core You can grab a cross compiler for compiling ARM binaries on a 64bit intel based desktop system - eg OSX, Linux, or Windows (in the Linux subsystem)

Using a cross compiler on a Linux system

`wget -c https://developer.arm.com/-/media/Files/downloads/gnu-a/10.2-2020.11/binrel/gcc-arm-10.2-2020.11-x86_64-arm-none-linux-gnueabihf.tar.xz`

Unpack somewhere useful, eg /opt

`tar xf gcc-arm-10.2-2020.11-x86_64-arm-none-linux-gnueabihf.tar.xz`

and add to your path

`export PATH=$PATH:/opt/gcc-arm-10.2-2020.11-x86_64-arm-none-linux-gnueabihf/bin`

optionally set CC `export CC='/opt/gcc-arm-10.2-2020.11-x86_64-arm-none-linux-gnueabihf/bin/arm-none-linux-gnueabihf-gcc'`

then follow up with the usual make..

Using Docker

Docker has a arm7 cross compiler which is easy to install on Mac or Linux (assuming you have docker installed already!)

`docker run --rm dockcross/linux-armv7 > /usr/local/sbin/dockcross-linux-armv7`

`chmod +x /usr/local/sbin/dockcross-linux-armv7`

You can then cross-compile with

`/usr/local/sbin/dockcross-linux-armv7 make ./MakeFile`

or (to compile a fictitious hello.c -> hello.arm)

`/usr/local/sbin/dockcross-linux-armv7 bash -c '$CC hello.c -o hello.arm'`

Using msys on Windows 10 (Note that WSL is the recommended approach for compiling under Windows)

After installing msys, download the current 10.2 binary release from the ARM site e.g. `gcc-arm-10.2-2020.11-mingw-w64-i686-arm-none-linux-gnueabihf.tar.xz`
From this location: https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-a/downloads

Note: the 10.3 version has isssues; please do not use that version.

Place the tar.xz file into your /opt/ folder under msys (e.g. `C:\msys64\opt\`); to decompress it, do the following *from within MSYS*:

`cd /opt`

`unxz *.xz`

`tar xvf gcc-arm-10.2-2020.11-mingw-w64-i686-arm-none-linux-gnueabihf.tar`

Then, you will probably want to rename the "gcc-arm-10.2-2020.11-mingw-w64-i686-arm-none-linux-gnueabihf" folder to something shorter, like "arm".

Then when running MSYS set your PATH variable to point to it:

`export PATH=$PATH:/opt/<folder name from above>/bin`

Alternatives

Another, more complicated option for big projects targeting for ARM

Compiling the u-boot boot loader for MiSTer

There are few reasons why you would want to modify, build, and install the u-boot boot loader used by MiSTer. It is assumed anyone undertaking this is aware of the risks and prepared to deal with them.

Disclaimer: Although unlikely, misconfiguring the boot loader could lead to hardware damage.

Realistically, in the worst case, you will have to use a computer to reinstall the original u-boot image on the appropriate partition of your SD card.

Why would you want to build a custom u-boot? The most common scenario is that there is some hardware peripheral or feature of the Altera Cyclone V SoC FPGA that you seek to enable, possibly for experimentation or adding a custom hardware capability to your MiSTer. In short, the "why" is left to the reader, and the "how" is addressed below.

High-level steps for building a MiSTer-compatible u-boot

Building a working u-boot image seems to requires an ARM cross-compilation toolchain with GCC version 4 or 5 (earlier may work but was not tested). **Warning: In my experience, using a later version of GCC produces a u-boot image that is unable to reboot without a power cycle.**

1. Set up a cross-compilation environment as documented in the instructions for Compiling the Linux Kernel for MiSTer, but do not download or install the ARM toolchain.
2. I suggest downloading and installing the Linaro 5.5-2017.10 toolchain, as it is the latest version that I found works properly. The current stock u-boot image distributed with MiSTer was compiled with GCC 4.8, so Linaro 4.9-2017.01 is another reasonable choice and still available for download as a binary at the time of this writing.

```
wget -c https://releases.linaro.org/components/toolchain/binaries/5.5-2017.10/arm-linux-gnueabihf/gcc-linaro-5.5.0-2017.10-x86_64_arm-linux-gnueabihf.tar.xz
```

3. Continue with the steps for installing the toolchain detailed in Compiling the Linux Kernel for MiSTer, substituting "5.5-2017.10" for the Linaro version.
4. The u-boot source is available in the https://github.com/MiSTer-devel/u-boot_MiSTer repository (--depth=1 clones only the recent version, slimming down the download; omit this if you would like the full commit history):

```
git clone --depth=1 https://github.com/MiSTer-devel/u-boot_MiSTer
```

5. Compile the source:

```
cd u-boot_MiSTer
make MiSTer_defconfig
make -j6
```

6. Copy the u-boot image over to your MiSTer device. Note that the file must be copied to `linux/uboot.img` on your SD card. I suggest making a backup copy of the original `linux/uboot.img`. For example, if you are using `scp`:

```
scp root@[IP address of your MiSTer]:/media/fat/linux/uboot.img uboot.img.orig
scp u-boot-with-spl.sfp root@[IP address of your MiSTer]:/media/fat/linux/uboot.img
```

7. Install the u-boot image on the appropriate partition of your SD card:

```
ssh root@[IP address of your MiSTer]
cd /media/fat/linux
./updateboot
```

8. Power cycle your MiSTer (as opposed to a warm reboot).

Analog Simulation

What is the problem?

Many old computers have analog parts, for example audio boards that have partly analog synthesis and old desktop computers with tape input.

Currently there is no convenient way to directly simulate electronic circuits, for example based in their spice netlists, in real time on the MiSTer. So to implement these type of circuits in fpga, you will have to come up with some kind of simulation of the behavior of the electronic circuit.

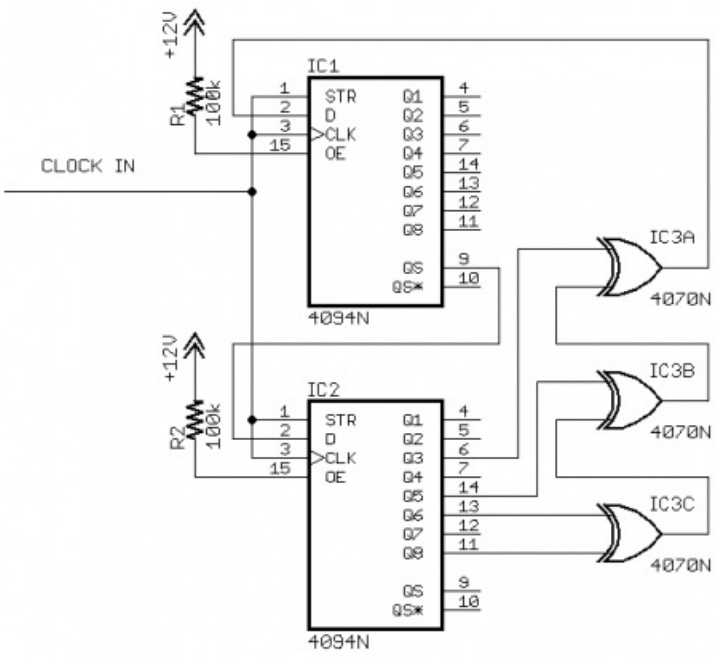
The Arcade-Battlezone core is an example of a core with analog sound synthesis.

A way to approach the problem:

1. Find youtube video's of people playing the actual game, to get some idea of the sounds it has.
2. Play the game in an emulator, paying extra attention to sounds that do not sound the same in the emulator as in the video of the actual machine
3. Identify the digital and analog parts of the schematic.

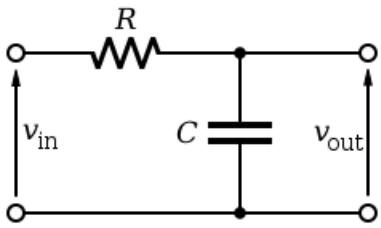
   One thing to keep in mind is that the digital parts run at the system clock speed, or sometimes a separate clock source that is closer to the audible frequency spectrum, in Arcade Battlezone there is a 12khz input to a digital noise circuit, of which the output goes into and analog integrator circuit. This creates an axplosion sound. The implementation of this sound is listed at "inverting amplifier" below.

   The analog parts will be outputting at audio sample rate, i.e. 48khz
4. Implement all the digital parts, common digital components in sound systems are:
   - linear feedback shift register:



   An implementation of a similar circuit in arcade battlezone can be found here:

   - flipflops, like the jk flipflop, implemented here:
5. Identify analog circuits with isolated behavior, i.e. 1 input, 1 output. These can be individually implemented and tested.
6. Identify common, easily recognisable and implementable parts, such as:
   - low pass filters



This can be implemented using an iir low pass filter, you can find the parameters using this spreadsheet

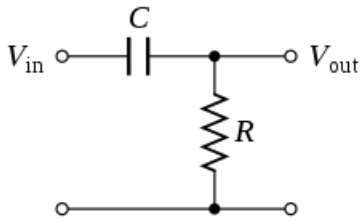
Some implementations of iir filters:

- https://github.com/MiSTer-devel/Arcade-Blockade_MiSTer/blob/main/rtl/audio/blockade_lpf.v
- https://github.com/MiSTer-devel/Arcade-Blockade_MiSTer/blob/main/rtl/audio/iir_1st_order.v

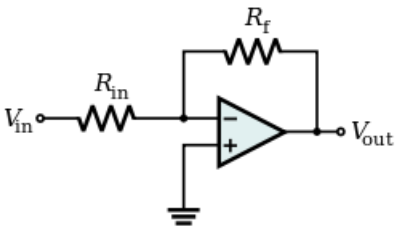
A simpler type:

- https://github.com/jopdorp/Arcade-BattleZone_MiSTer/blob/sound/rtl/iir.sv

- high pass filters

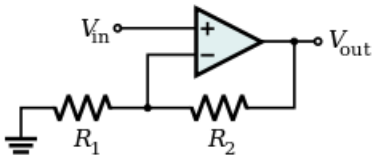


- inverting amplifiers



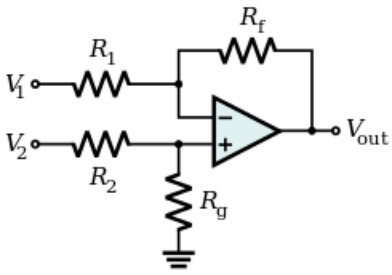
This is essentially a sign inversion of the sample, followed by a (fixed sign) multiplication.

- non inverting amplifiers

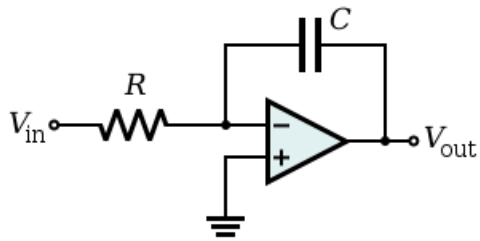


This is just a (fixed sign) multiplication of the sample

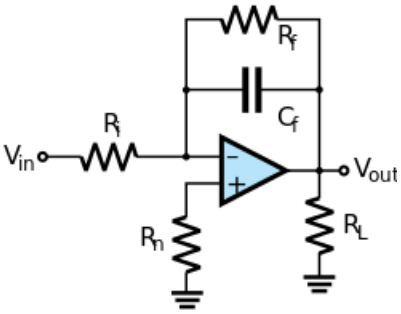
- differential amplifiers



- inverting integrators



or

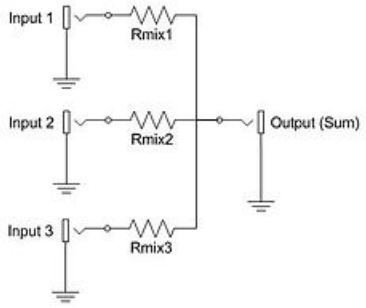


This is essentially a sign inversion of the sample, followed by a multiplication, with the result being stored in a reg.

The multiplication is run repeatedly, each audio clock cycle, Resulting in a "release/decay" type amplifier envelope.

An example of an implementation of this can be found here

- other common opamp circuits
- additive mixers



An additive mixer just adds up/averages two signals. If the resistance is higher for one signal than the other, this will result in a volume difference between them.

7. All the parts that are going to need more attention, make note of what these parts are an try to guess what they are for. Differentiators and integrators combined with 555 timers and noise inputs can be tricky.
8. Implement the mixers
9. Implement the filters
10. Implement the difficult parts of the circuit:
    1. Implement the circuit in a simlator.
    2. Figure out what input goes into the circuit when the game gets played.
    3. Run the simulation with the correct input and save the output as a wav file
    4. Analyse the output by looking at it in a wave editor like audacity and listening to it.
    5. Is the input always the same? (not noise as input) Does the circuit always response in the same way? (not generating noise)
        1. Sample the output of the simulation
        2. Convert the sample into an array literal and trigger it as needed, usually the sample will just be triggered by a pulse in this case.

6. If the analog circuit has noise as input or has variable inputs such as a number controlled frequency or speed you will need to:
    1. make a mathematical model of the analog circuit.
    2. implement the model in an easy to use programming language such as python
    3. compare the output of the mathematical model with the output of the circuit simulation using the same inputs. They have to be close, but not identical, but they should be identical to the hearing.
    4. rework your mathematical model to not have to use any dividers in real time, this is done in two steps
        1. by algebra
        2. by precalculating any left-over divisions into division lookup-tables
    5. you can also create other kind of lookup tables, for example a sine wave.
    6. implement the mathematical model in HDL
    7. verify the outputs of the HDL with a tesbench
    8. hook up the chip to the rest of our core!

Handy links:

Developer journeys of analog circuit implementations

MiSTer - Head On

Informational resources

- For finding the right values of a filter: https://www.micromodeler.com/dsp/
- Common operational amplifier uses, such as differentiators and integrators: https://en.wikipedia.org/wiki/Operational_amplifier_applications
- Explanation of how 555 timers work: https://www.electronics-tutorials.ws/waveforms/555_oscillator.html
- Spice programming https://www.allaboutcircuits.com/textbook/reference/chpt-7/fundamentals-spice-programming/

Code examples

- example of a noise-based sound python implementation in a notebook: https://github.com/jopdorp/Arcade-BattleZone_MiSTer/blob/sound/spice/Red%20baron%20crash%20circuit%20sim.ipynb
- example of a piece of python code that generates a lookup table: https://github.com/jopdorp/Arcade-BattleZone_MiSTer/blob/sound/rtl/generate_control_coltages_to_frequency.py
- example of a common digital noise generator: https://github.com/jopdorp/Arcade-BattleZone_MiSTer/blob/sound/rtl/noise_source_shell_explo.sv
- example of a sine-based sound that takes noise as input: https://github.com/jopdorp/Arcade-BattleZone_MiSTer/blob/sound/rtl/bang_sound.sv
- example of a script that can take the output of a spice simulation and generate a wav file from it: https://github.com/jopdorp/Arcade-BattleZone_MiSTer/blob/sound/spice/spicetowav.py
- example of a spice netlist for a noise based sound: https://github.com/jopdorp/Arcade-BattleZone_MiSTer/blob/sound/spice/bang.cir

Tooling

- https://www.falstad.com/circuit/
- http://ngspice.sourceforge.net/
- https://www.audacityteam.org/