Batocera native CRT output

This guide only applies to Batocera **v32** on the x86_64 and RPi builds. For **v30**/**v31** and on x86, refer to the legacy guide. This article is also not for the faint of heart. Word-count: about 9k.

An automated script has been developed which can be used to configure and activate CRT capabilities in x86_64 Batocera versions between **v32** and **v36**. Use it after reading through this guide at least once to understand its usage.

Batocera has the capability to natively output an analogue signal, should you have the hardware to do so. The specifics are later in this guide, but that basically means you have a VGA/DVI-I port on your graphics card (which is becoming exceedingly rare with modern graphics cards) or on your motherboard (the CPU will need integrated graphics to support it, probably along with particular BIOS settings). Modern cards with only digital outputs have a minimum bandwidth that's above the acceptable rate that CRTs can interpret, and thus won't be compatible for use in this guide.

You can however use an active converter (glossary below) to convert a digital signal to an analogue one (this will not be native) to use such digital-only graphics cards with a CRT display, however this guide is solely for running the pure analogue signal from the card to your display and the configurations made in this guide will not be compatible with such converters.

Recommended watching:

- Displaced Gamers' excellent video on the history of 240p - This explains the background of the “240p” signal common with early generation consoles and why “scanlines” were a thing.

Recommended reading:

- RetroRGB's RGB guide - This focuses more on the aspect of connecting your physical analogue consoles to modern displays, but it covers much of the terminology used throughout this guide and explains particular details about analogue standards.
- buttersoft's AussieArcade forum post: A guide to connecting your Windows PC to an SD CRT TV, PVM or Arcade Monitor - This focuses more on connecting Windows computers to CRT displays, but it goes over a lot of the same things we will have to later on. This is a very technical document, more so than this guide is.

Foreword

It should go without saying, but unlike digital signals, analogue signals and the devices that receive them tend not to have safety limitations on what they are capable of. This can result in breaking displays, blowing old fuses, overheating ancient capacitors, making things catch fire, etc. You should perform adequate research before attempting any of this with your old equipment. Although this guide has safety precautions built into it, the safety of you and your equipment is solely your own responsibility.

This guide would not have been possible without the following people to name a few:

- jfroco's work to output Batocera on a CRTs.
- rtissera's knowledge, enthusiasm and willingness to add 15 KHz patches.
- Calamity for his knowledge, drivers, tools, GroovyMame, Switchres, etc.
- Substring's work on GroovyArcade, SDL, KMS, etc.
- D0023R Doozer's continued work at adding 15 KHz support to the Linux kernel.
- dmanlcf's work on keeping the 15 Khz patches up to date for Batocera.
- Rion and Atari for compiling this guide.

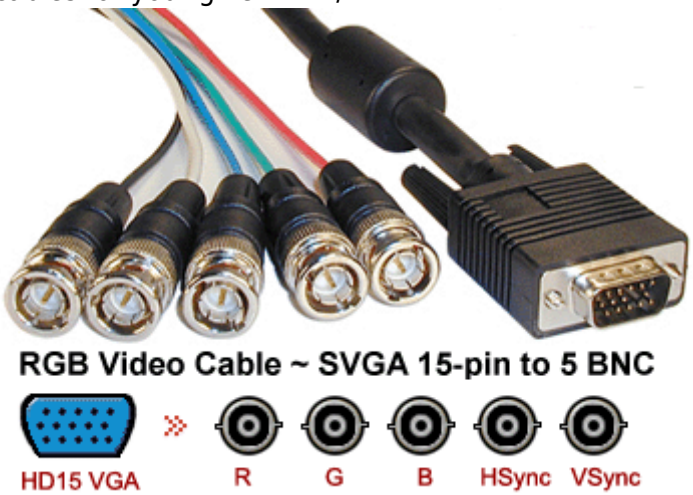
Glossary

The field of analog signals and CRT-related standards is very complex and occasionally misleading. We won't dive into every aspect in this guide but we will explain some relevant terms here (sorted in alphabetical order) to avoid confusion. Feel free to come back to this section at any time throughout this guide:

- **Active converters** Little boxes that convert one signal to another. These use tiny computers with processors and RAM to process the image digitally and send it out the other side as appropriate. Generally undesirable, as they can introduce lag and other artifacts. This is not considered “native”. This guide assumes that you are **NOT** using one of these (as they try to handle the entire conversion themselves, removing Batocera's control over the signal).

- **BNC cables** These are standard aerial coaxial cords that can carry any analogue signal. For our purposes, they will be carrying an RGB video signal. There were not many consumer TVs that

utilized this type of connection, but they were common with professional/broadcast reference monitors (PVM/BVM for short). You'll usually need to find a specific passive adapter that has the correct *amount* of BNC cables for your given PVM/BVM.

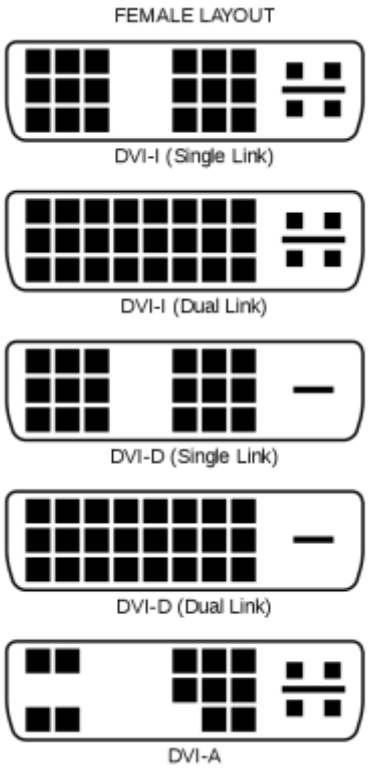


- **Component video cables** The three red blue green cable connection. Not to be confused with the similarly spelled composite cable, as it uses three cables instead of one. Not to be confused with RGB signal, as this uses a YPbPr signal. Them being colored red blue and green is purely coincidental and not related to RGB at all.

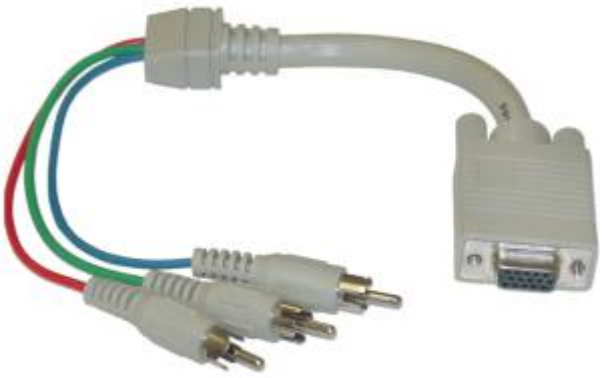
- **Composite signal** An *analogue* signal. The act of sending both luminance and color channels down a single wire. Utilized by the composite video cable.
- **Composite video cable** The yellow cable. That's enough, you know what I'm talking about already.

- **DisplayPort** The modern digital video connection port, common on modern graphics cards. Compatible with both DP and HDMI signals. This port cannot be used to send a native analogue signals so it is not recommended for use with CRT, but this can be worked around if you have the right equipment as explained further on in the guide.
- **DVI-I port** This is the 29 pin port in your graphics card, capable of analogue signals. You will typically find this port alongside the VGA port. Newer cards no longer include this port. It is of extreme importance to not confuse this with the identical-looking DVI-**D** port, which is only capable of digital signals. The DVI-I port has additional pins above and below the flat tongue,

whereas DVI-D doesn't. DVI-A would also work, but that is exceptionally rare.



- **EDID** Extended Display Identification Data (EDID) is a *digital* metadata signal used by display devices to describe their capabilities to a video source (in this case, your Batocera computer). Modern digital displays all have this, and Batocera uses that to automatically switch to a compatible resolution. Certain VGA-compatible CRT monitors also use this, but not all of them. This standard was introduced in 1994, so monitors manufactured before then do **not** have EDID.
- **Horizontal scan rate** Usually expressed in kilohertz. This is the number of times per second that a horizontal line is drawn on a display, as opposed to vertical refresh rate which is the number of times per second that an entire screen of image data is displayed.
- **Passive adapter/cables** These are simple cables that adapt one interface to another. They don't involve any circuitry, all the work is done by the machine sending out the signal. Passive cables can only go analogue-to-analogue or digital-to-digital, not analogue-to-digital or digital-to-analogue.

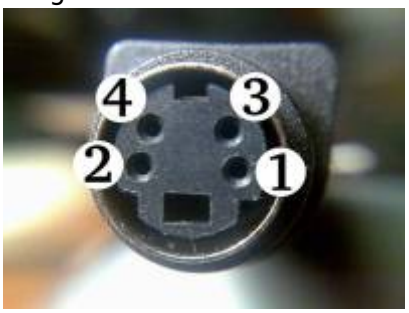


- **RGB video signal** In this context, refers to the *analogue* signal that has an individual channel

for each Red, Blue and Green color signal. RGB is not compatible with component/composite/S-video by default, and a transcoder must be used to switch between the signals, but they otherwise are all analogue signals.

- **SCART** This is the 21 pin port that superseded the standard composite video output. Popular in Europe and Japan, but not many other places. Carries both analog video and audio signals. Natively uses an RGB video signal, but is backwards compatible with YPbPr, composite and S-video signals.

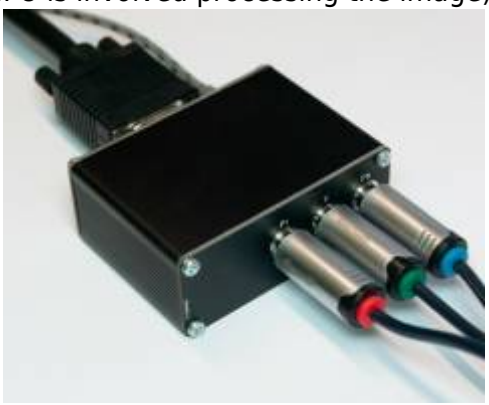
- **S-video port** A slightly better version of composite video. This separated the color and luminance signals into two separate wires each, reducing color bleed and artifacts, but in practice the difference wasn't that significant. Common with DVD players, but not much else.



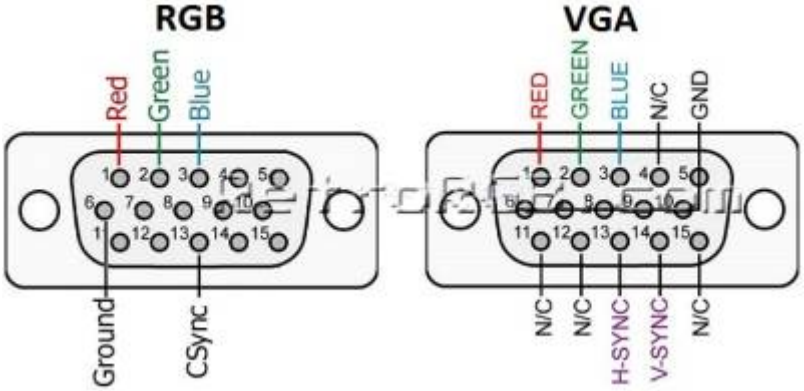
- **S-video signal** An *analogue* signal. Essentially the exact same signal as a composite signal, but the color and luminance signals are separated. Trivial to switch between this and composite hardware-wise, but ultimately the device needs to understand what type of signal it's receiving to properly display it.
- **RF modulator** This is a device that can translate composite, component or RGB analogue signals into a radio frequency. This is required for CRT displays that only have an RF input. VCRs usually had one of these built-in to them, and make for a fine substitute.

- **Three-pole A/V composite cable** A more modern analogue signal cable which violates standards by running a video signal through a 3.5mm headphone jack splitting into RCA cables. This is specifically designed to be used in the Raspberry Pi in its composite video modes, and will not be compatible with other devices/computers. Is only capable of carrying a composite signal, the other plugs are strictly for audio.

- **Transcoder** A device that can convert an RGB signal into an inferior signal like component/composite/S-video or back. Usually requires its own power source to function, but isn't an active converter (no CPU is involved processing the image).



- **Vertical refresh rate** The number you're probably more familiar with, typically measured in just hertz. This is the number of full frames drawn per second. Typically this is 60 Hz for all modern displays, but older CRTs were split between 60 Hz and 50 Hz depending on where you lived.
- **VGA port** This is the 15 pin port ubiquitous with old computer video output and classroom projectors. It is important to note the distinction between a VGA signal and the DSUB connector used to carry that signal, as not all DSUB connectors actually send a VGA signal. Refer to this console-related article on RetroRGB about it for more info. Essentially, a DSUB port is capable of sending out either an RGB or VGA signal, given the right settings.



- **VGA signal** A unique hybrid *digital/analogue* signal. This became the standard for computer monitors in the late 80s/early 90s, and became **the** video standard for most devices. Although this signal has slowly begun being phased out, it's still compatible with pretty much every display ever created since its inception (if you can get a port hooked up to the display's interface that is). While the VGA signal itself is analogue, it was designed specifically for computer monitors and relies on digital signalling features absent from most consumer TVs. As such, the signal cannot be transcoded into other analogue video signals without highly specialized hardware. This is outside of the scope of this article, but assuming you can get it all working then you can still apply the later sections of this guide like ES scaling. Unlike other

analogue signals, VGA utilizes an EDID to notify the machine of its resolution capabilities, so therefore it cannot be configured in the same manner as described in this guide; instead use the general method for changing resolution.

- **YPbPr video signal** A lower-bandwidth form of RGB, which has both the green and luminance channels on one wire. Used by component video cables, but can also be sent through a SCART connection.

Prerequisites

To achieve a native analogue signal on your setup, the following elements are needed:

- A network connection (wired is preferred, as it needs no configuration)
- For x86_64:
  - A graphics card or integrated graphics with RGB analogue output
    - **AMD**/**ATI** graphics cards with **VGA** or **DVI-I** ports (preferred, **Radeon HD 3000** series and up to **Radeon R5**/**R7**/**R9 300**), these cards are:
      - **Radeon RX** up to the **300** series (excluding the **R9 390**) (**R9 380X** is the newest and most powerful AMD card supporting analogue)
      - **Workstation** up to the **FirePro W5000**
    - **Nvidia** graphics cards with **VGA** or **DVI-I** ports (major flaws and limited to Super Resolutions, cards made before 2008 may not have these limitations), these cards are:
      - **GTX** up to the **900** series cards (**980** is the recommended card to use)

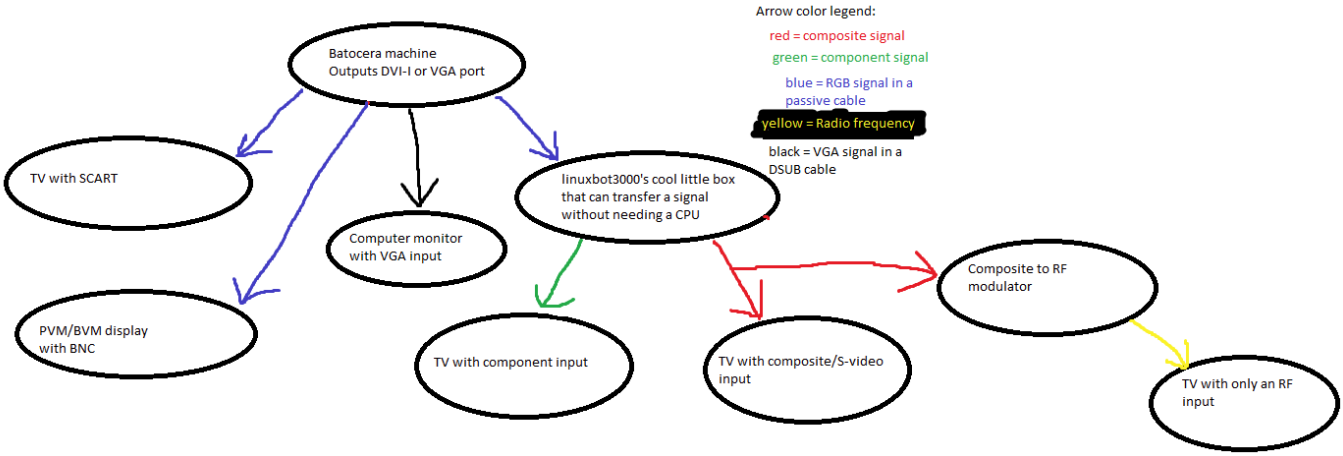
        Cards older than the **GTX 600** series may not be supported by the official Nvidia drivers, though performance with 5th gen systems and below is adequate with the nouveau drivers.

      - **Titan** cards up to the **Titan X** (but excluding **Titan X Pascal**) (**Titan X** is the most powerful card supporting analogue)
      - **Quadro** cards up to the **Quadro K5200**
    - **Nvidia** graphics card with **DisplayPort** (no native analog signal, requires disabling low-level driver limitations which are potentially dangerous, requires additional ratnest of cables):
      - If opting to go this route, an active converter or the CableDeconn DP to VGA adapter will be required. In general, this is not recommended as it is not a native analogue signal.
      - A transcoder will still be required.
    - **AMD** APUs with **VGA** or **DVI-I** ports on the motherboard (hit or miss, highly dependent on hardware limitations, older chips have a higher chance of working)
    - **Intel** integrated graphics with **VGA** or **DVI-I** ports on the motherboard (has some flaws and limitations, older chips have a higher chance of working)
  - If using a TV/PVM/BVM with **SCART**/**BNC** inputs, the appropriate **DVI-I**/**VGA** RGB to **SCART**/**BNC** RGB passive adapter/cable
  - If using a computer monitor with **VGA** input, simply a **VGA** cable
  - If using a TV with **component**/**composite**/**S-video** inputs, an appropriate **DVI-I**/**VGA** RGB to **component**/**composite**/**S-video** transcoder
    - If using a TV that only has **RF** input, you will need a **composite** to **RF** modulator in

addition to the transcoder
- For Raspberry Pi:
  - A Raspberry Pi with **composite** video output (the headphone jack)
  - The **three-pole A/V composite** cable for Raspberry Pi
  - A TV with **composite** inputs, *or*;
  - If using a TV that only has **RF** input, you will need a **composite** to **RF** modulator in addition to the transcoder
- A way to SSH into the Batocera machine (for Windows, PuTTY works fine)
- A way to edit files over the network (for Windows, WinScp and Notepad++ work fine)

It is highly recommended using an external editor like Notepad++ and WinScp for editing, as they are easier to use than command line tools and respect Unix line terminators (unlike Windows Notepad).

If any of that was confusing here's a professional-grade flowchart explaining what is required for each type of display (on PC):



Component Transcoders

Some appropriate VGA-to-component adapters:

RGB VGA to YPbPr Component transcoder/converter by linuxbot3000

- https://www.ebay.com/itm/174166352619
- homepage

GreenAntz RGB to component transcoder

- http://forum.arcadecontrols.com/index.php/topic,164245.0.html

Retrotink RGB2COMP (Scart to Component)

- https://www.retrotink.com/product-page/rgb2comp

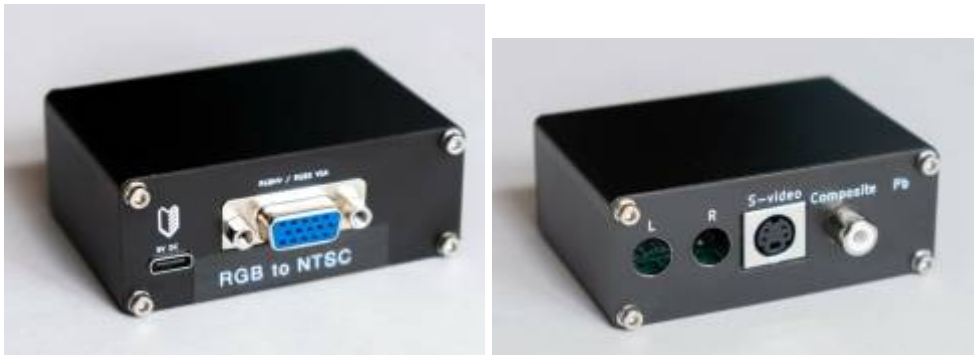
RGBS VGA SCART to YPBPR Component Transcoder

- https://www.aliexpress.com/item/502689543.html

Composite/S-Video Transcoders

If you need composite/S-video/RF, use these transcoders:

- RGB VGA to NTSC S-video and composite transcoder/encoder by linuxbot3000 (ebay) (homepage)



RGB to Composite & S-Video Ver2.0

(https://www.axunworks.com/product-p341706.html)

A/V composite cable (Raspberry Pi)

If you need the three-pole A/V composite cable:

- Link to Adafruit's store page for the cable
- Link to Amazon's store page for the cable

An Xbox 360E three-pole to A/V composite cable also works for the Raspberry Pi.

DisplayPort to VGA DAC

If you only have a DP connection, use the CableDeconn DP to VGA adapter before sending the signal to the transcoder of your choice:

- Link to official website
- Link to Amazon's store page for the adapter

If you are in a situation where you must use an active adapter, the CableDeconn DP to VGA adapter is one of the best options. It uses a repurposed Realtek RTD 2168 audio chip for its conversion, which doesn't rely on an oscillating crystal (no out-of-sync timing it has to account for) and no automatic deinterlacer (pretty much standard for all other active adapters, this will ruin your signal's quality).

USB-C to VGA DAC

If you only have a USB-C connection, use the StartTech.com USB type C to VGA, 3m, CDP2VGA3MBNL (Note: this 3m cable has the RTD2166 chip. Shorter versions of the same brand use a different chip that may not be compatible) before sending the signal to the transcoder of your choice:

- Link to official website
- Link to Amazon's store page for the adapter

If you are in a situation where you must use an active adapter and only have a USB type C connection the StartTech.com USB type C to VGA adapter is one of the best options. It uses a repurposed Realtek RTD 2166 audio chip for its conversion, which doesn't rely on an oscillating crystal (no out-of-sync timing it has to account for) and no automatic deinterlacer (pretty much standard for all other active adapters, this will ruin your signal's quality).

Using a Raspberry Pi

Friendly reminder that this guide was written for **v32**, this function may not work identically on future versions, if at all.

This is way easier than on PC, as all the necessary modifications can be done on the SD card before even turning on the Pi.

If using Windows, remember to use Notepad++ to edit files so as not to corrupt them. The built-in Notepad will corrupt them.

After Batocera has been flashed to the SD card, open up the boot partition on it labelled “BATOCERA”

and open the `cmdline.txt` file. Add the following to the beginning of the line:

```
video=Composite-1:720x480@60ie
```

Save the file (make sure not to accidentally add extra lines/spaces as it may interfer with Batocera booting). Then in that same folder, open `config.txt`. Comment out any line that contains `hdmi` by adding a # in front of it.

```
#hdmi-yadda-yadda
```

In the main section (not inside any section like `[RPI3]`), add the following lines:

```
max_framebuffer_width=320
max_framebuffer_height=240
framebuffer_width=320
framebuffer_height=240
enable_tvout=1
sdtv_mode=0
sdtv_aspect=1
audio_pwm_mode=2
disable_fw_kms_setup=1
enable_uart=1
```

Then the Raspberry Pi is set up to output to a CRT display via the composite output. Read the editing files section, and then skip forward to the section about EmulationStation.

Although most of the rest of the guide also applies to RPi, RPi does not support switchres or standalone MAME. Hence, any configuration that applies to them would not be applicable to RPi. RetroArch does have its own configuration though.

If after doing this on a Pi 4, you get the menu working but no emulators launch correctly, try using the FKMS driver instead of the real KMS driver.

Under the `[pi4]` header, comment out `dtoverlay=vc4-kms-v3d-pi4` and uncomment `dtoverlay=vc4-fkms-v3d` to do so.

If after doing this on a Pi 3, you get a black screen after the boot logo, try editing this line under the `[pi3]` header:

```
dtoverlay=vc4-kms-v3d
```

so that it is instead:

```
dtoverlay=vc4-kms-v3d,composite=1
```

Connecting the PC to a CRT display

The first step is to safely connect your Batocera computer to the CRT display itself. If using a VGA connection with a EDID CRT monitor, it is as simple as connecting the display, turning it on and then turning the Batocera machine on. Keep reading this section for all other displays. If your CRT monitor does not send an EDID (the standard was introduced in 1994, so monitors manufactured before then **do not have EDID**), then you should still follow the following instructions just to be safe.

During the boot process and resizing of the partition it will boot up in a non-supported resolution. Keep the CRT TV off for the moment or on another AV channel so it doesn't receive **dangerous signals**. **These signals can destroy your CRT TV.**

Also keep in mind that during the BIOS boot process the same rule applies. Have the CRT TV off or on another input when first booting up.

To solve this we have 4 options:

- Do the aforementioned, leaving the CRT TV off or on another channel.
- Have a look at gambaman's excellent solution The ultimate VGA to SCART adapter over at Build Your Own Arcade Controls Forum (BYOAC).
- Use buttersoft's passthrough dongle based on gambaman's design in the link above.
- Flash you AMD/ATI card with ATOM-15.

Once everything is safe, you can turn on your Batocera machine. If you'd like to edit files using the Batocera machine itself, you can hook up a secondary digital display in the meantime. If you have multiple ports and use another port, then you'll not be able to do this once you disable the other port outputs in the upcoming section. If you have a modern digital display that can handle older standard signals over older standards cables without blowing up (such as a LCD monitor with both HDMI and VGA ports), you can use that in place of your CRT until everything is configured correctly for it.

Editing the configuration files

How to edit system configuration files in Batocera

Batocera is unlike other typical Linux-based distributions as it contains most system-critical configuration files (such as files in the `/etc/` or `/usr/` folders) in its virtual filesystem, read this for more information. This filesystem is expanded from the `/boot/batocera` firmware on boot into RAM, so any changes made to it by default will be forgotten on shutdown. In order to make these changes permanent, we must save the filesystem overlay (this will be mentioned in this guide whenever it is required). Overlay files are removed when updating Batocera, so they will need to be re-created again after every update (overlay files are **not** portable between different versions of Batocera).

Batocera does not share its virtual filesystem over the network share and nor is it accessible by attempting to access the drive physically from another operating system. In order to edit these files

you must use Batocera itself while it is running by either utilizing its included file manager (press `[F1]` on the systems list) or by utilizing command line tools via SSH/xterm.

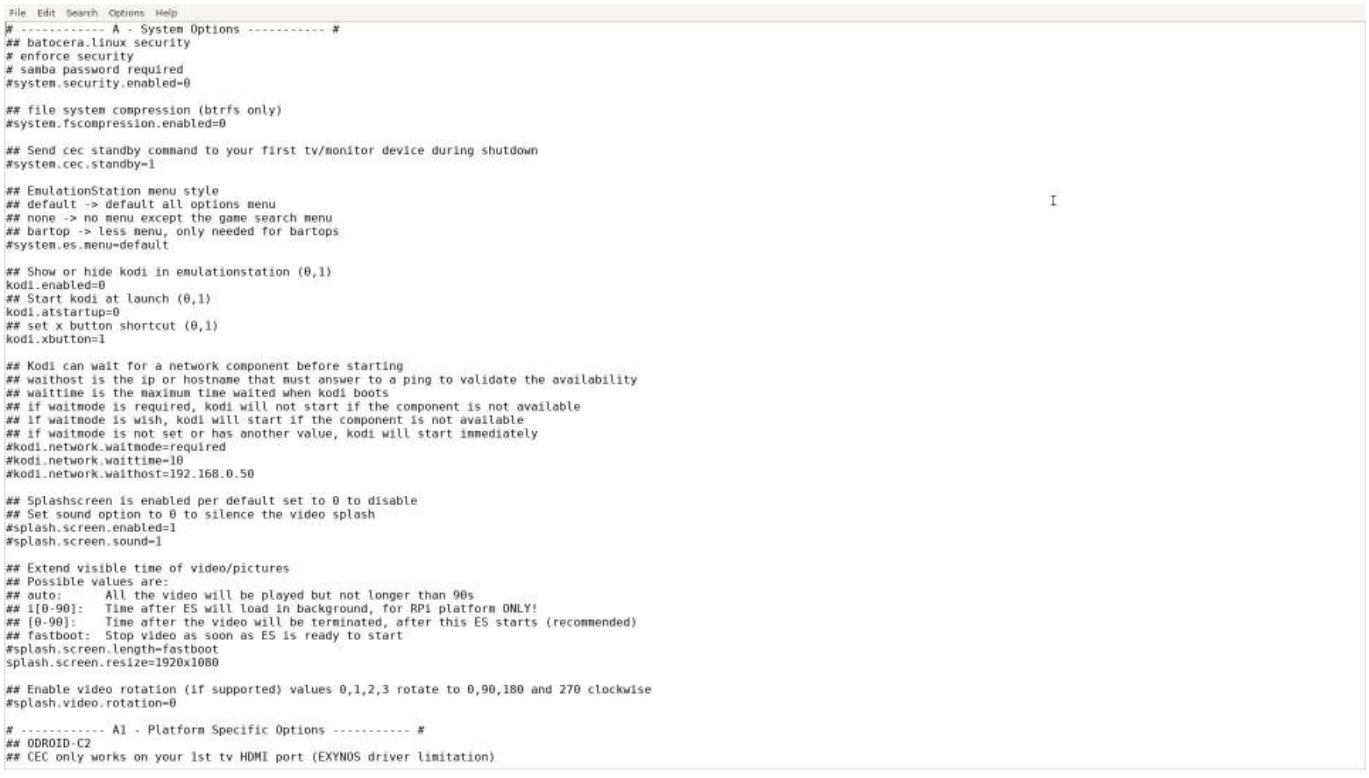
The two exceptions to this are files stored in the `/boot/` partition (the one readable by other systems) and the `/userdata/` partition (the default location of the share folder).

Using the file manager to edit files

The easiest way to manage configuration files is via the built-in file manager (x86/x86_64 only). Press `[F1]` on the system select screen to bring up the file manager window:

Click on the “Share” folder on the left sidebar, and then click the “Up” arrow near the top to get to the root of the whole system:

You can open text files here by just double-clicking them, a basic text editing application will open them:



You can use ordinary system shortcuts like `[Ctrl]+[C]` to copy, `[Ctrl]+[V]` to paste, `[Alt]+[F4]` to close an application and `[Alt]+[Tab]` to switch focus between applications.

Using nano to edit files via SSH

This is a convenient option if you don't have a second display to use the file manager with. Here's a

crash course in using the nano command line text editor:

- Type `nano /path/to/my/file.txt` to open that file (it will show you a blank document if the file doesn't exist).
- Nano operates like a standard text editor, start typing to fill in its contents. If in SSH, you can paste the contents of your clipboard by right-clicking the terminal screen (it doesn't matter where).
- All functions are tied to `[Ctrl]`/`[Alt]` key shortcuts. The only ones you need to know are `[Ctrl]+[S]` to save the file, and `[Ctrl]+[X]` to exit.

Using WinSCP and Notepad++

You can find a mini-tutorial for WinSCP on its respective page.

Change the main resolution on PC

If using a VGA signal display with EDID, this step is not necessary. If using a VGA signal display with**out** EDID, you should use `640x480eS` (note the lack of the “i”) instead of `640x480ieS`.

If using a CRT TV, the `syslinux.cfg` file will need to be changed to be able to use Batocera at a lower resolution. To achieve this, we need to modify the boot partition:

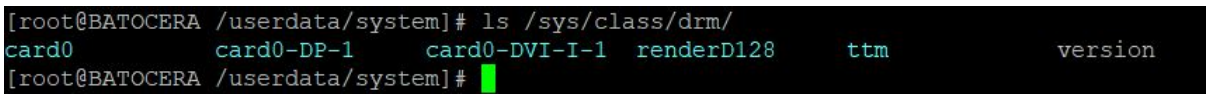
1. Connect to Batocera using SSH. See Access Batocera Linux via ssh for more information.
2. Make the boot partition writable. In the SSH session, run `mount -o remount,rw /boot`.

Then, we have to identify the video output we will be using:

1. Get your graphics card's analog DVI/VGA output (**without** the `card#` string) using the following command: `ls /sys/class/drm/`
   - This will return something like

     ```
     card0          card0-DP-1     card0-DVI-I-1  renderD128     ttm
     version
     ```

     Here's a screenshot of it:

     In this example our card output is **DVI-I-1**
2. Search for the `syslinux` file to enable booting in low resolution

To make the next step a little bit easier for Windows users, connect to Batocera using WinScp and install Notepad++

    - If legacy, the file will be at `/boot/boot/syslinux/syslinux.cfg`
    - If UEFI, the file will be at `/boot/EFI/BOOT/syslinux.cfg`
3. Append a space, followed by `video=`*`[your-card-output]`*`:640x480ieS` to the APPEND line in the file (make sure there are no additional spaces after!)

In our example:

```
    APPEND label=BATOCERA console=tty3 quiet loglevel=0
vt.global_cursor_default=0 mitigations=off
```

would become

```
    APPEND label=BATOCERA console=tty3 quiet loglevel=0
vt.global_cursor_default=0 mitigations=off video=DVI-I-1:640x480ieS
```

For other supported boot resolutions, see this documentation on Github. Here is an example `syslinux.cfg` file:

Click to expand

syslinux.cfg

```
UI menu.c32

TIMEOUT 50
TOTALTIMEOUT 300

SAY Booting Batocera.linux...

MENU CLEAR
MENU TITLE Batocera.linux
MENU SHIFTKEY

LABEL batocera
    MENU LABEL Batocera.linux (^normal)
    MENU DEFAULT
    LINUX /boot/linux
    APPEND label=BATOCERA console=tty3 quiet loglevel=0
vt.global_cursor_default=0 mitigations=off video=DVI-
I-1:640x480ieS
    INITRD /boot/initrd.gz

LABEL verbose
    MENU LABEL Batocera.linux (^verbose)
    LINUX /boot/linux
    APPEND label=BATOCERA vt.global_cursor_default=0
    INITRD /boot/initrd.gz
```

Don't worry if your CRT display can render at a higher resolution, we'll be fine-tuning this for EmulationStation later. We just need to *not* send a signal that will destroy your CRT during the boot procedure.

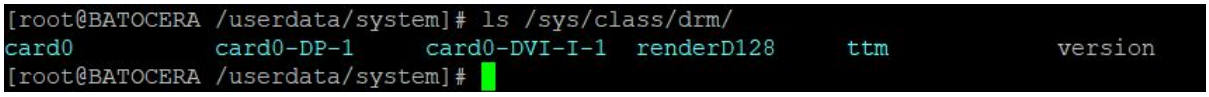
Disabling the other video output on PC

It is highly recommended to disable the digital exclusive outputs (HDMI/DisplayPort (DP) port) to get video exclusively out of the analog output (DVI-I/VGA port). This is not just for performance, but to avoid conflicts with Switchres later on in this guide. To do so:

1. Get all the outputs on your graphics card that are capable of digital output we want to disable (**with** the `card#` string this time) using the `ls /sys/class/drm/` command (eg. card0-DP-1).
    - This will return something like:

      ```
      card0          card0-DP-1     card0-DVI-I-1  renderD128     ttm
      version
      ```

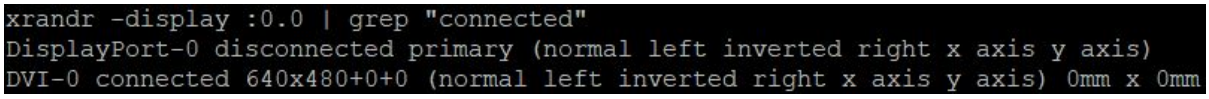
      Here's a screenshot of it:

      In this example, it would be `card0-DP-1` (the DisplayPort)
2. Get port name for the connector we want to disable using xrandr: `xrandr -display :0.0 | grep "connected"`. This is the first word on the line for the related display (eg. “DisplayPort-0”).
    - This will return something like

      ```
      DisplayPort-0 disconnected primary (normal left inverted right x
      axis y axis)
      DVI-0 connected 655x500+0+0 (normal left inverted right x axis y
      axis) 0mm x 0mm
      ```

      Here's a screenshot of it:

      In this example our card output is `DisplayPort-0` (notice that it is marked as disconnected). Note all the displays shown here.
3. Place the following file into `/etc/X11/xorg.conf.d/`, replacing the outputs as appropriate, to disable both the card and the port output:

   10-monitor.conf

   ```
   Section "Monitor"
       Identifier "[card-output]"
       Option "Ignore" "true"
   EndSection

   Section "Monitor"
   ```

```
    Identifier "[port-name]"
    Option "Ignore" "true"
EndSection
```

Here is an example `10-monitor.conf` file using the example outputs from above:

10-monitor.conf

```
Section "Monitor"
    Identifier "card0-DP-1"
    Option "Ignore" "true"
EndSection

Section "Monitor"
    Identifier "DisplayPort-0"
    Option "Ignore" "true"
EndSection
```

Finish by saving the filesystem overlay with the following command:

```
batocera-save-overlay
```

This will make the changes persist to the next boot (updating Batocera will remove them, however, so remember to re-apply the steps in this guide if you intend on updating). You can check that the changes have taken affect by running `xrandr -display :0.0 | grep "connected"` again; this time you'll no longer see the digital output that you've disabled:

```
xrandr -display :0.0 | grep "connected"
DVI-0 connected primary 640x480+0+0 (normal left inverted right x axis y axis) 0mm x 0mm
```

Enabling Nvidia 900 series cards to show lower modelines

This is a workaround to allow the GTX 960, 970, 980 and 980ti (ones with a DVI-I port) to show lower resolutions than what they technically report as supported to the motherboard. It is not necessary on other cards.

Place the following file at `/etc/X11/xorg.conf.d/99-nvidia.conf` and run `batocera-save-overlay` afterward:

99-nvidia.conf

```
Section "Device"
    Identifier     "Card0"
    Driver         "nvidia"
    Option         "ModeValidation"
"AllowNon60HzDFPModes,NoMaxPclkCheck,NoEdidMaxPClkCheck,AllowInterlaced
Modes,NoMaxSizeCheck,NoHorizSyncCheck,NoVertRefreshCheck,NoEdidDFPMaxSi
zeCheck"
```

```
EndSection
```

Make 640x480i the default resolution on PC

If you are using a VGA signal display, it may not support an interlaced signal (indicated by the “i” at the end of the resolution and the “interlace” option in the timings information). This step isn't necessary for VGA signal displays that have an EDID in the first place anyway. Here's a handy list of the maximum progressive resolutions used by old EDID-less IBM monitors from 1987 to 1993:

Click here if you have an old IBM monitor from 1987 to 1993 to view their maximum resolutions

Model	Month/Year	Standard	Physical size	Viewable	Resolution	Dot pitch (mm)	Notes
8512	Apr 1987	VGA	14“	11.8”	640×480	0.41	
8513	Apr 1987	VGA	12“	10.4”	640×480	0.28	
8514	Apr 1987	XGA	16“	14.4”	1024×768	0.31	
8515	Sept 1991	XGA	14“	12.3”	640×480	0.28	
8511	Mar 1992	VGA	14“	11.8”	640×480	0.39	
6312	Sept 1992	SVGA	14“	11.7”	1024×768	0.28	
6314	Sept 1992	SVGA	14“	12”	1024×768	0.28	
6319	Sept 1992	SVGA	15“	12.6”	1024×768	0.28	
9515	Sept 1992	XGA	14“	12.3”	1024×768	0.28	
9518	Sept 1992	VGA	14“	12.3”	640×480	0.28	
9517	Nov 1992	XGA	17“	14.7”	1280×768	0.26 (stripe)	Trinitron
6317	Apr 1993	SVGA	17“	14.7”	1280×1024	0.28	
6318	Apr 1993	SVGA	14“	12.1”	800×600	0.39	“Low End”

Taken from page 2 of Personal Systems Reference IBM Monitors April 1987 to 2005 - withdrawn.

Get the system level ID for your display. Run `xrandr -display :0.0 | grep "connected"`. You will get an output similar to the following:

```
xrandr -display :0.0 | grep "connected"
DVI-0 connected primary 640x480+0+0 (normal left inverted right x axis y axis) 0mm x 0mm
```

Identify the port name of the port you intend to use for your CRT display. In this case it is `DVI-0` Then deduce what modeline is compatible with your display. This can usually be found by looking up the model number of your display along with “specifications”. If you don't know, then you can use the following 640x480i modeline that should be accepted by pretty much every display.

To enable the “640x480i” modeline to be the default xinitrc configuration:

1. Open the `/etc/X11/xinit/xinitrc` file. You should make a backup of this file before editing it just in case.
2. Go to the blank line just before `openbox –config-file /etc/openbox/rc.xml –startup “emulationstation-standalone”` and insert a new blank line.
3. Add the new xrandr modeline to it. This will force all running programs on Batocera to run at this default full-screen resolution. For example, you could paste in this text in that new blank line:

```
########################
#####-CRT CONFIG-#######
########################
##-Default Resolution-##
########################
xrandr -display :0.0 --newmode "640x480i" 13.10 640 664 728 832 480 484
490 525 interlace -hsync -vsync
xrandr -display :0.0 --addmode DVI-0 "640x480i"
xrandr -display :0.0 --output DVI-0 --mode "640x480i"
########################
########################
```

   - Or if you were using a VGA monitor with no EDID you would use this instead:

```
########################
#####-CRT CONFIG-#######
########################
##-Default Resolution-##
########################
xrandr -display :0.0 --newmode "640x480_60 30.120000KHz
60.000000Hz" 24.577920 640 694 701 816 480 482 483 502   -hsync -
vsync
xrandr -display :0.0 --addmode VGA-0 "640x480"
xrandr -display :0.0 --output VGA-0 --mode "640x480"
########################
########################
```

     or if that didn't work for your EDID-less VGA monitor:

```
########################
#####-CRT CONFIG-#######
########################
##-Default Resolution-##
########################
xrandr -display :0.0 --newmode "800x600_60 37.620000KHz
60.000000Hz" 41.193900 800 891 902 1095 600 602 603 627   -hsync -
vsync
xrandr -display :0.0 --addmode VGA-0 "800x600"
xrandr -display :0.0 --output VGA-0 --mode "800x600"
########################
########################
```

Save the file, then make the file permanent by running `batocera-save-overlay` again, and reboot. You are now booting up in 640x480i (480p if you used the EDID-less VGA variant) mode. In **MAIN MENU → SYSTEM SETTINGS → VIDEO OUTPUT**, set your default video output to the port you intend to use (“DVI-0” or “VGA”). Reboot one more time.

Here's an example /etc/X11/xinit/xinitrc file wit the CRT modeline added for reference (click to expand)

xinitrc

```
#!/bin/sh

# hide mouse cursor
unclutter --noevents -b

# disable dpms to prevent screen from blanking
xset -dpms
xset s off

# allow coredumps for ES
ulimit -H -c unlimited
ulimit -S -c unlimited emulationstation

# dbus launch is required for the gio/gvfs/trash:///...
eval `dbus-launch --sh-syntax --exit-with-session`

### intel-iris-driver ###
irisdriver="$(/usr/bin/batocera-settings-get -f "$BOOTCONF" intel-
i965-driver)"
if test ! -z "${irisdriver}" -a "${irisdriver}" = true; then
    # Force use i965 driver through global environment variable
    export MESA_LOADER_DRIVER_OVERRIDE=i965
fi
####################


### nvidia ###
# these are 3 variables used only by nvidia to take nvidia gpu
over intel cards when such hybrid cards are available
nvidia_driver="$(/usr/bin/batocera-settings-get -f /boot/batocera-
boot.conf nvidia-driver)"
if test "${nvidia_driver}" = "true"
then
    nvidia_prime="$(/usr/bin/batocera-settings-get -f
/boot/batocera-boot.conf nvidia-prime)"
    if test "${nvidia_prime}" = "true"
    then
    export __NV_PRIME_RENDER_OFFLOAD=1
    export __VK_LAYER_NV_optimus=NVIDIA_only
    export __GLX_VENDOR_LIBRARY_NAME=nvidia
    fi
```

```
fi

### radeon ###
# variable for AMD Dynamic Switchable Graphics to take amd-radeon
gpu over intel cards when such hybrid cards are available
radeon_prime="$(/usr/bin/batocera-settings-get -f /boot/batocera-
boot.conf radeon-prime)"
if test "${radeon_prime}" = "true"
then
    export DRI_PRIME=1
fi

########################
#####-CRT CONFIG-#######
########################
##-Default Resolution-##
########################
xrandr -display :0.0 --newmode "640x480i" 13.10 640 664 728 832
480 484 490 525 interlace -hsync -vsync
xrandr -display :0.0 --addmode DVI-0 "640x480i"
xrandr -display :0.0 --output DVI-0 --mode "640x480i"
########################
########################

openbox --config-file /etc/openbox/rc.xml --startup
"emulationstation-standalone"
```

Disable EmulationStation from forcing 60Hz

By default, EmulationStation will try to force a refresh rate at 60 Hz at all times. This can have adverse effects on games like frame pacing (uneven scrolling), audio (crackling) and timing (speed of gameplay, physics) issues. Here's a video of someone explaining frame pacing very slowly.

We will disable ES from forcing 60 Hz at all times by commenting out some lines.

Make a backup of `/usr/bin/emulationstation-standalone`. You can do so from SSH with the following:

```
cp /usr/bin/emulationstation-standalone /usr/bin/emulationstation-
standalone.bak
```

Then, underneath the `# try to force 60hz (specific to xorg)` line, comment (add a `#` in front of) the following config lines from it:

```
    FRAMERATE="$(/usr/bin/batocera-settings-get es.framerate)"
    test -z "${FRAMERATE}" && FRAMERATE=60
```

```
    which xrandr && xrandr -r "${FRAMERATE}"
```

Your file should look like this now:

```
    # try to force 60hz (specific to xorg)
    # FRAMERATE="$(/usr/bin/batocera-settings-get es.framerate)"
    # test -z "${FRAMERATE}" && FRAMERATE=60
    # which xrandr && xrandr -r "${FRAMERATE}"
    ###################
```

Don't forget to `batocera-save-overlay` once again. Reboot.

Managing overscan and centering in EmulationStation

As you might have noticed some of EmulationStation's menu is cut off and not centered. You may have gotten lucky and have a really well tuned CRT display so these steps may not be necessary, but for most CRT displays they will be. For this configuration you need to be able to look at your CRT display directly to see the changes we are going to make.

Alternatively: use a theme optimized for CRTs that moves all elements inwards to account for overscan and to disable all “crop overscan” options in emulators/cores, getting a more authentic retro-gaming experience!

Offset (position)

We will start by aligning the top-left of the image with the top-left corner of the display. This will be important for the next step (scaling). We will do this by utilizing the following argument:

```
--screenoffset [x] [y]          Move the canvas by [x] pixels right and [y] pixels down.
```

Open `userdata/system/batocera.conf` and go down to this comment:

```
## Configurations generated by Batocera.linux
```

Add the following lines to right before that comment:

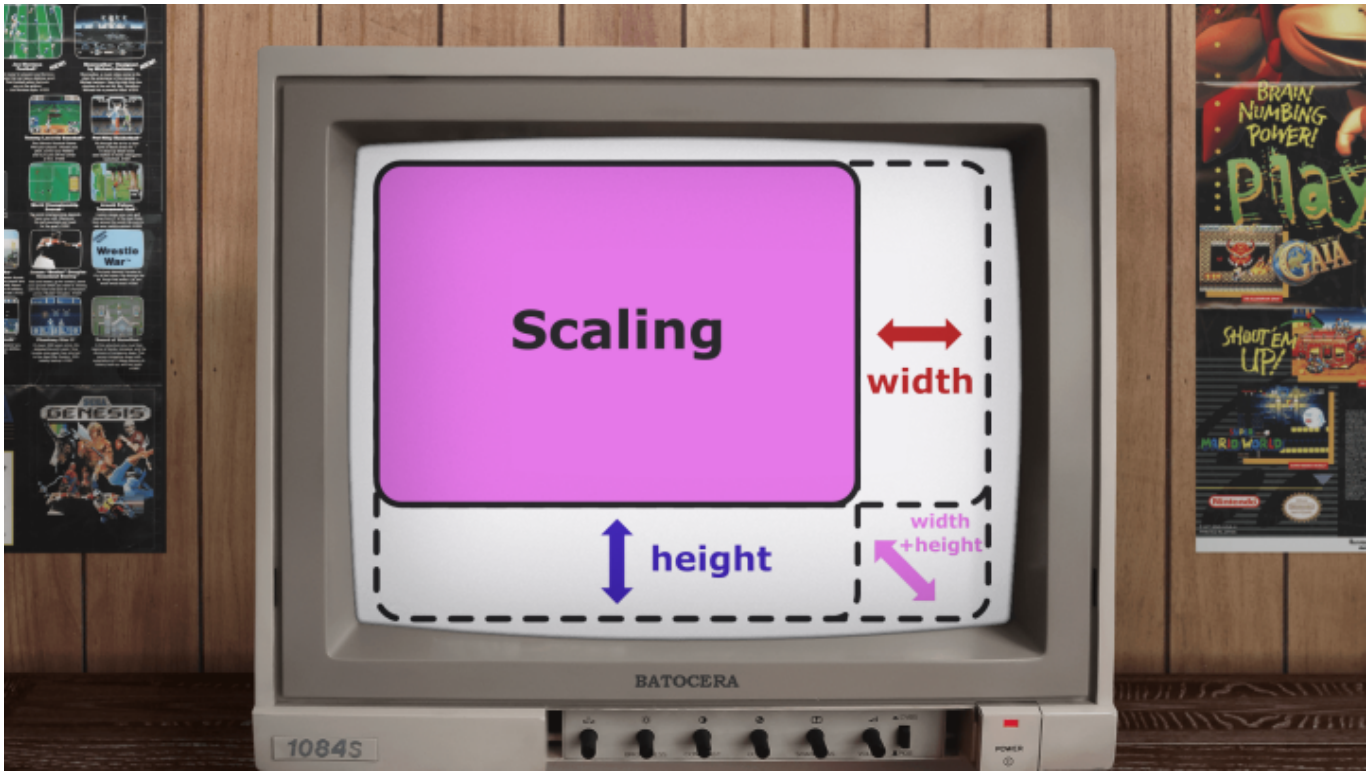
```
## ES Settings
es.customsargs=--screenoffset 00 00
```

Save the file. To see the changes in effect, restart EmulationStation with the following command:

```
batocera-es-swissknife --restart
```

Increase or decrease these numbers until the top-left pixel is aligned where you want it to be. Check below for comparisons.

Scaling (size)



CRTs were a bit loosey-goosey with having every pixel line up 1:1 with the signal, even with a perfectly physically aligned screen you may see pixels blurring into each other. Whether you want to “fix” this is up to you. This part will simply show you how to scale your screen so you don't miss out on any content. We will do this by utilizing the following argument:

```
--screensize [width] [height]   Output resolution (top-left is the anchor). Increasing will stretch the image out to the right and downwards; decreasing will squash the image from the right side toward the left and from the bottom upwards.
```

Let's start by using the same resolution as you boot resolution. In this case, `640x480i`.

Open `userdata/system/batocera.conf` again and go down to the `es.customargs` line you made earlier.

Append `--screensize 640 480` to appear a space before `--screenoffset ## ##`. Following the example from before, it would end up looking like this:

```
## ES Settings
es.customsargs=--screensize 640 480 --screenoffset 00 00

## Configurations generated by Batocera.linux
```

After adjustment, it may look like this:

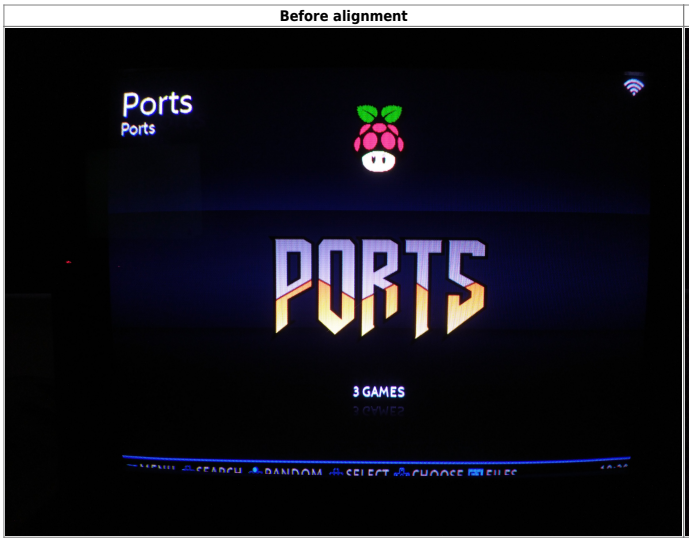
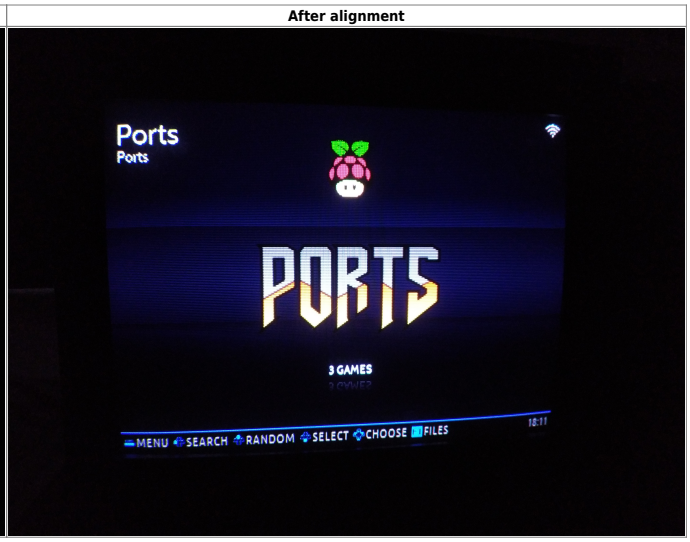
```
## ES Settings
es.customsargs=--screensize 640 488 --screenoffset 38 16

## Configurations generated by Batocera.linux
```

Save the file. To see the changes in effect, restart EmulationStation with the following command:

```
batocera-es-swissknife --restart
```

Increase or decrease these numbers until the bottom-right pixel is aligned where you want it to be. Check below for comparisons.

Results of proper scaling and offset

Before alignment	After alignment
	

Persist screen scaling after exiting an emulator

We also need to make a script so this setting persists even after exiting an emulator. Otherwise, EmulationStation will reset back to the default offset and scaling when returning to the menu:

Video demonstrating misalignment and flickering after exiting an emulator.Video demonstrating misalignment and flickering after exiting an emulator.

Make a folder inside `/userdata/system/` called `scripts`. Then download the following file and save it to that location. Edit the `[your video port]` to your respective port (example below).

first_script.sh

```
#!/bin/bash

#Case selection for first parameter parsed
case $1 in
    gameStop)
        xrandr -display :0.0 --output [your video port] --mode
"640x480i"
    ;;
esac
```

Make the script executable with `chmod +x /userdata/system/scripts/first_script.sh`

Here is an example file using a DVI port for reference:

first_script.sh

```
#!/bin/bash
#This will force ES to refresh its display, grabbing the correct custom
arguments we've set.

#Set logfile location and filename
#logfile=/tmp/scriptlog.txt

#Case selection for first parameter parsed
case $1 in
#    gameStart)
#        echo "START" > $logfile
#        echo "$@" >> $logfile
#    ;;

    gameStop)
        xrandr -display :0.0 --output DVI-0 --mode "640x480i"
    ;;
esac
```

Finish with `batocera-save-overlay` and reboot. You should now be able to exit an emulator without EmulationStation forgetting your scaling and offset!

Video demonstrating that ES no longer forgets alignment after exiting an emulator.Video demonstrating that ES no longer forgets alignment after exiting an emulator.

Adding Modelines (Read Resolutions) for Emulators

Right now we only have one usable modeline and that is 640x480i.

We would like to add more modelines for external emulators like for example PS2, PSP, GameCube, Wii, Wine (PC) and so on.

To do this we are going to use a program called Switchres made by Calamity. This has been integrated into Batocera as of v30. More information about Switchres can be found on the GroovyMame wiki.

However it is possible to bypass setting up switchres entirely by defining the resolution modes manually in the config file appropriate to the GPU being used. This is not ordinarily recommended but can make the process much more automated and portable as the video modes will now appear in the menu as available video modes.

Manually define modes for use in the menu

Place this file at `/userdata/system/99-nvidia.conf`:

99-nvidia.conf

```
# Config for 15Khz Monitor
Section "Monitor"
        Identifier      "Disabled"
        Option  "ignore"        "true"
##########################################
EndSection

Section "Monitor"
    Identifier  "CRT"
    VendorName  "Batocera"

    Option      "Enable" "true"
    Option      "DPMS" "False"
    Option      "DefaultModes" "False"
        Modeline "3600x480" 73.413510 3600 3747 4092 4679 480 483
489 523 interlace  -hsync -vsync
    Modeline "1920x240" 37.778 1920 1977 2154 2376 240 243 245 265
-hsync -vsync
        Modeline "1920x256" 39.015450 1920 1998 2181 2493 256 276
279 313 -hsync -vsync
        Modeline "1920x480" 73.413510 3600 3747 4092 4679 480 483
489 523 interlace -hsync -vsync
        Modeline "2560x256" 52.42 2560 2664 2912 3328 256 257 260
276 -hsync -vsync
        Modeline "2560x448" 52.22 2560 2664 2912 3328 448 467 473
523 interlace -hsync -vsync
        Modeline "1280x480_60,0Hz 15,7KHz (60Hz)" 26.181840 1280
1362 1429 1639 480 490 496 533 -hsync -vsync interlace
        Modeline "1024x600_50,0Hz 16,0KHz (50Hz)" 20.161 1024 1032
1127 1264 600 601 607 638 interlace -hsync -vsync
        Modeline "768x576" 15.627975 768 799 872 997 576 583 589
627 interlace  -hsync -vsync
    Modeline "854x480" 17.415900 854 889 971 1110 480 483 489 523
interlace  -hsync -vsync
    Modeline "864x486" 17.802060 864 900 984 1126 486 488 494 527
interlace  -hsync -vsync
        Modeline "800x600_50,0Hz 16,0KHz (60Hz)" 15.823 800 808
883 992 600 601 607 638 interlace -hsync -vsync
        Modeline "720x480_60,0Hz 15,7KHz (60Hz)" 13.959 720 728
794 888 480 483 489 524 interlace -hsync -vsync
        Modeline "640x480_60,0Hz 15,7KHz (60Hz)" 12.324 640 648
706 784 480 483 489 524 interlace -hsync -vsync
        #Modeline "320x240" 6.514560 320 333 364 416 240 242 245
261   -hsync -vsync

EndSection

Section "Device"
  Identifier    "GRPX"
```

```
    Option "RenderAccel"   "1"
    Option "ModeValidation" "NoVertRefreshCheck, NoHorizSyncCheck,
NoMaxSizeCheck, NoMaxPClkCheck ,
NoVesaModes,NoXServerModes,NoEDIDModes, NoPredefinedModes
,NoExtendedGpuCapabilitiesCheck ,
AllowDpInterlaced,AllowNonEdidModes,NoDisplayPortBandwidthCheck
,NoEdidDFPMaxSizeCheck,NoDualLinkDVICheck"
    Option "UseEDID" "False"
    Option "NoEDIDModes" "true"
    Option "monitor-DVI-0" "CRT"
    Option "monitor-DVI-1" "CRT"
    Option "monitor-VGA-0" "CRT"
    Option "monitor-VGA-1" "CRT"
    Option "monitor-HDMI-0" "CRT"
    Option "monitor-HDMI-1" "CRT"
    Option "monitor-HDMI-2" "CRT"
    Option "monitor-DP-0" "CRT"
    Option "monitor-DP-1" "CRT"
    Option "monitor-DP-2" "CRT"
    Option "monitor-DIN" "Disabled"
    Option "monitor-DIN-0" "Disabled"
    Option "monitor-DIN-1" "Disabled"
    #Option "monitor-DP-0" "Disabled"
    #Option "monitor-DP-1" "Disabled"
    #Option "monitor-DP-2" "Disabled"
    #Option "monitor-HDMI-0" "Disabled"
    #Option "monitor-HDMI-1" "Disabled"
    #Option "monitor-HDMI-2" "Disabled"

EndSection


Section "Screen"
    Identifier     "Screen0"
    Device         "GRPX"
    Monitor        "CRT"
    Option         "AllowIndirectGLXProtocol" "off"
    Option         "TripleBuffer" "on"
EndSection


Section "ServerFlags"
    Option "blank time" "0"
    Option "standby time" "0"
    Option "suspend time" "0"
    Option "off time" "0"
    Option "dpms" "false"
    Option "Xinerama" "0"
    Option "AllowEmptyInitialConfiguration" "true"
EndSection
```

If this method is taken, skip over to this section. If you are not doing that, read on.

Creating the switchres.ini file

Switchres will be activated once the appropriate `switchres.ini` file with the correct permissions has been found at `/etc/switchres.ini`. If you intend on copying the file to this destination, you must do so from within Batocera (the virtual filesystem cannot be accessed when Batocera isn't running) and running `batocera-save-overlay`.

You can download the `switchres.ini` file required from the Switchres Github page, transfer it to your Batocera machine over the network share and then use its file manager (F1) to put the file in the `/etc/` folder.

Creating switchres.ini using only SSH

If you cannot do the above, or would just like to use only SSH, do the following:

We will use `nano` to create a file named `switchres.ini` inside the `/etc/` folder. Run the following command:

```
nano /etc/switchres.ini
```

This will open up a new empty file for editing. Copy and paste the contents of the INI file below into your SSH session (remember, you only need to right-click to paste):

Click here to reveal

switchres.ini

```
#
# Switchres config
#

# Monitor preset. Sets typical monitor operational ranges:
#
# generic_15, ntsc, pal                    Generic CRT standards
# arcade_15, arcade_15ex                   Arcade fixed frequency
# arcade_25, arcade_31                     Arcade fixed frequency
# arcade_15_25, arcade_15_25_31            Arcade multisync
# vesa_480, vesa_600, vesa_768, vesa_1024  VESA GTF
# pc_31_120, pc_70_120                     PC monitor 120 Hz
# h9110, polo, pstar                       Hantarex
# k7000, k7131, d9200, d9800, d9400        Wells Gardner
# m2929                                    Makvision
# m3129                                    Wei-Ya
```

```
# ms2930, ms929                                 Nanao
# r666b                                         Rodotron
#
# Special presets:
# custom   Defines a custom preset. Use in combination with
crt_range0-9 options below.
# lcd      Will keep desktop's resolution but attempt variable
refresh, use in combination with lcd_range
#
    monitor                   arcade_15

# Define a custom preset, use monitor custom to activate
# crt_range0-9   HfreqMin-HfreqMax, VfreqMin-VfreqMax,
HFrontPorch, HSyncPulse, HBackPorch, VfrontPorch, VSyncPulse,
VBackPorch, HSyncPol, VSyncPol, ProgressiveLinesMin,
ProgressiveLinesMax, InterlacedLinesMin, InterlacedLinesMax
# e.g.: crt_range0  15625-15750, 49.50-65.00, 2.000, 4.700, 8.000,
0.064, 0.192, 1.024, 0, 0, 192, 288, 448, 576
    crt_range0                auto
    crt_range1                auto
    crt_range2                auto
    crt_range3                auto
    crt_range4                auto
    crt_range5                auto
    crt_range6                auto
    crt_range7                auto
    crt_range8                auto
    crt_range9                auto

# Set the operational refresh range for LCD monitor, e.g.
lcd_range 50-61
    lcd_range                 auto

# Force a custom modeline, in XFree86 format. This option
overrides the active monitor preset configuration.
    modeline                  auto

# Forces an user mode, in the format: width x height @ refresh.
Here, 0 can used as a wildcard. At least one of the three values
# must be defined. E.g. user_mode 0x240 -> SR can freely choose
any width based on the game's requested video mode, but will
# force height as 240.
    user_mode                 auto



#
# Display config
#

# Select target display
# auto               Pick the default display
```

```
# 0, 1, 2, ...       Pick a display by index
# \\.\DISPLAY1, ...  Windows display name
# VGA-0, ...         X11 display name
    display                   auto

# Choose a custom video backend when more than one is available.
# auto         Let Switchres decide
# adl          Windows - AMD ADL (AMD Radeon HD 5000+)
# ati          Windows - ATI legacy (ATI Radeon pre-HD 5000)
# powerstrip   Windows - PowerStrip (ATI, Nvidia, Matrox, etc.,
models up to 2012)
# xrandr       Linux - X11/Xorg
# drmkms       Linux - KMS/DRM (WIP)
    api                       auto

# [Windows] Lock video modes reported as unsupported by your
monitor's EDID
    lock_unsupported_modes    1

# Lock system (non-custom) video modes, only use modes that have
full detailed timings available
    lock_system_modes         0

# Ignore video mode's refresh reported by the OS when checking
ranges
    refresh_dont_care         0

# Keep changes on exit (warning: this skips video mode cleanup)
    keep_changes              0



#
# Modeline generation config
#

# Enable on-the-fly generation of video modes
    modeline_generation       1

# Allow interlaced modes (existing or generated)
    interlace                 1

# Allow doublescan modes (warning: doublescan support is broken in
most drivers)
    doublescan                0

# Force a minimum dotclock value, in MHz, e.g. dotclock_min 25.0
    dotclock_min              0

# Maximum refresh difference, in Hz, allowed in order to
synchronize. Below this value, the mismatch does not involve
penalization
```

```
    sync_refresh_tolerance    2.0

# Super resolution width: above this width, fractional scaling on
the horizontal axis is applied without penalization
    super_width               2560

# Physical aspect ratio of the target monitor. Used to compensate
aspect ratio when the target monitor is not 4:3
    aspect                    4:3

# [Experimental] Attempts to compensate consumer TVs vertical
centering issues
    v_shift_correct           0

# Calculate horizontal borders with 1-pixel precision, instead of
the default 8-pixels blocks that were required by old drivers.
# Greatly improves horizontal centering of video modes.
    pixel_precision           1

# Calculate all vertical values of interlaced modes as even
numbers. Required by AMD APU hardware on Linux
    interlace_force_even      0



#
# Custom video backend config
#

# [X11] adjusts the crtc position after a new video mode is set,
maintaining the relative position of screens in a multi-monitor
setup.
    screen_compositing        0

# [X11] stacks the screens vertically on startup to allow each
screen to freely resize up to the maximum width. Useful to avoid
video
# glitches when using super-resolutions. screen_reordering
overrides screen_compositing.
    screen_reordering         0

# [Windows] dynamically adds new modes or updates existing ones,
even on stock AMD drivers*. This feature is experimental and is
# disabled by default. It has the following limitations and
problems:
# - Synchronization is not perfect yet and the new modes may not
always be ready on time for mode switching, causing a wrong
display
#   output.
# - A plug-n-play audio notification will be present on startup
and exit, if the explorer shell is used.
# - Refreshing the hardware is an expensive task that takes time,
```

```
specially if the app has already entered fullscreen mode. This
#   makes it unpractical for games that switch video modes more
than once.
# * When used with stock AMD drivers instead of CRT Emudriver,
usual limitations apply: no support for low resolutions (below
640x480)
#   nor low dotclocks.
#   Not a problem however if you're using a 31 kHz monitor.
    allow_hardware_refresh    0

# Pass a custom video timing string in the native backend's
format. E.g. pstring timing for Powerstrip
    custom_timing             auto



#
# Logging
#

# Enables verbose mode (0|1)
    verbose                   0

# Set verbosity level (from 0 to 3)
# 0: no messages from SR
# 1: only errors
# 2: general information
# 3: debug messages
    verbosity                 2
```

Press `[Ctrl]+[S]` to save, and then exit with `[Ctrl]+[X]`.

Selecting your monitor type

Switchres will need to know what kind of CRT display you have in order to generate modelines for it. Essentially, this is just setting the maximum bandwidth that the modeline can use. Open up the `switchres.ini` file you just created with either Batocera's file manager or `nano /etc/switchres.ini` again. Scroll down to the `monitor` line. By default, it is:

```
    monitor                   arcade_15
```

There are several types of CRT display presets to choose from. Here are a few common ones:

```
generic_15          Generic 15.7 kHz
arcade_15           Arcade 15.7 kHz - standard resolution
pal                 PAL TV - 50 Hz/625
ntsc                NTSC TV - 60 Hz/525
arcade_15ex         Arcade 15.7-16.5 KHz - extended resolution
```

```
pc_31_120           PC CRT 31 KHz/120Hz
pc_70_120           PC CRT 70 KHz/120Hz
```

`arcade_15` works well for most standard CRTs. `generic_15` is also a good if your CRT is a bit more picky. `pc_31_120` for old CRT monitors (pre-1994) and `pc_70_120` for newer CRT monitors (post-1994) in general.

Replace `arcade_15` with your intended preset. For example, if we were switching to `generic_15`:

```
    monitor                   generic_15
```

Save the file and close it. If using `nano`, that's `[Ctrl]+[S]` and `[Ctrl]+[X]`.

If your display device does not support one of the built-in presets, then you'll need to use the custom monitor type here.

Changing the file permissions

We need to change the file access permissions. Run the following command:

```
chmod 0777 /etc/switchres.ini
```

Once again, run `batocera-save-overlay` to save the changes.

Creating a modeline

In order to create a new modeline we must first create the appropriate syntax for it. A typical modeline applied via xrandr to a particular display looks like the following:

```
xrandr -display :0.0 --newmode "320x240_60" 6.514560 320 333 364 416 240 242
245 261 -hsync -vsync
xrandr -display :0.0 --addmode "320x240_60"
```

and it will be put into the `/etc/X11/xinit/xinitrc` file. Let's get started.

You can refer to the section below for a modeline to generate. In this example we are going to generate the modeline for **320×240@60hz**

```
switchres 320 240 60 -i switchres.ini -c
```

It should output a single line similar to this:

```
Switchres: Modeline "320x240_60 15.660000KHz 60.000000Hz" 6.514560 320 333
364 416 240 242 245 261   -hsync -vsync
```

We have now generated the modeline based on our monitor preset **arcade_15**. Copy this line (from after the word Modeline) and save it to a temporary text file.

```
"320x240_60 15.660000KHz 60.000000Hz" 6.514560 320 333 364 416 240 242 245
261   -hsync -vsync
```

You can opt to shorten the name by removing the `15.660000KHz 60.000000Hz` part (it is implied):

```
"320x240_60" 6.514560 320 333 364 416 240 242 245 261 -hsync -vsync
```

Then add this line to `/etc/X11/xinit/xinitrc`.

Then we will need to tell xorg about the name of this modeline. Simple, add another line underneath that with just the part in double-quotes (“).

```
"320x240_60" 6.514560 320 333 364 416 240 242 245 261 -hsync -vsync
"320x240_60"
```

This mode will tell xorg that the modeline's name is “320x240_60”, and it will use the information from the line above it to enforce it. Why does there need to be an extra line to do this? I'd like to know too.

Alternatively, you can use the online tool The XFree86 Modeline Generator to calculate new modelines as well. Instructions on its usage are outside the scope of this article if you choose to use it.

Switchres usage

Regular modeline generation (read the creating a modeline section above):

```
# switchres 320 240 60 -i switchres.ini -c
```

which would output:

```
Switchres: Modeline "320x240_60 15.660000KHz 60.000000Hz" 6.514560 320 333
364 416 240 242 245 261   -hsync -vsync
```

You can also skip reading the `switchres.ini` file by directly specifying the monitor in the command:

```
# switchres 320 240 60 -m generic_15 -c
```

which would output:

```
Switchres: Modeline "320x240_60 15.660000KHz 60.000000Hz" 6.514560 320 333
364 416 240 242 245 261   -hsync -vsync
```

Forced modeline generation:

```
# switchres 854 480 60 -f 854x480@60 -i switchres.ini -c
```

which would output the following:

```
Switchres: Modeline "854x480_60i 15.690000KHz 60.000000Hz" 17.415900 854 889
971 1110 480 483 489 523 interlace  -hsync -vsync
```

Telling xinitrc about the new modeline(s)

Now we need to add the modeline to the `xinitrc` configuration. Open `/etc/X11/xinit/xinitrc` and scroll down to

```
########################
#####-CRT CONFIG-#######
########################
##-Default Resolution-##
########################
xrandr -display :0.0 --newmode "640x480i" 13.10 640 664 728 832 480 484 490
525 interlace -hsync -vsync
xrandr -display :0.0 --addmode DVI-0 "640x480i"
xrandr -display :0.0 --output DVI-0 --mode "640x480i"
########################
########################

openbox --config-file /etc/openbox/rc.xml --startup "emulationstation-
standalone"
```

On a new line after `xrandr -display :0.0 --output DVI-0 --mode "640x480i"`, type `xrandr -display :0.0 –newmode` and then paste in the line you copied to the temporary text file created in the previous section.

Then on another new line, type `xrandr -display :0.0 –addmode` followed by just the part enclosed in double-quotes from your modeline (including the double-quotes).

In our example the file would end up looking like this:

```
########################
#####-CRT CONFIG-#######
########################
##-Default Resolution-##
########################
xrandr -display :0.0 --newmode "640x480i" 13.10 640 664 728 832 480 484 490
525 interlace -hsync -vsync
xrandr -display :0.0 --addmode DVI-0 "640x480i"
xrandr -display :0.0 --output DVI-0 --mode "640x480i"
########################
#####-Modelines-########
########################
```

```
xrandr -display :0.0 --newmode "320x240_60" 6.514560 320 333 364 416 240 242
245 261 -hsync -vsync
xrandr -display :0.0 --addmode "320x240_60"

openbox --config-file /etc/openbox/rc.xml --startup "emulationstation-
standalone"
```

Finish with `batocera-save-overlay` and reboot.

Modelines

Here are some resolutions to use

- Super Resolutions
  - 2560×240@60
  - 2560×248@58
  - 2560×256@57
  - 2560×264@55
  - 2560×272@54
  - 2560×280@52
  - 2560×288@51
  - 2560×448@60
  - 2560×464@60
  - 2560×480@60
  - 2560×496@58
  - 2560×512@57
  - 2560×544@54
  - 2560×560@52
- Emulator/Wine/PC
  - 240×240@60
  - 256×192@60
  - 288×224@60
  - 320×180@60
  - 320×200@60
  - 320×240@60
  - 320×240@60i
  - 320×256@55
  - 320×256@60
  - 352×240@60
  - 360×200@60
  - 360×240@60
  - 380×284@60
  - 384×216@60
  - 384×480@60i
  - 400×240@60
  - 416×240@60
  - 426×240@60
  - 428×240@60
  - 456×256@55

- 460×200@60
- 464@272@50
- 480×270@50
- 480×270@60
- 480×272@60
- 512×480@60i
- 640×240@60
- 640×360@60
- 640×480@60
- 640×480@60i
- 854×480@60i (4:3 from 16:9)

Switchres - minimum dot clock

Most of this information has been gathered from GroovyMame's wiki.

Fun fact about CRTs: they receive more data than they actually display and keep those extra bits outside of the frame at the top and the bottom. This will be relevant for the next paragraph.

The dot clock. Essentially, the amount of pixels drawn per second. You can work this out by multiplying your width, your amount of horizontal lines (this is your height, but a bit larger than it) and frames per second. For example, a theoretical resolution of 320×240 at 60 FPS would be calculated by 320×262.5×60, which equals 5,040,000 pixels per second. This is typically measured in MHz (dot clock), but that's confusing so I'm just going to stick with pixels per second (p/s) for the unit and “bandwidth” for the noun.

The issue with modern graphics cards is that they have a **minimum** required bandwidth to function, anything below that will simply not render (or in some cases, lock up the hardware). Typical minimum bandwidths for these cards are 8 Mp/s, 12 Mp/s and 25 Mp/s. Sometimes the minimum bandwidths will differ depending on which ports are used, digital ports tend to have an even higher minimum bandwidth.

So what can we do about this? The answer is simple:

Super resolutions

CRTs are beautiful in the sense that they don't think in pixels, only lines. So essentially, what we can do is send a **BLOODY HUGE** amount of pixels per line to overcome the minimum required bandwidth for our graphics card. These are what super resolutions are.

It's typically good practice to use a *multiple* of what the native width of the original system was in order to not have pixels overlapping into other pixels when shrunk down by our CRT. However, we need to practice a bit of caution, as the transistors and other components in a CRT can only handle so much p/s before becoming overloaded and unresponsive, or possibly even damaged!

So therefore, we will let Switchres calculate the appropriate super resolution for us. It will work out the super resolution that is both a multiple of the original system's width and that satisfies the minimum

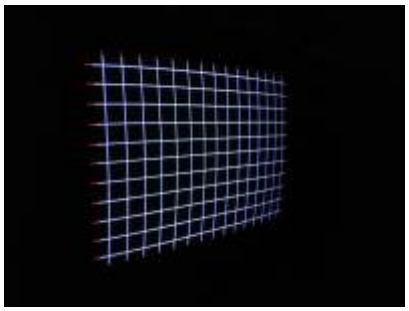
required bandwidth for our card. A good starting point is 8 million p/s, increasing if that doesn't work for your hardware.

Here's an example: let's say you're playing a game that runs at a native 640×480 at 60 FPS. Our graphics card has a minimum required bandwidth of 25 Mp/s. The bandwidth of that is calculated by 640x480x60 = 18,432,000 p/s. This is close to the required 25 Mp/s but not quite. Switchres calculates our super resolution as 1280×480, as 1280 is a *multiple* of 640 and the total bandwidth is 1280x480x60 = 36,864,000 p/s. This is well above our required minimum, and thus suited for our purposes.

Same goes for any other modeline, like for 256×224 it would calculate 512x224x60, 768x224x60 and so on until it meets the minimum required bandwidth (dot clock) set in the INI.

Super resolutions are only supported by MAME and Libretro cores.

Testing modelines



configure switchres.ini to your monitor (WinScp)

Connect to Batocera via SSH

Have a keyboard connected

Exit Emulation Station

```
/etc/init.d/S31emulationstation stop
```

From putty type for this to test modeline 640×480@60 with configured monitor profile

```
DISPLAY=:0 switchres 640 480 60 -i switchres.ini -s -l grid
```

You can change between 2 types of grids with Tab.

Exit pressing Escape from keyboard to try a new modeline. Do not use `Pause Break` key in SSH. Exit only with connected keyboard. If you do then you need to reboot.

Go back to Es with the command

```
batocera-es-swissknife --restart
```

Modelines to test your switchres.ini settings with

```
dotclock_min 0 (Default) 0-25 with decimal support 1.1, 8.3, 10.5 and so on
```

Always start with default setting 0 and work your way up. The example below is a low resolution so if your card can handle it then there is no need to change the dotclock.

Arcade Modelines

For example the Arcade game Dottori Kun (dotrikun)

```
DISPLAY=:0 switchres 128 192 61.035156 -i switchres.ini -s -l grid
```

AMD Troubleshooting Tearfree and Freezes

If you get uneven frame-pacing and microstutters in Mame and/or Retroarch we need to disable the option **TearFree** in the driver configuration file located at `/etc/X11/xorg.conf.d`

- 20-amdgpu.conf (GCN 3, GCN 4, GCN 5, RDNA & RDNA 2) or download an already fixed 20-amdgpu.conf

```
Section "OutputClass"
     Identifier "Fix AMD Tearing"
     Driver "amdgpu"
     MatchDriver "amdgpu"
     Option "TearFree" "false"
EndSection
```

- 20-radeon.conf (TeraScale and older, GCN 1 & GCN 2) or download an already fixed 20-radeon.conf

```
Section "OutputClass"
     Identifier "Fix AMD Radeon Tearing"
     Driver "radeon"
     MatchDriver "radeon"
     Option "TearFree" "false"
EndSection
```

Don't forget to `batocera-save-overlay` and reboot.

Games freeze or EmulationStation freeze.

This has been fixed by a new interlaced resolution patch by Calamity in v34 (beta) and newer versions

This has been reported by some users and myself included. It's not know yet what is the underlying reason. It could be a driver or kernel issue.

We need to add **Option “DRI” “2”** for radeon & **Option “DRI” “3”** for amdgpu in the driver configuration file located at `/etc/X11/xorg.conf.d`

- 20-amdgpu.conf (GCN 3, GCN 4, GCN 5, RDNA & RDNA 2) or download an already fixed 20-amdgpu.conf

```
Section "OutputClass"
     Identifier "Fix AMD Tearing"
     Driver "amdgpu"
     MatchDriver "amdgpu"
     Option "TearFree" "false"
     Option "DRI" "3"
EndSection
```

- 20-radeon.conf (TeraScale and older, GCN 1 & GCN 2) or download an already fixed 20-radeon.conf

```
Section "OutputClass"
     Identifier "Fix AMD Radeon Tearing"
     Driver "radeon"
     MatchDriver "radeon"
     Option "TearFree" "false"
     Option "DRI" "2"
EndSection
```

Don't forget to `batocera-save-overlay` and reboot.

- Short explanation of expression mentioned here

TearFree is a tearing prevention option which prevents tearing by using the hardware page flipping mechanism

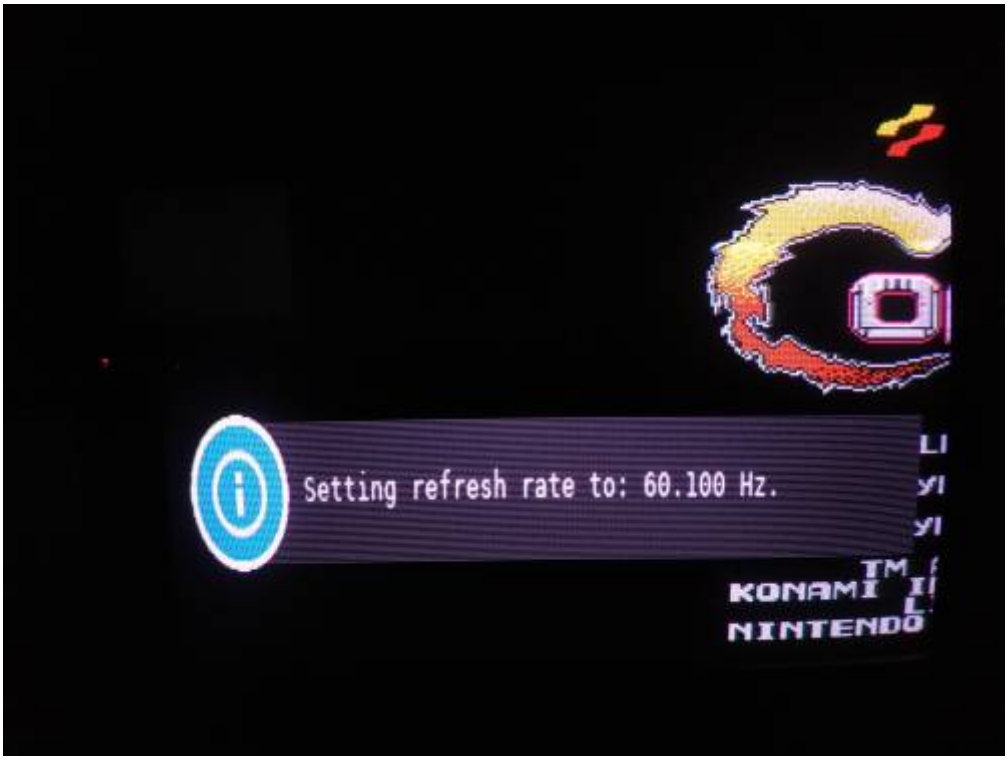
Microstutter short irregular frame dips

Frame-pacing uneven distribution of frames

Configure Libretro cores for use with Switchres

Optimizing RA for low resolution output

RetroArch's default menus and on-screen notifications were designed for high definition displays and make be illegible/absolutely HUGE on low resolution displays:



The following settings will do two things mainly: use the older RGUI menu system which was designed for smaller screens and disable post-display enhancements (not needed or applicable for CRT). Open `/userdata/system/batocera.conf`

Add this

```
## CRT CONFIG
global.retroarch.menu_driver=rgui
global.retroarch.menu_show_advanced_settings=true
global.retroarch.menu_enable_widgets=false
global.smooth=0
```

If you want a more authentic experience by disabling save states, shaders and bezels you can also add the following (these can also be toggled in **GAMES SETTINGS**):

```
global.rewind=0
global.autosave=0
global.shaderset=none
global.bezel=none
```

It is highly recommended to disable bezels if you're only intending on playing 4:3 systems.

Libretro core and directory overrides for use with CRT Switchres

You can define custom resolutions for consoles by using RetroArch's core override option. It works a tiny bit differently from native RetroArch in Batocera, but the process should be very familiar if you're

a long-time user of RetroArch.

For Switchres-specific configuration, we'll use a specially named INI file in the core overrides folder. Copy the existing `/etc/switchres.ini` to `/userdata/system/.config/retroarch/config/[core name]/[core name].switchres.ini`. To do this via SSH:

```
cp /etc/switchres.ini /userdata/system/.config/retroarch/config/[core name]/[core name].switchres.ini
```

with of course replacing `[core name]` with the name of your intended Libretro core.

Then open the file and scroll down to:

```
# Forces an user mode, in the format: width x height @ refresh. Here, 0 can used as a wildcard. At least one of the three values
# must be defined. E.g. user_mode 0x240 -> SR can freely choose any width based on the game's requested video mode, but will
# force height as 240.
    user_mode                auto
```

Here you can change `auto` to your desired resolution in [width]x[height] format. For instance:

```
    user_mode                768x576
```

This will then force RetroArch to output at a 576i resolution for that core. Switchres will still calculate an appropriate super resolution, it will just treat your user defined resolution as if though *that* were the console's native output resolution.

You can usually discover what resolution a console used by searching for it on Wikipedia. You can also give Libretro's official documentation a try, but they don't seem to have geometry information for a lot of systems (as of writing).

Forcing integer scaling while Switchres is enabled

Switchres is good at all for automatically sorting out our super resolutions, but what if we wanted **integer scaling** on our CRT?

Some consoles, such as portables like the Game Boy Advance, don't display their content at 480i. You might instead want to switch your CRT into progressive scan mode (by lowering the resolution to half of 480) and display the raw resolution with a correct aspect ratio.

This will require a combination of Switchres and RetroArch core overrides.

First create the `/userdata/system/.config/retroarch/config/[core name]/[core name].switchres.ini` as mentioned above and set your `user_mode` to a low vertical resolution like 240. This will be your final display output. It will need to be in a format your TV can handle, so if you want to force progressive scan it should have a vertical resolution of 240 lines (horizontal doesn't matter as much). It would have this line for example:

```
    user_mode                   320x240
```

The next step can be done via the RetroArch's **Settings → Video → CRT SwitchRes** menu explained in the next section, but we may as well set this while we're here. Create the new text file `/userdata/system/.config/retroarch/config/[core name]/[core name].cfg`. For example, `/userdata/system/.config/retroarch/config/mGBA/mGBA.cfg` would be the file created for the mGBA core.

The open that file and enter the following information:

core-name-here.cfg

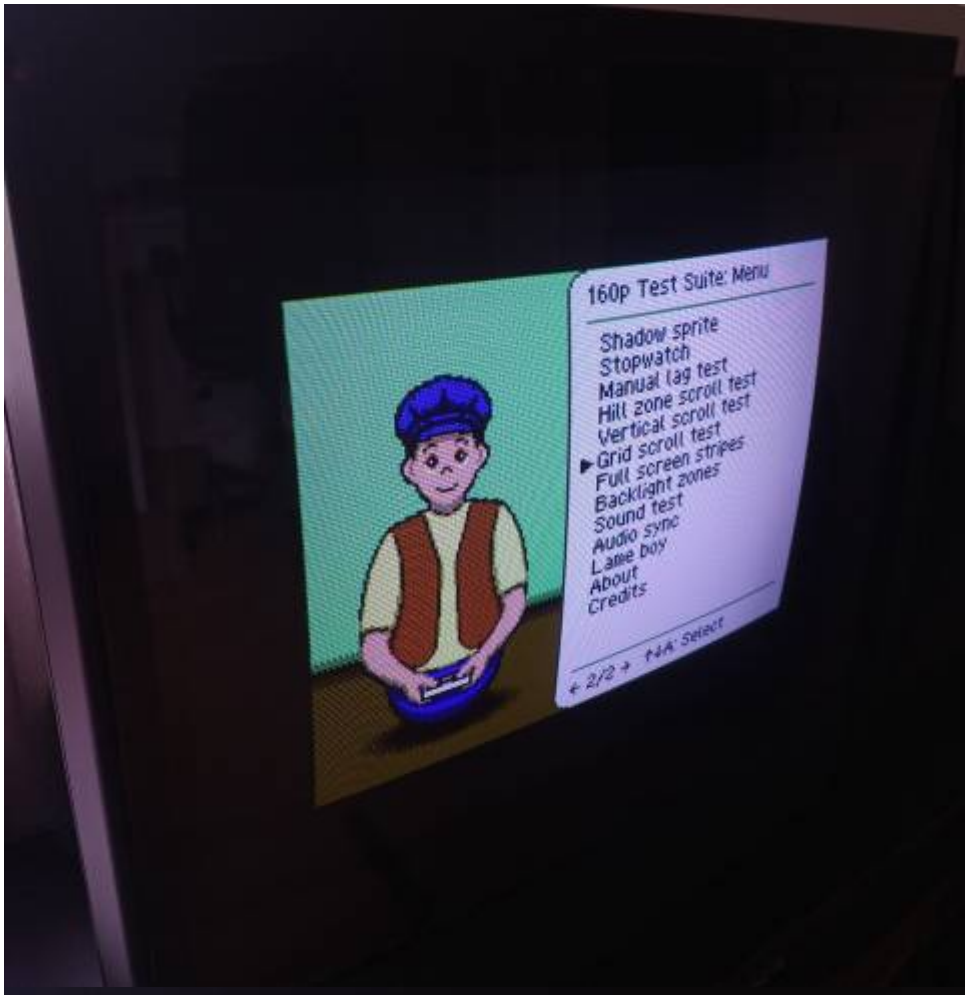
```
aspect_ratio_index = "#"
custom_viewport_height = "#"
custom_viewport_width = "#"
video_scale_integer = "true"
```

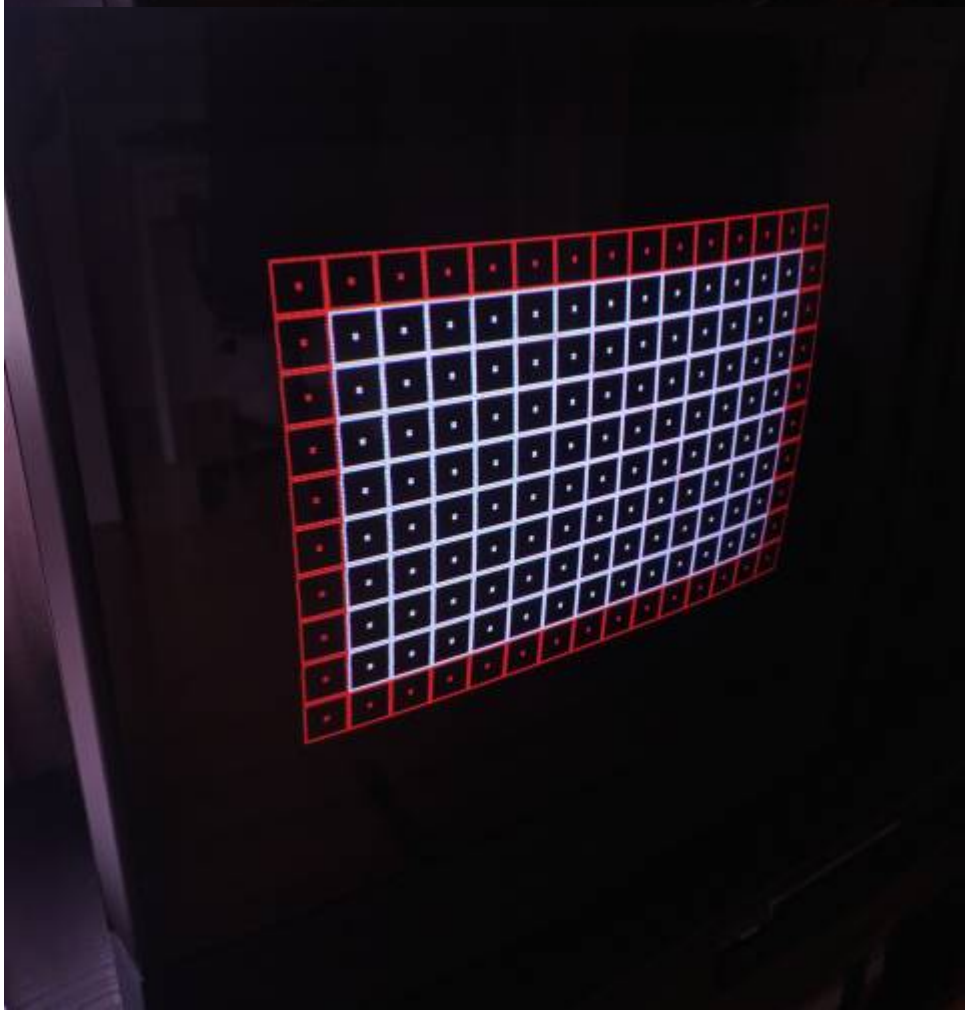
where the custom viewport's resolution is that of the native resolution of our emulated system. For example, this would be the resulting `mGBA.cfg` file for the Game Boy Advance:

mGBA.cfg

```
aspect_ratio_index = "23"
custom_viewport_height = "160"
custom_viewport_width = "240"
video_scale_integer = "true"
```

The result of doing this is the Game Boy Advance screen being integer scaled in the center of our CRT screen with pixel perfect accuracy:


160p Test Suite: Menu
Shadow sprite
Stopwatch
Manual lag test
Hill zone scroll test
Vertical scroll test
▶Grid scroll test
Full screen stripes
Backlight zones
Sound test
Audio sync
Lame boy
About
Credits
← 2/2 → +A: Select



More information on how use core and directory overrides can be found at Libretro's official documentation (keep in mind the paths are different for Batocera).

Global Libretro Switchres configuration

Be sure you have configured your switchres.ini before going any further.

Configure Libretro CRT Switchres in RetroArch

Choose any libretro core and game inside Batocera using your gamepad or keyboard.

Example: Core: Atari 2600 Game: H.E.R.O.

Access the menu and configure (in this order):

```
Main Menu -- Video -- CRT SwitchRes -- Use High Resolution Menu (Optional)
Main Menu -- Video -- CRT SwitchRes -- CRT Super Resolution (Native)
Main Menu -- Video -- CRT SwitchRes -- CRT SwitchRes [INI]
```

It should change to 240p/480i

```
Main Menu -- Settings -- Configuration -- Save Configuration on Quit [ON]
```

Finally, save the configuration by exiting back to Emulationstation.

Per core overrides

In case a particular core/content needs to use a particular Switchres without disrupting other cores/content. Choose any Libretro core and game inside Batocera using your gamepad or keyboard.

Example: Core: Atari 2600 Game: H.E.R.O.

Access the menu and configure (in this order):

```
Main Menu -- Settings -- Configuration -- Save Configuration on Quit [OFF]
Main Menu -- Settings -- Configuration -- Use Global Core Option Files [OFF]
Main Menu -- Video -- CRT SwitchRes -- Use High Resolution Menu (Optional)
Main Menu -- Video -- CRT SwitchRes -- CRT Super Resolution (Native)
Main Menu -- Video -- CRT SwitchRes -- CRT SwitchRes [INI]
```

It should change to 240p/480i

Finally, save the configuration for this core:

```
Quick Menu -- Overrides -- Save Core overrides
```

This can be done with any content run in a Libretro core.

Troubleshooting

Should anything go wrong during the configuration you can always delete your Core setting for a specific core in the directory

```
/userdata/system/.config/retroarch/config/"Core_Name"
```

You can also complete remove the directory and start again (Only remove the `retroarch` folder)

```
/userdata/system/.config/
```

Same goes for (Only remove the `retroarch` folder)

```
/userdata/system/configs/
```

Or reset/delete everything using the development tool `batocera-es-swissknife` (reboot to get a clean config)

```
batocera-es-swissknife --reset-ra
```

Creating you own Boot Resolution with Switchres

Let's say you have a PC CRT monitor and like to add the boot resolution 1024×768@60hz. This can be done by using a custom **Extended Display Identification Data (EDID)**.

This will use the Monitor preset `pc_31_120` **Pc Crt 31-120hz** at 1024×768@60hz

```
switchres 1024 768 60 -m pc_31_120 -e
```

The file will be named `pc_31_120.bin`.

- Create the folder `mkdir /lib/firmware/edid/`
- Move the file `mv pc_31_120.bin /lib/firmware/edid/`

1. Search for the `syslinux` file
   - If legacy, the file will be at `/boot/syslinux.cfg` or `/boot/boot/syslinux.cfg`
   - If UEFI, the file will be at `/boot/EFI/syslinux.cfg` or `/boot/EFI/BOOT/syslinux.cfg`
2. Replace the boot resolution with your edid file `video=`**`YourCardOutput`**`:e drm.edid_firmware=`**`YourCardOutput`**`:edid/pc_31_120.bin`
   - In our exemple :

```
    APPEND label=BATOCERA console=tty3 quiet loglevel=0
vt.global_cursor_default=0 mitigations=off video=DVI-I-1:e
drm.edid_firmware=DVI-I-1:edid/pc_31_120.bin
```

Configure GroovyMame for CRT display

By default, GroovyMame's menus are designed for use on a HDTV, and the text is illegible on our CRT:

Some settings need to be adjusted from their defaults to fix this. First we need to generate the default configuration files and move them to the userdata partition:

1. Go to the MAME folder using the command `cd /usr/bin/mame`
2. Generate the ini files needed by running the MAME script with the following command: `./mame -cc`
    - This will create the following files

```
/usr/
 └─bin/
   └─mame
     ├─mame.ini
     ├─plugin.ini
     └─ui.ini
```

3. Create a new folder in `/userdata/system/` named `.mame` using the command `mkdir /userdata/system/.mame/`
4. Move the files `mv /usr/bin/mame/*.ini /userdata/system/.mame/`

Now let's configure those files.

mame.ini

1. Open `/userdata/system/.mame/mame.ini` with your preferred text editor. If still in SSH you can use `nano /userdata/system/.mame/mame.ini` or if exploring the share on Windows

you can use Notepad++ for example.

2. Under `# CORE SEARCH PATH OPTIONS`, change

```
fontpath                  .
```

to

```
fontpath                  /usr/share/fonts/TTF/
```

This will allow you to use all the cool and snazzy fonts that Batocera has available to MAME!

We only care about one though.

3. Under `# CORE MISC OPTIONS`, change

```
uifont                    default
```

to

```
uifont                    uismall.bdf
```

This switches to the legible font for menus.

4. Under `# OSD FULL SCREEN OPTIONS`, change

```
modesetting               0
```

to

```
modesetting               1
```

This will force the use of Switchres.

5. Save and close the file. `[Ctrl]+[S]` to save.

ui.ini

1. Open `/userdata/system/.mame/ui.ini` with your preffered text editor.
2. Under `# UI OPTIONS`, change

```
infos_text_size           0.75
```

to

```
infos_text_size           1.00
```

This will **embiggen** the text.

3. Also in `# UI OPTIONS`, change

```
font_rows                 30
```

to

```
font_rows                 19
```

This will prevent our now larger text from rolling off the screen!

4. (Optional) Change the following line

```
skip_warnings             0
```

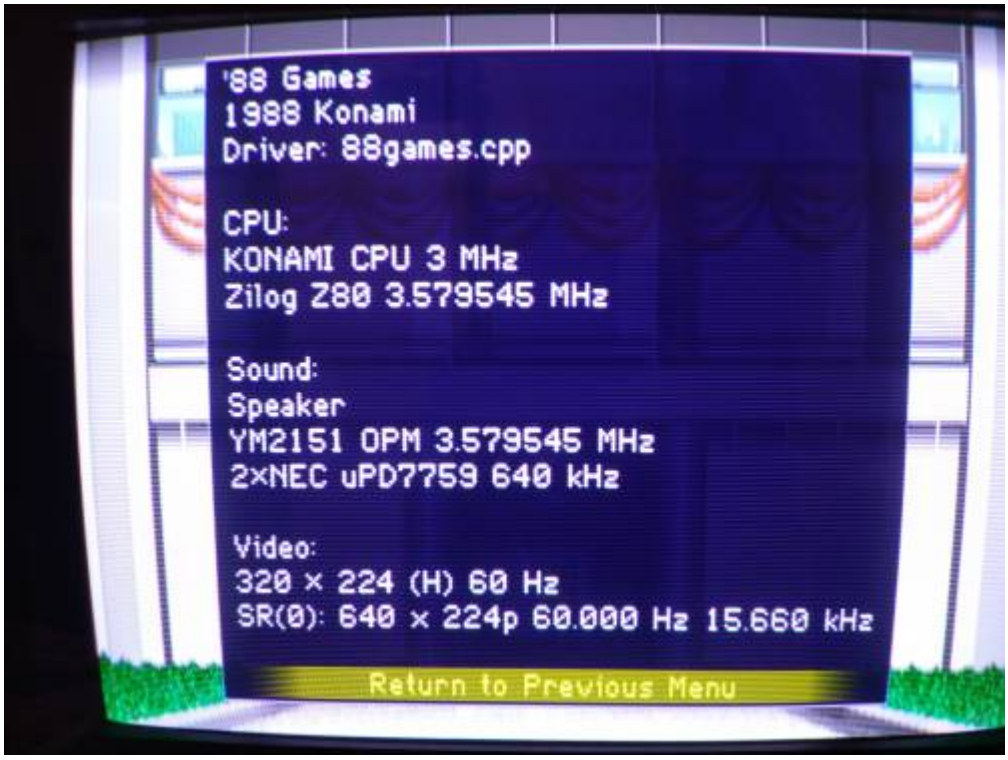
to

```
skip_warnings             1
```

This will get rid of the “there are known problems with this machine” warning from appearing.

If you'd like to double-check your file against one that's already set up (but possibly for an older version of MAME), you can check it out on this Google Drive link.

Now when we launch GroovyMame and go into its menus, it's a lot clearer on our CRT.



Let's choose GroovyMame as the default emulator for all MAME ROMs in Batocera and set a few other things too:

In EmulationStation, navigate to the MAME system and enter it. Press [SELECT] to open the quick menu and navigate to **ADVANCED SYSTEM OPTIONS** and set the following options:

- **EMULATOR**: MAME
- **VIDEO MODE**: AUTO
- **DECORATION**: NONE
- **GRAPHICS BACKEND**: AUTO
- **BGFX BACKEND**: AUTO
- **BGFX VIDEO FILTER**: AUTO
- **CRT SWITCHRES**: ON
- **TATE MODE**: AUTO (Change if intending to rotate your CRT display to play vertical TATE MODE games)

Advanced Miscellaneous Emulator Configuration

Dolphin - scale image to window size -

By default dolphins aspect ratio for 4:3 is set to `AspectRatio = 2` to force the aspect ratio to 4:3, but this don't actually fill the screen on a CRT and will sometimes leave black borders on the left and right sides. In Batocera **v33** and above, this can be solved by simply setting the aspect ratio for GameCube/Wii systems (in **ADVANCED SYSTEM SETTINGS**) to “FULL”.

In Batocera **v32** and lower, we need to edit the file `dolphinGenerator.py` and change the value for Aspect 4:3 to `AspectRatio = 3` to set the aspect ratio to “Stretch to window”.

Navigate to

```
/usr/lib/python3.9/site-packages/configgen/generators/dolphin/
```

Edit

```
dolphinGenerator.py
```

Find the line

```
# Ratio
def getGfxRatioFromConfig(config, gameResolution):
    # 2: 4:3 ; 1: 16:9  ; 0: auto
    if "ratio" in config:
        if config["ratio"] == "4/3":
            return 2
        if config["ratio"] == "16/9":
            return 1
    return 0
```

Change to

```
# Ratio
def getGfxRatioFromConfig(config, gameResolution):
```

```
    # 3: 4:3 ; 1: 16:9  ; 0: auto
    if "ratio" in config:
        if config["ratio"] == "4/3":
            return 3
        if config["ratio"] == "16/9":
            return 1
    return 0
```

Finish with

```
batocera-save-overlay
```

Set Core Aspect ratio to 4:3.

Dolphin - Turn off onscreen notifications -

From

```
# PanicHandlers displaymessages
dolphinSettings.set("Interface", "UsePanicHandlers",        '"False"')
dolphinSettings.set("Interface", "OnScreenDisplayMessages", '"True"')
```

To

```
# PanicHandlers displaymessages
dolphinSettings.set("Interface", "UsePanicHandlers",        '"False"')
dolphinSettings.set("Interface", "OnScreenDisplayMessages", '"False"')
```

Finish with

```
batocera-save-overlay
```

Libretro, turn off force notification messages

This behavior has since been fixed, but this section will remain here for people still having issues with this.

Batocera's config generation scripts may overwrite your preferences set in `batocera.conf`. This can be fixed by editing the config generator scripts.

Navigate to the directory

```
/usr/lib/python3.9/site-packages/configgen/generators/libretro/
```

Edit the file `libretroConfig.py`

Find the line

```
retroarchConfig['video_font_enable'] = '"true"'
```

Change to

```
retroarchConfig['video_font_enable'] = '"false"'
```

Finish with

```
batocera-save-overlay
```

From:
https://wiki.batocera.org/ - **Batocera.linux - Wiki**

Permanent link:
https://wiki.batocera.org/batocera-and-crt?rev=1690720755

Last update: **2023/07/30 12:39**