adafruit learning system

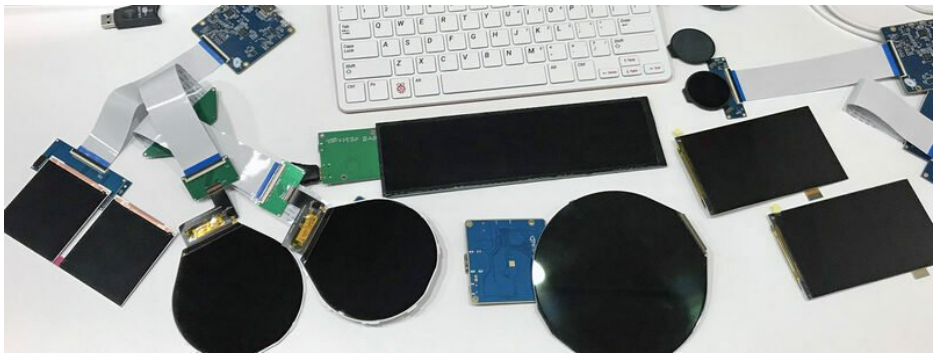
Using Weird Displays with Raspberry Pi

Created by Phillip Burgess

Last updated on 2020-12-28 08:54:30 PM EST

Guide Contents

Overview



You read that right. Not *wired.* **WEIRD.**

We've seen a proliferation of novel **graphical displays** in consumer devices lately — tailor-made for some specialized task, or just to stand out among competitors. The arrival of international commerce sites like AliExpress lets anyone source parts directly from display manufacturers. Hobbyists and tech-focused artists, who once had to settle for pulling interesting parts from years-old junk, now get access to new display technology just as consumer electronics manufacturers are themselves ramping up. It's pretty dang exciting.

Another staple of inventive projects, the **Raspberry Pi** computer, can often be paired up with these peculiar displays *if* you know the right spells. You're no longer limited to the usual 16:9 or 4:3 rectangle — tall, squat, square or round displays are among the many choices out there now.

It's not all latest-and-greatest though. This guide covers some old-school varieties as well — they're often useful or interesting, but might still need a little help.

Paired correctly, anything that could normally be done on a Raspberry Pi monitor — MP4 **video playback**, OpenGL **3D graphics**, custom **PyGame** code or other special effects — can all be presented on these unique screens at high resolution and buttery smooth full video speeds.

Important Disclaimers

Adafruit provides technical support only for the items we sell. While anyone is welcome to discuss and share findings in the Adafruit Forums (https://adafru.it/jlf), we won't always have answers for unfamiliar hardware sourced elsewhere.

There is **no guarantee** that these methods will work for all displays — you are venture capitalist of *Your Own Weird Project* and assume the associated financial risk. Some of these screens are pricey and many can't be returned, so please don't over-extend yourself financially.

This guide is not an endorsement of any manufacturer or seller.

Display Types

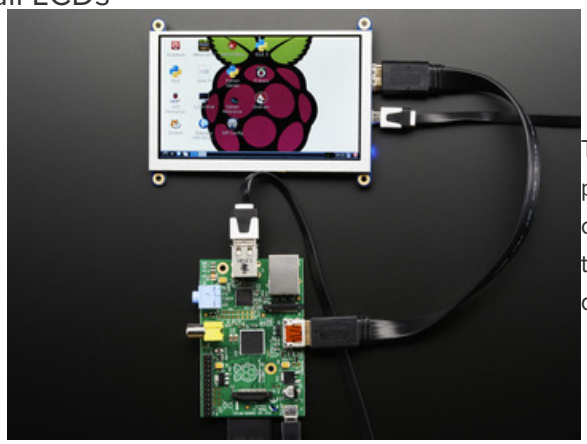
A common theme to the displays covered in this guide is that they all connect to an **HDMI output** on the Raspberry Pi. These tips **do not apply** to SPI or PiTFT displays (using the Pi's GPIO header) or DPI (Display Parallel Interface) displays (such as Pimoroni HyperPixel or Raspberry Pi Touch Display), which have their own setup methods or installers.

Some displays feature **touch** input, but that's a whole extra layer of complexity and is **not covered** in this guide. We're focusing strictly on getting a usable **image**.

Self-Contained Monitors

This first group are devices *intended* as computer or video displays, not repurposed from some other application...

Small LCDs

- 

These screens are sometimes seen in car entertainment or for portable video setups. Usually 5 to 10 inches diagonally and only rarely full HD resolution. You might find them as a ready-to-use product in a slick plastic housing, or as bare display, driver and button-control boards.

In Raspberry Pi circles these are often used in portable gaming projects or small task-focused systems like 3D print servers.

Pros: relatively inexpensive, and any of the ones in the Adafruit shop (https://adafru.it/PnE) are *known* compatible with Raspberry Pi.

Cons: older screens that do not specifically mention "**IPS**" (in-plane switching) may have poor contrast and viewing angles.

DVI Monitors

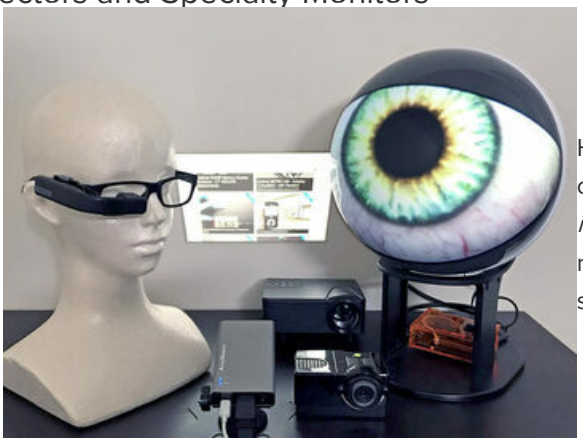
- 


Prior to the ubiquity of 16:9 aspect ratio displays with HDMI connectors, an earlier generation of computer monitors used a ***DVI*** connection. Seldom seen now, you might still find these at flea markets or thrift shops.

Pros: often super cheap, like $10. Usually plug-and-play with Raspberry Pi. Late models — 20" with a 4:3 aspect ratio — are an ideal size for retro arcade cabinet builds!

Cons: Requires **HDMI-to-DVI adapter** or cable. Brightness and contrast are inferior to later screens, though usually adequate. Finding one is a game of chance.

Projectors and Specialty Monitors

- 
Highly weird video projectors, head-mounted displays and other esoteric varieties — *as long as they include an HDMI input* — are often plug-and play. But some are picky about resolution and may demand specific video settings. The subsequent pages of this guide may help.

Analog Televisions

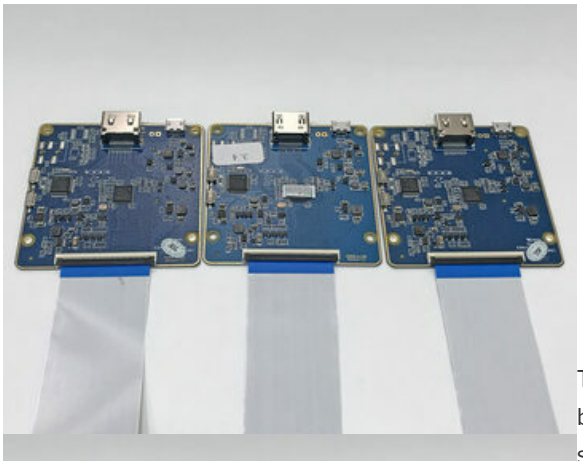
- 

  Vintage televisions can be connected to Raspberry Pi if you're after a specific look...but this requires some special doodads. This is the topic of a separate guide (https://adafru.it/PnF) and won't be covered here.

"Raw" Displays

The weirdest of the weird, these are the cutting-edge variety you might find direct from overseas companies, often designed with some esoteric application in mind, but can be repurposed for Raspberry Pi use. None of these is especially cheap...all cost more than the Pi itself. If you're after a distinct look or special effect, it may be a reasonable gamble.

Many of these currently seem to trace back to a Shenzhen company called *Wisecoco*. Just a data point, not an endorsement, something to search when exploring AliExpress and elsewhere. There are others.

- 

  These displays *must* be paired with an **HDMI** adapter/driver board - they are not SPI devices. When buying, you may see separate options for a bare display (or two) and display-plus-HDMI-adapter for a bit extra, so make sure you get the latter.

- 

If you find yourself with a variety of weird displays, **keep the screens and their adapters together** — and label them! They are not interchangeable. The adapters may look identical but may have unique firmware for the raw display they're meant to drive.

These HDMI adapters are comparable in size to a Raspberry Pi Model A or a HAT board. This is **only for size reference**, they are ***not*** HAT boards nor even Raspberry Pi specific (they'll usually work with PCs and Macs and other HDMI sources too). So they don't neatly stack on a Pi, unfortunately. The adapter/driver will usually feature a full-size HDMI input, USB micro-B connector for power, and an FPC connector for a ribbon cable to the screen(s). Then there's an adapter from the FPC cable to the display, sometimes called a Hirose connector or just "motherboard connector." Each display's will be different.

The **connectors** on these screens and driver boards are typically rated for only a **couple dozen** mating cycles, unlike HDMI plugs which are near indestructible. So try to set them up once and keep them connected.

Also, while you might find longer FPC ribbon cables, you will ***never*** find extension cables for the tiny Hirose connectors on the displays themselves! Most two-screen setups will have two connectors on one PCB, limiting how the screens can be positioned, but some feature a splitter to two FPC cables and Hirose adapters...go for that if it's available, even if it costs a little extra.

Round Displays

- 

Seen in smart watches, thermostats, VR headsets, car gauges and internet appliances that want to stand out. Early ones had enormous bezels, but the latest displays go nearly edge-to-edge!

These present themselves to the Pi as a square or rectangular framebuffer. Pixels outside the circle are simply clipped from view.

Project ideas might include robot eyes, cutting-edge cosplay and props, that sort of thing. The shape still catches folks off guard and is a little "magic."

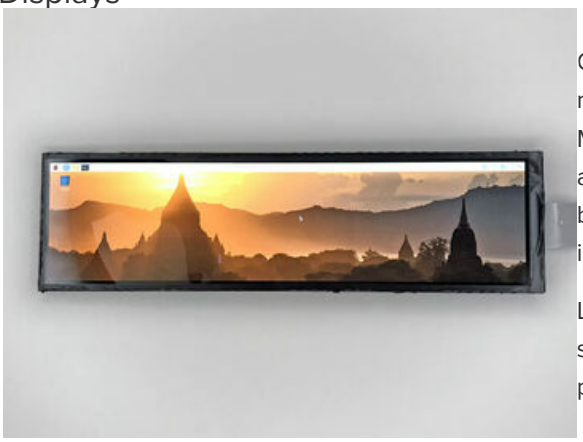
Square Displays

- 

With a high pixel density for their size, and driven in pairs, these are likely intended for virtual reality headsets.

The corresponding HDMI adapter that came with this set can drive *two* of these screens, and a Pi 4 has two HDMI outputs... sadly, that leaves us a couple squares short of a full video cube.

Bar Displays

- 

Originally for automotive use (backup cameras, rear view mirror, instrumentation) or rack-mount servers (system status). Many variants, from little-wider-than-16:9 to incredibly long and thin ribbons. Some include touch input, though there may be no software support on the Pi, depending on implementation.

Larger ones are sometimes used in retro arcade projects, showing marquee graphics for whatever game is currently playing.

I can't help but picture these bar displays for a retro-future "cyberdeck" computer. Together with a Raspberry Pi 400 and a battery pack, the combination would be reminiscent of the 1980s Tandy Model 100, but with supercomputer powers.

It would be neat to see the MacBook Pro OLED Touch Bar here, but have yet to see any sort of standardized interface adapter for that one.

High DPI or "Retina" Displays

- 

  Ultra high resolution screens as on present-generation smartphones and tablets.

  This particular two-screen setup did not work on Raspberry Pi...it was glitchy at best. This *won't* be the case with all of them...others may work fine, and this setup does in fact work with a Windows PC. It's important to illustrate that this is sometimes a **gamble**. It's Wild West stuff. Returns to overseas companies are rarely practical, shipping may exceed the cost of the hardware...so you may be stuck with it.

Interocitor

- 

  We've yet to find a triangular Interocitor screen. It's just *too weird,* you hear me? TOO WEIRD.

  Image credit: Universal Pictures, *This Island Earth*

Plug and Play (Hardware Scaling)

Most of the “Monitor” (not “Raw”) devices on the prior page — pretty much anything packaged for consumer use — can work directly with any Raspberry Pi with an HDMI port (including mini or micro HDMI). They’ll either downsample the Pi’s default 1080p resolution, or the two can negotiate a mutually-compatible resolution. Super easy.

When downsampled — when the negotiated resolution doesn’t match the LCD’s native pixel count — the results can be disappointing. Graphics will appear blurry, the aspect ratio might be stretched, and user interface elements like menus and buttons are small and illegible. If this happens, the next pages may be helpful in getting the Pi to generate a pixel-perfect image.

This is *especially* common on low-cost projectors. They’ll boast that they accept “full HD 1920x1080 input”…but then downsample this to 640x480 or *even less* sometimes.

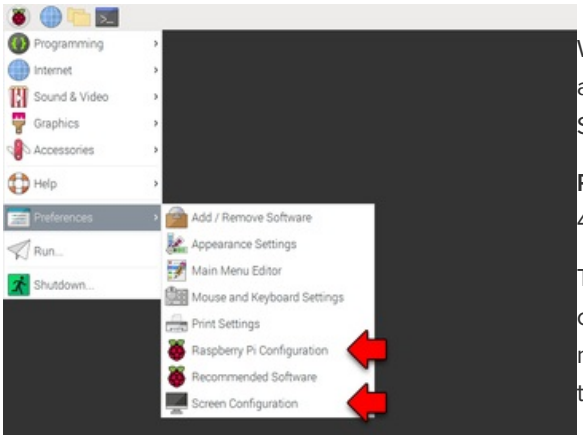
It’s helpful to know the display’s native pixel resolution. This can usually be found alongside other specifications in the back of the manual somewhere, if one is provided. But if not, there are ways to sniff this out…

Pi 4 or 400 + Desktop + Luck

Connecting weird displays used to be an ordeal. But, if you're using a **Raspberry Pi 4** or **400**, paired with the **"desktop" Raspberry Pi OS** (*not* the "Lite" variant)...and with a little bit of **luck**...everything is *much* simpler now. Quite a few of these displays are now plug-and-play! A stable and usable resolution is often negotiated, and only minor adjustments may be needed.

This isn't *always* the case though. If you find a display that won't work automatically, the following explains how to work things out **manually**.

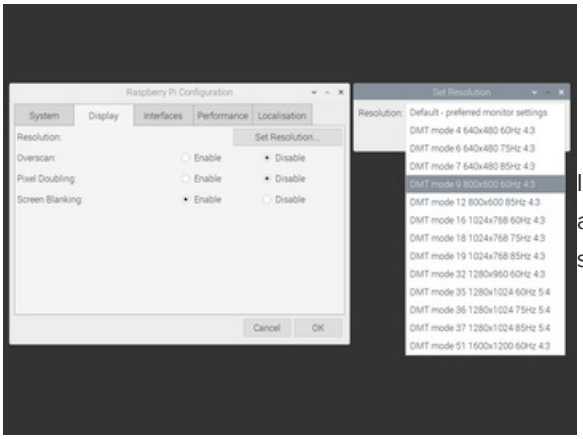
The Raspberry Pi 4 and 400 have **two HDMI outputs**. If two displays are connected, and if *either one* fails to sync, often that will spoil the party for *both*...you'll get no video to either screen. When trying out new displays, connect just **one at a time** and reboot. For any that won't sync, follow the steps on the next page. Once resolved, you can configure distinct settings for each HDMI port (also explained there).

- 

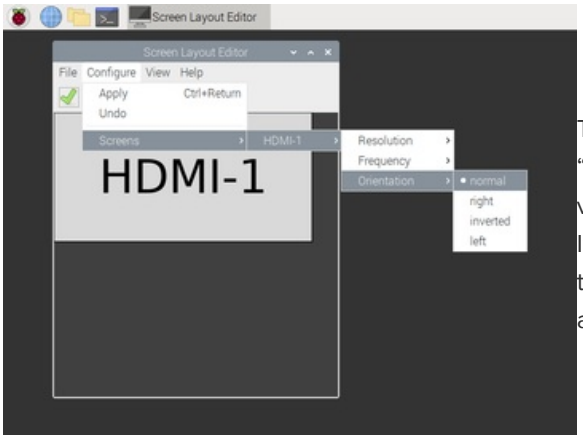
With a display working for the first time, you might need to adjust some stuff like rotation or turning off the black border. Settings for these can be found under the Pi (system) menu...

Pi→Preferences→Raspberry Pi Configuration and (on Pi 4 or 400) **Screen Configuration**

The Pi menu can't be seen on round displays because it's clipped off the edge, but can still be accessed by moving the mouse all the way up and left and clicking. The pixels are there, you just can't see them.

- 

In the **Raspberry Pi Configuration** tool, click the **Display** tab to access overscan (border) settings, and — if the screen supports different modes — configure the display resolution.

- 

The **Screen Configuration** tool lets you change which way is "up." This is handy if you want to use an old monitor in a vertical "portrait" orientation — playing Dig Dug, for example. If you have multiple displays connected, you can also inform the system of their physical arrangement here, so the mouse and windows move between them more naturally.

As mentioned above, a handful of displays still won't play nice. Or you might have a need for the "Lite" Raspberry Pi OS, or a third-party operating system. In these situations you can still get in there and do it The Old-Fashioned Way.

Everything Else

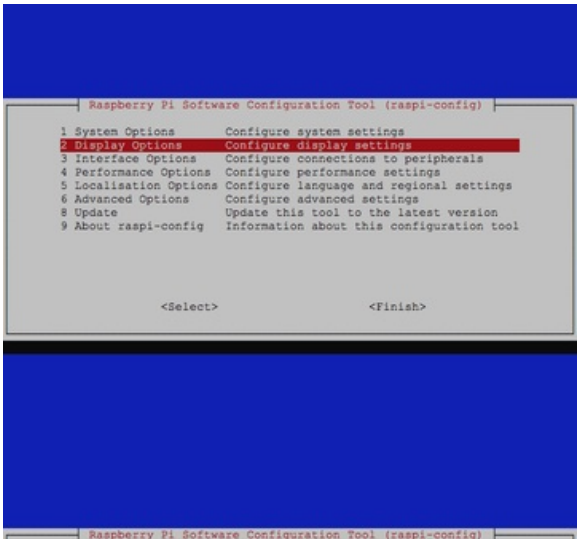
If your project has the luxury of using the desktop OS on a Raspberry Pi 4 (or 400), do it! Interfacing weird displays is *so much simpler.* If you *must* use an earlier Pi, or the command-line "Lite" OS, or just have a cantankerous display that even the Pi 4 won't negotiate with, this is how it's done...

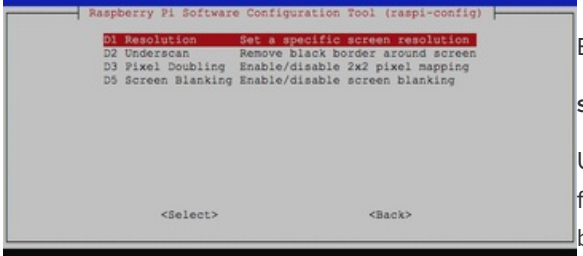
Things Needed

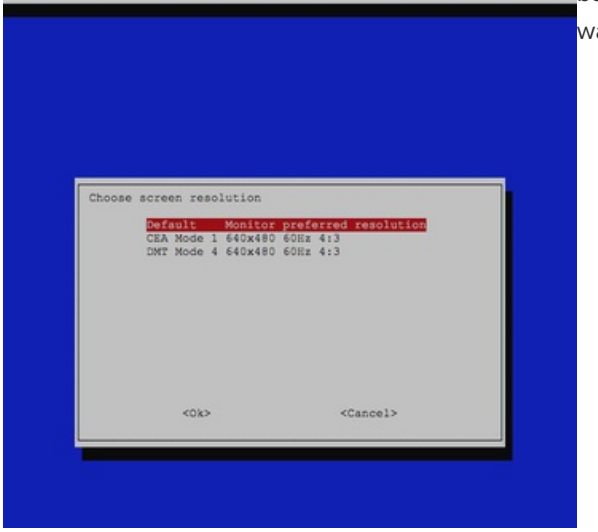
You'll need the Raspberry Pi system booting and on a network and/or *temporarily* connected to a known-compatible monitor. We'll assume some familiarity with Pi setup and file editing here...it's covered in other tutorials.

- With a network connection — either **Ethernet** or **WiFi** — you wouldn't need the temporary monitor, instead you'd log in remotely and can copy-and-paste commands from this page. This is usually best. **ssh** (allowing remote login) can be enabled by creating an empty **/boot/ssh** file on the SD card.
- With a compatible **monitor** and **keyboard** connected, you can follow the steps outlined here and might not need a network connection. There's no software to install, just some system configuration. You may be connecting either the compatible monitor or weird display between reboots.
- Another option is to insert the micro SD card *in a different computer* (e.g. Windows, Mac or Linux) and edit the video configuration there, then move the card to the Pi and try booting. The card's **/boot** partition will mount on most computers and files can be edited.

There will be some **command-line invocations** and some **editing of files** with your text editor of preference. And you will likely be **rebooting** several times.

- 

- 

- 

Before going down a technological rat hole, try...

sudo raspi-config

Under **Display Options** is a **Resolution** setting, and you might find a compatible option there. Or you might just get kicked back to the main menu, in which case we'll do this the manual way...

Basic Configurating

All of the setup is done in the file **/boot/config.txt** and (if editing on Pi) must be done as **root**, e.g.:

sudo nano /boot/config.txt

The default **config.txt** file has a number of HDMI-related settings peppered throughout, most of them commented out (preceded with a **#** character). I like to collect these all in a group, usually at the end of the file, so they're easier to find and work with while ironing out any problems.

These **four lines** are the **minimum** starting requirement. They should all be **enabled** (remove any leading **#** character) and the values should match what's shown here:

```
hdmi_force_hotplug=1
hdmi_group=2
hdmi_mode=87
hdmi_force_mode=1
```

`hdmi_force_hotplug=1` tells the Pi to output a signal even if an HDMI display is not detected. Since these displays are *weird,* this just makes extra sure *something* is coming out of the Pi, whether it likes the display or not.

`hdmi_group=2` with `hdmi_mode=87` enables custom video configuration, which we'll set up shortly...it tells the system that we might not be using a common resolution like 1080p. And think of `hdmi_force_mode=1` as saying "no, I really mean it."

Then there's some optional additions...

To disable the black border around the screen, use:

```
disable_overscan=1
```

A few displays *require* the border, in which case set this to `0`. Usually it's television screens being repurposed as computer monitors, but a few odd items like the Vufine wearable display seem to prefer it.

If using a **DVI** or **HDMI monitor** and it *almost* works but flashes on and off, try:

```
config_hdmi_boost=7
```

It's ignored by the Raspberry Pi 4 and 400, but other boards can use this to generate a stronger video signal. The default is normally `2` or `5` on some Pi models. `7` is usually an ample boost. It can go as high as `11` but that's not recommended unless you're seeing issues with very long cables. Work up incrementally (rebooting with each change) to find the *lowest reliable value.*

If Pi and monitor aren't negotiating capabilities correctly...yet it's a consumer monitor that you'd *expect* to be plug-and-play...try the **boot_delay** setting, which pauses the system for a number of seconds before loading the kernel. This can give some monitors the ready time they need if everything's powered on at once:

```
boot_delay=5
```

The value (5 in this example) is the number of seconds to wait before booting. If that makes the two devices behave in a plug-and-play manner, all this other stuff might be unnecessary!

Finally, this setting routes audio to the HDMI port if the display supports it (some monitors have built-in speakers)...so it's not strictly *video* related, but may as well go in the same group:

```
hdmi_drive=2
```

None of this actually configures the display resolution yet, we're just setting the stage.

Getting the Pi and display to actually sync up involves some trial and error. That's why this is best done through a remote **ssh** session, or editing **config.txt** on the SD card in another system.

Also, none of these changes take effect until the system is **rebooted**. Writing to the **config.txt** file has no immediate effect; you must **reboot each time.**

Try hdmi_cvt First

CVT stands for the "Coordinated Video Timing" VESA standard. This is the easier way of defining a custom video mode...it seems to work more often than not, but might not be sufficient for all displays. It's worth a try.

You will need: the display's native pixel resolution and refresh rate. If unsure about refresh rate, 60 Hz is usually a reasonable guess.

Add a line like the following to **/boot/config.txt**, replacing the `[tags]` with suitable numbers (explained below):

```
hdmi_cvt=[width] [height] [framerate] [aspect] [margins] [interlace] [rb]
```

For example:

```
hdmi_cvt=800 400 60 6 0 0 0
```

or sometimes just:

```
hdmi_cvt=800 400 60
```

The first three values are **required**:

`[width]` and `[height]` are the display's exact native pixel dimensions.

`[framerate]` is the display's refresh rate, in Hertz...often `60`, but occasionally higher or lower values.

The remaining four values are **optional**...if unspecified, defaults will be used:

`[aspect]` is a value from `1` to `6` indicating the nearest width-to-height aspect ratio:

- `1` = 4:3
- `2` = 14:9
- `3` = 16:9
- `4` = 5:4
- `5` = 16:10
- `6` = 15:9

There doesn't seem to be much harm in getting this wrong. Most of the "raw" displays I've experimented with are either 1:1 or 2:1, and using values of `4` and `3` respectively — or just leaving it off — has worked just fine. Default if unspecified is `3`.

`[margins]`, with a value of `0` (disabled, the default) or `1` (enabled), seems to be the inverse of the config.txt's `disable_overscan` setting. The latter seems to take precedence and overrides any setting here, I think.

`[interlace]` states whether the display is progressive scan (`0`, the default) or interlaced (`1`). Usually everything is progressive scan nowadays.

`[rb]` is for “reduced blanking,” a way of saving video signal bandwidth on LCDs. `1` enables reduced blanking, while `0` (the default) disables it.

hdmi_timings If Necessary

If `hdmi_cvt` doesn’t get the Pi and display talking, there’s an even more fiddly level of control possible, where every aspect of the video signal is specified. Fortunately there is a **command-line tool** to help us puzzle this out. With the display (and adapter/driver board, if present) connected to the Pi and powered on, run the following command:

```
/opt/vc/bin/tvservice -d edid.dat; /opt/vc/bin/edidparser edid.dat
```

This will output a *ton* of data, so hopefully you’re using a terminal with scrollback capability…or you can redirect the output to a file, or pipe it through the *more* command.

The above command probes the display for *every compatible resolution setting…*sometimes several times for each. You’ll get something like the following that just scrolls on for pages…

```
HDMI:EDID found monitor S/N descriptor tag 0xff
HDMI:EDID found monitor range descriptor tag 0xfd
HDMI:EDID monitor range offsets: V min=0, V max=0, H min=0, H max=0
HDMI:EDID monitor range: vertical is 23-75 Hz, horizontal is 15-240 kHz, max pixel clock is 340 MHz
HDMI:EDID monitor range does not support GTF
HDMI:EDID failed to find a matching detail format for 800x400p hfp:400 hs:20 hbp:400 vfp:20 vs:4 vbp:12 pixel cloc
:42 MHz
HDMI:EDID calculated refresh rate is 60 Hz
HDMI:EDID guessing the format to be 800x400p @60 Hz
HDMI:EDID found unknown detail timing format: 800x400p hfp:400 hs:20 hbp:400 vfp:20 vs:4 vbp:12 pixel clock:42 MH
z
HDMI:EDID established timing I/II bytes are 00 00 00
HDMI:EDID standard timings block x 8: 0x0000 0000 0000 0000 0000 0000 0000 0000
HDMI:EDID parsing v3 CEA extension 0
HDMI:EDID monitor support - underscan IT formats:no, basic audio:yes, yuv444:yes, yuv422:yes, #native DTD:4
```

You’ll want to scroll through this looking for any mentions of a resolution that best matches the display hardware. In the example above, it’s an 800x400 pixel framebuffer that feeds two 400x400 round displays…a truly Weird Display.

If you’re using one of the weird “raw” displays like this, you’ll also see something like the following toward the end of the list, referencing a 640x480 pixel resolution:

```
HDMI:EDID adding mandatory support for DMT (4) 640x480p @ 60Hz
HDMI:EDID adding mandatory support for CEA (1) 640x480p @ 60Hz
HDMI:EDID best score mode initialised to CEA (1) 640x480p @ 60 Hz with pixel clock 20 MHz (score 25)
HDMI:EDID best score mode is now CEA (1) 640x480p @ 60 Hz with pixel clock 25 MHz (score 61864)
HDMI:EDID DMT mode (4) 640x480p @ 60 Hz with pixel clock 25 MHz has a score of 18432
HDMI0:EDID preferred mode is updated to CEA (1) 640x480p @ 60 Hz with pixel clock 25200000 Hz
```

This is fake…apparently displays are *required* to report a basic 640x480 pixel mode…but the weird display drivers seldom support it or provide up/down-sampling. So, unless you know that actually *is* the display’s true native resolution, pretend you didn’t see it, focus on the prior data.

For the 800x400 pixel weird display, these are the lines we’re after:

```
HDMI:EDID guessing the format to be 800x400p @60 Hz
HDMI:EDID found unknown detail timing format: 800x400p hfp:400 hs:20 hbp:400 vfp:20 vs:4 vbp:12 pixel clock:42 MHz
```

Copy and paste those lines into a note for later reference, you'll need all those numbers.

Now go back to editing **/boot/config.txt**

Before getting carried away with **hdmi_timings**, try feeding just those first three values from the first line into **hdmi_cvt**...unless you already tried this combination previously.

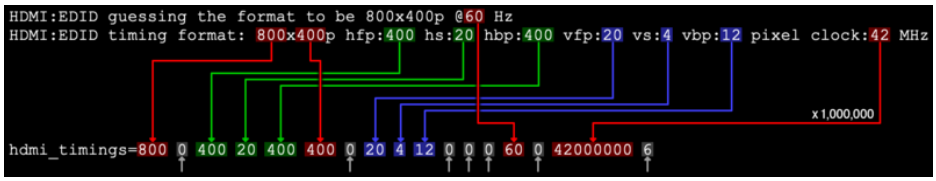
```
hdmi_cvt=800 400 60
```

Reboot and see what happens. If it sticks, great! If not, you'll go back into **config.txt** to specify the full **hdmi_timings**, which expects more arguments than a Monty Python sketch:

```
hdmi_timings=[h_active_pixels] [h_sync_polarity] [h_front_porch] [h_sync_pulse] [h_back_porch] [v_active_lines] [v
sync_polarity] [v_front_porch] [v_sync_pulse] [v_back_porch] [v_sync_offset_a] [v_sync_offset_b] [pixel_rep] [fram
_rate] [interlaced] [pixel_freq] [aspect_ratio]
```

So we can take values shown by the **tvservice** command and plug them into the corresponding spots in the **hdmi_timings** line. The `hfp:`, `hs:` and `hbp:` values shown by tvservice stand for *horizontal front porch, horizontal sync pulse* and *horizontal back porch.* `vfp:`, `vs:`, and `vbp:` are the same for vertical. And you can see there are corresponding spots in the **hdmi_timings** syntax above, but in a different order. A few values are simply defaults.

Rather than try to explain every item...because there's like a billion... let's just illustrate how to reorder the output of the **tvservice** command into a usable **hdmi_timings** line:



The first five values relate to horizontal resolution and timing, second five are for the vertical, and others are for refresh rate and bandwidth (add six zeros), and the rest are sufficiently obscure that you can probably use the default values above.

This example's a little tricky because the horizontal porch values and the vertical resolution are *coincidentally* the same. That won't always be true...others will be different. Don't mix them up.

Be sure to include *every item* and in the *correct order.* If you miss one, or mix them up, the Pi might not boot. You'll have to take the SD card to another computer and fix the file there.

It's rare, but I've encountered at least one display that worked fine with **hdmi_cvt** but not **hdmi_timings**, so don't feel pressured to be "extra 1337" with the latter if the former will do. Also one situation where the Pi 4 auto-detect worked better than any amount of cvt/timings fiddling...apparently making better use of this information, so use that if the situation allows.

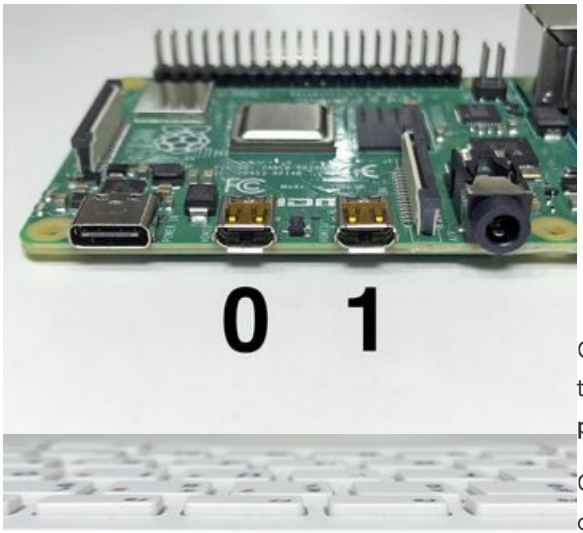
Multiple Displays (Pi 4 and 400)

The Raspberry Pi 4 and 400 have two HDMI outputs. It's possible to specify a unique configuration for

each by appending a `:0` or `:1` to the **hdmI_cvt** and/or **hdmi_timings** commands:

```
# 88mm round displays on HDMI 0
hdmi_cvt:0=1600 800 60 6 0 0 0

# Bar display on HDMI 1
hdmi_timings:1=480 0 30 30 30 1920 0 6 6 6 0 0 0 60 0 66000000 4
```

- 

On the **Raspberry Pi 4**, HDMI outputs 0 and 1 are labeled on the board, but for posterity: HDMI **0** is closer to the **USB-C port**, HDMI **1** closer to the **audio** jack.

- 

On the **Pi 400**, HDMI **0** is closer to the **SD card** slot, HDMI **1** is closer to **USB-C**.

Changing the Screen Orientation

Sometimes a screen will boot up sideways or upside-down from what you'd expect. Or you might just want it aimed differently...for example, some retro arcade games were designed for a vertical "portrait" display, or you might have a bar display that you'd expect would be wide and short, but its default layout is actually tall and thin. Add a **display_hdmi_rotate** line (or **display_rotate** in older versions of the operating system) to **/boot/config.txt** to set this up:

```
display_hdmi_rotate=3
```

With multiple displays, you can rotate them independently by adding `:0` and `:1` as previously shown with **hdmi_cvt** and **hdmi_timings**.

The value passed to **display_hdmi_rotate** works like so:

display_htmi_rotateresult	
0	No rotation (use default for screen)

1	Rotate 90 degrees clockwise
2	Rotate 180 degrees
3	Rotate 90 degrees counterclockwise
0x10000	Horizontal flip
0x20000	Vertical flip

*Flip seems dubious, but consider…many 1980s arcade games had you watching a **reflection** of a vertically flipped screen, either to make the cabinet shallower, or to add background artwork using a Pepper's Ghost effect. This could be useful for a heads-up display or other Weird Ideas.*

With exceptionally wide or tall displays you may need a line or two to override the Pi's notion that it has a 1920x1080 framebuffer…otherwise it won't use the whole display, only a section in the middle.

For example, with the aforementioned "wide" bar display (which is actually a tall display), here's how we change its behavior from 480x1920 to the 1920x480 that we'd prefer, and use the whole screen:

```
display_hdmi_rotate=3
max_framebuffer_width=480
max_framebuffer_height=1920
```

On Pi 4 and 400, some rotation settings are only possible if 3D acceleration is disabled.

Look for this line in **/boot/config.txt**:

```
dtoverlay=vc4-fkms-v3d
```

and add a **#** to comment it out, like so:

```
#dtoverlay=vc4-fkms-v3d
```

Reboot and the screen should be pointed however you'd like, albeit without acceleration. This is another reason why the desktop (rather than lite) OS is preferred if the situation allows…the Screen Configuration tool has a lot of flexibility for things like this.

Anything Else?

If we've overlooked something, here is the official Raspberry Pi documentation **(https://adafru.it/PvD) on config.txt video settings, which goes into even more detail on some of these fiddly settings.**