Raspberry Pi Care and Troubleshooting

Created by Carter Nelson

https://learn.adafruit.com/raspberry-pi-care-and-troubleshooting

Last updated on 2023-08-29 03:55:37 PM EDT
# Table of Contents

**Overview**  
- But first...  

**Expectations**  
- Why is there a Raspberry Pi?  
- What is the Raspberry Pi?  
- With great power...  
- Is a Raspberry Pi an Arduino?  

**General Care**  
- Don't Get It Wet  
- Be Careful of Camera Flashes  
- Don't Short Header Pins  
- Don't Let 5V Touch Other Pins  
- Don't Short 3.3V Pins to Ground  

**WARNING!**  
- DO THIS FIRST!!!  
  - Headless Configuration  
  - Attaching a Monitor  
  - Power Supply  
  - Did It Work?  

**RPI4 Recovery**  
- Sanity Check  
- Create a Recovery SD Card  
- Use Recovery SD in Pi 4  
- Next Steps  

**Troubleshooting**  
- Boot Error Code via LED blink (Pi 4 and Pi 400, but possibly others!)  
- Getting System Information  
- Viewing System Log  
- Is my I2C device there?  
- Make Sure Power Is Good  
- If you're using a (earlier) Pi 4, use the Official Pi Power Supply  
- Make sure USB cable is good.  
- Use an updated operating system image.  
- Use a text editor, not a word processor, for configuration files  

**FAQ**  

**Resources**  
- The Adafruit Forums  
- The Raspberry Pi Foundation  
- GPIO Header Pinouts
Overview

The Raspberry Pi() is an amazing little device. It gives you access to low level hardware interfaces that are normally off limits - does your Mac Book Pro have I2C ports? SPI ports? a CSI camera interface? The Pi does. It is also capable of running the full Linux operating system to provide tons of functionality for accessing and programming these interfaces and the hardware peripherals attached to them.

While the whole idea of the Raspberry Pi is to have fun connecting to and learning about its hardware pins, this is best done with some understanding of exactly what the Raspberry Pi is. There are various ways in which the Raspberry Pi can inadvertently be damaged in the process of using it. This guide will try to cover the most commons ways, so you can steer clear of them. Even if the Raspberry Pi seems to be working fine, other odd issues can arise. This guide will also try to provide some guidance on how to help troubleshoot these.

But first...

...what is this Raspberry Pi thing? What is this little Linux computer that was given away for free on the cover of a magazine? And what should you expect from you adventures into the land of Pi?
Let's start there...

**Expectations**

The intent here is to provide some background on what the Raspberry Pi is and why it came into existence. Knowing what you’re dealing with is always helpful.

**Why is there a Raspberry Pi?**

It is important to remember some of the original motivations that brought the Raspberry Pi into existence. It was not akin to a company like Apple thinking up a cool glittry new product like an iPhone and then making and selling it. Instead, it was created by a relatively small group of people to help solve a perceived problem - the lack of an easily available computing device that people could use to gain direct hands on experience without spending a small fortune.
This led a computer scientist from Cambridge by the name of Eben Upton to try and create such a device. The story spans over many years and involved numerous contributors. It is worth hearing the whole thing:

In a simple quote, the motivation comes down to:

"We need to create a generation of producers not consumers."

Eben Upton, 2013, Wired

Consumers buy and use smartphones. Producers actually make the smartphones - and all the other techy goodness in our future world. There are lots of exciting technical details to learn. Basic electronics, I2C, SPI, and other interfaces, software to drive it all, etc. It's essentially a journey of learning that never ends. Having a relatively cheap and easy to access device helps bootstrap this journey.

A key point in the requirements for the Raspberry Pi was that it cost only $25. This was to make it reasonably cheap to acquire. But also, the creators of the Raspberry Pi knew there was going to be a high potential for damaging the boards in the process of playing around with them. So they also did not want it be a large amount of money lost when the blue smoke monster showed up.

This price point was met. The original Model A was $25.

What is the Raspberry Pi?

It is basically the brains taken out of a smartphone and put on a board that gives you direct access to a lot of the brain's functions. Think of it like this:
The real unique feature of the Raspberry Pi is the 2x20 row of header pins across the top. These are often referred to as the GPIO pins, or GPIO header, for General Purpose Input and Output. Those pins are directly connected to the brain. There's also a little heart - that's what provides power to the brain and everything else. It is also directly connected to the GPIO header in a few places.

So not only do you have direct access to the brain, you also have direct access to the heart. On other devices, like your smartphone, those connections are buried deep inside and you generally do not have access to them.

The Raspberry Pi has its brain and heart completely exposed.

With great power...

There's a bit of a double edged sword to the Raspberry Pi's GPIO header. By giving direct access to the brain, you can do all kinds of amazing things. You can even connect to the heart to help power those things. This is what the Raspberry PI was designed to allow you to do. However, there is no protection mechanism from damaging either the brain or the heart.

Don't let that prevent you from exploring what can be done. Simply understand this as the general nature of the way it is.

Read this guide for some general tips on how to care for your Raspberry Pi. Follow other guides carefully to make sure all connections are correct. And if you do somehow damage your Raspberry Pi, try not to get overly worried about it. It is never fun when this happens. But it is how we learn. And there is no substitute for the direct hands on experience you have gained.
Is a Raspberry Pi an Arduino?

No. But they are similar in that both provide you access to the brain taken out a device. Where the Raspberry Pi brain was taken from something like a smartphone, Arduino brains are more like what you would find in a microwave oven. They are intended for doing small simple tasks really well. With a Raspberry Pi, there are thousands of these kinds of tasks going on in a complex dance.

So why not just use a Raspberry Pi for everything? Pi's cost from $5 to $35 and Arduino's have about the same price range. So what's the dif?

Consider the task of talking to NeoPixels. This is done by sending out a digital signal with a tight timing requirement. The signal has to go high/low/high/low etc. just right for the NeoPixels to work. An Arduino can do that really well, since it has nothing else to do. However, on a Raspberry Pi, there's network stuff going on, maybe a web server is running, oh, and now someone is printing something out, and someone else just logged in remotely, cron jobs to check, now someone started playing an MP3, etc. With all that going on, a Raspberry Pi has a difficult time maintaining the timing requirements for the NeoPixels. Getting NeoPixels to work on a Raspberry Pi has been accomplished, but via a complex work around. Using NeoPixels on an Arduino board is a simple matter of writing and uploading a program.

So...Arduino's are really good for doing one or a few things very precisely. That's important in a lot of applications. Like driving NeoPixels!
General Care

These are basic guide lines to consider while using your Raspberry Pi. By knowing some of the common gotchas, hopefully you won't repeat them yourself.

Don't Get It Wet

Let's start with an fairly obvious and easy one - don't get it wet. Be careful not spill liquids on the board. It won't magically spit out hairballs that turn into more Raspberry Pi's the next day. Sorry. It will most likely destroy the Raspberry Pi. Not what you want.

Be Careful of Camera Flashes
This is an oldy but a goody. As far as we know, it only affects Raspberry Pi Model 2B's, circa 2015. But if you are having your Raspberry Pi reset when exposed to the xenon flash from a camera, then it might be this. Makezine has a good write up on it here:

Is The New Raspberry Pi 2 Camera Shy?

The fix is relatively simple - cover up the U16 component so the light can not reach it. The link above shows where to find U16. A more in depth nerdy write up was done by the Raspberry Pi Foundation and can be found here:

Xenon Death Flash: a free physics lesson

Don't Short Header Pins

This row of header pins on the Raspberry Pi are its signature feature.

Connecting to these pins is an essential part of playing with the Raspberry Pi. However, accidentally "shorting" (directly connecting) these pins together can cause damage. This could happen when trying to connect wires, making measurements, etc.. Using something like a T-Cobbler with a ribbon cable can help make connections easier.

Assembled Pi T-Cobbler Plus - GPIO Breakout
This is the assembled version of the Pi T-Cobbler Plus. It only works with the Raspberry Pi Model Zero, A+, B+, Pi 2, Pi 3 & Pi 4! (Any Pi with 2x20...
https://www.adafruit.com/product/2028
However, it is still easy to accidentally wire things incorrectly on the breadboard and end up shorting pins together.

Note:

- Connecting GND pins together is OK.
- Connecting 3.3V pins together is OK.
- Connecting 5V pins together is OK.

**Don't Let 5V Touch Other Pins**

The Raspberry Pi is a 3.3V device. This means it can not tolerate 5V on any of its pins. Be very careful not to do this, especially since there are two pins that output 5V.

Those two 5V pins are often used to power externally attached hardware, which is fine. You will probably connect to and use these pins. Just make sure that 5V doesn't get connected directly to any of the other GPIO header pins. Same with 5V coming from anywhere else.

**Don't Short 3.3V Pins to Ground**

You power the Pi with 5V via the micro USB connector. So where does this 3.3V come from? It comes from an on board voltage regulator. Be careful not to short these 3.3V pins to ground.

Doing so will very quickly damage the voltage regulator. And since the brain uses that 3.3V, it pretty much kills the Pi.
WARNING!

The Raspberry Pi, unlike your tablet or computer is a bare computer board - essentially like a person with the top of their head missing and the brains exposed. Sure it’s fun to poke and prod, connect hardware and sensors directly to the GPIO pins on the Pi, but those have no protection circuitry on them!

Since pins are exposed it’s really easy for people to damage the Pi by static shock or misconnecting pins.

When using the GPIO pins, never connect the 5V or 3.3V power pins to ground, even for a moment! This will damage the regulator permanently. Also, never connect 5V or higher or less than 0V to any of the GPIO pins! This can also permanently damage the Broadcom processor.

Really, we mean it! Once you connect anything to the GPIO pins or touch them with your hands, the warranty is void! Be mindful when you use the GPIO, or be OK with the possibility you will be replacing the Pi.

DO THIS FIRST!!

Just got your new Pi and excited to get it setup for Octoprint on your 3D printer rig, Kodi for binge streaming, RetroPie for old skool gaming, Kali for l33t haxx0rzing? Can’t wait to toss on those cool looking heat sinks that came with the kit so you can overclock the Pi and pwn the crypto scene? Or that neat HAT/Bonnet add on board you also bought?

WAIT! STOP! DON’T!

These options are great, but have potential hiccups to get going due to their complexity. If you jump straight to one of these configurations and run into issues with the Pi "not booting", the troubleshooting becomes extremely troublesome. Is it the OS? Did something get shorted by touching the GPIO pins? Is the HAT/Bonnet configured correctly? Or is it the Pi itself? Unknown.

Even if you’ve configured a million Pi’s in a similar way before - DO THIS FIRST:

Boot the Pi without anything attached, other than power, using the standard Raspberry Pi OS.

Get the Raspberry Pi OS from the Raspberry Pi website:
and follow their instructions for installing the OS image to an SD card. Once done, insert the SD card in the Pi, apply power, and let it boot. The non-Desktop Lite version works fine for this check.

**Do NOT just reused an SD card that "works in another Pi".**

**Headless Configuration**

Pretty much every current model Pi includes WiFi. Therefore, it's fairly easy to configure the Pi to connect to your local network. You can then try pinging the Pi or SSHing in to see if it booted correctly.

The necessary configuration can be done using rpi-imager:

**Configuration Using rpi-imager**

so that the resulting SD card image will be ready to go.

**Attaching a Monitor**

If you're not super familiar with "pinging" or "SSHing" into the Pi, then you'll probably want to attach a monitor to the Pi's HDMI output. This should be fine, but can introduce potential issues. HDMI cables can be bad. HDMI adapters can be bad. EDIDs (identification handshaking between the Pi and the monitor) may get wrapped around the axle. Digital Rights Management (DRM) stuff can introduce many other unknowns. So as simple as this should be, it is not super robust.

Go ahead and try attaching a monitor if it's the only option you're familiar with. One suggested change that seems to help in a lot of cases is to enable safe mode on the HDMI output. To do so, edit /boot/config.txt and find these lines:

```
# uncomment if you get no picture on HDMI for a default "safe" mode
#hdmi_safe=1
```

To uncomment and enable the feature, remove the # from the second line so it looks like:

```
# uncomment if you get no picture on HDMI for a default "safe" mode
hdmi_safe=1
```
This won't be super awesome 4k 120FPS HDMI, but we don't care. We just want to see something usable.

**Power Supply**

This should be simple, but even just getting power to the board can be an issue. In theory, should be able to use a 5V supply with enough current for whatever model Pi you're using, like 3 amps for a Pi 4, etc. But make sure you're using a USB cable that can actually deliver the necessary current (3 amps is a fair amount of juice).

Specific to the Pi 4, early models had a known issue with the USB-C port that would make some power supplies just give up. So make sure to use a known good power supply, like this one:

![Official Raspberry Pi Power Supply 5.1V 3A with USB C](https://www.adafruit.com/product/4298)

**Did It Work?**

How do you know if it worked? All we are looking for is a basic bootup. If you can ping the Pi or get to a basic login prompt, that is good enough. For the headless setup, try SSHing in. If a monitor is attached, just seeing a boot screen is enough.

You now know the basic Pi board hardware itself is nominally OK. You can now move on to whatever specific project you have planned for the Pi. Keeping all the other considerations outlined in this guide in mind as you proceed.

---

**RPI4 Recovery**

The following information is specific to the Raspberry Pi Model 4, which has an entirely new boot flow (relative to previous Pi models. Boot code is now located on a built-in boot EEPROM (NOT on the SD card. Older models used a boot.bin file located on the SD card.)
Sanity Check

This is a simple sanity check of the Pi's state.

- Remove power.
- Remove everything connected to Pi.
- Remove SD card.
- Apply power.

If the EEPROM bootloader is working correctly, you should see a repeating pattern of 4 blinks on the green LED. This is the warning code ( ) that indicates "start*.elf not found", which is expected since the SD card that contains that file has been removed.

If you see the LED blink the expected pattern, you do NOT need to attempt this recovery.

If the expected blink pattern did not appear, then it may be a corrupt EEPROM bootloader image. The following is how to attempt a recovery to re-image the EEPROM bootloader code.

<table>
<thead>
<tr>
<th>Long Flashes</th>
<th>Short Flashes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>Generic failure to boot</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>start*.elf not found</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>Kernel image not found</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>SDRAM failure</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>Insufficient SDRAM</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>In HALT state</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Partition not FAT</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Failed to read from partition</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Extended partition not FAT</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>File signature/hash mismatch - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>SPI EEPROM error - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>SPI EEPROM is write protected - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>I2C error - Pi 4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Unsupported board type</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>Fatal firmware error</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>Power failure type A</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>Power failure type B</td>
</tr>
</tbody>
</table>

Create a Recovery SD Card

To attempt a recovery repair of the EEPROM bootloader, a special recovery SD card image is used. This can be created using the rpi-imager software tool ( ).
Install the SD card into the computer running rpi-imager.

Launch rpi-imager and click the CHOOSE OS button.

Scroll down and pick the Misc Utility images option.

Select Bootloader from the options.
Select SD Card Boot from the options.

It will return to the main window. Click CHOOSE STORAGE and select the SD card.

Then click WRITE to create the recovery SD card image.
Let it write the image.

The contents of the newly created SD card image should look something like this.

Use Recovery SD in Pi 4

Now that the recovery SD image has been created, do the following on the Pi 4 to attempt bootloader image recovery.
Put the SD card in the Pi.

Apply power to the Pi.

Now wait at least 10 seconds for the process to run.
When done, the green LED will blink continuously forever.

Next Steps

Now try booting the Pi with an SD card containing a normal OS image. If you need to reuse the same SD card, you'll need to use rpi-imager again to reimage it with an OS image.

If you already have an SD card with an OS image - make sure it's the latest OS release version.

Troubleshooting

When things don't seem to be working as expected, it's time to troubleshoot. These are some of the common tools and commands that come in very handy for this process. You can also post in the Adafruit Forums and we'll try and help as well.

If your Pi is not booting at all, this sticky in the Raspberry Pi forums is a good place to start.
Boot Error Code via LED blink (Pi 4 and Pi 400, but possibly others!)

From https://www.raspberrypi.com/documentation/computers/configuration.html#led-warning-flash-codes

If a Pi fails to boot for some reason, or has to shut down, in many cases an LED will be flashed a specific number of times to indicate what happened. The LED will blink for a number of long flashes (0 or more), then short flashes, to indicate the exact status. In most cases, the pattern will repeat after a 2 second gap.

<table>
<thead>
<tr>
<th>Long flashes</th>
<th>Short flashes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>Generic failure to boot</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>start+elf not found</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>Kernel image not found</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>SDRAM failure</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>Insufficient SDRAM</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>In HALT state</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Partition not FAT</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Failed to read from partition</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Extended partition not FAT</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>File signature/hash mismatch - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>SPI EEPROM error - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>SPI EEPROM is write protected - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>I2C error - Pi 4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Unsupported board type</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>Fatal firmware error</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>Power failure type A</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>Power failure type B</td>
</tr>
</tbody>
</table>

Getting System Information

Knowing specifics of your operating system and hardware are very useful for troubleshooting. Here are some commands to use to get this information.

To figure out what version of Linux you are running, use:

```
uname -a
```
To get hardware details, use the following, `cat` is used to dump the contents of a file:

```
cat /proc/cpuinfo
```

The model info can also be found here:

```
cat /sys/firmware/devicetree/base/model ; echo
```
An `echo` was added to the above since the file apparently does not end on a new line. This just makes the output look cleaner.

**Viewing System Log**

There are actually numerous system log files (look in `/var/log`), but the basic kernel one, accessed with the command `dmesg`, is the first place to look. There can be a lot of output from running this command. Too much to fit on one screen. One option is to send the output to a file:

```
dmesg &gt; dmesg.txt
```

And then you can view that file in a text editor, or upload it to the forums so support there can take a look.

Another option is to "pipe" the output to something like `more`, which will allow you to paginate through the listing:

```
dmesg | more
```

Another useful way is to filter for specific text using `grep` (the `-i` ignores case):

```
dmesg | grep -i some_text
```

Hmmm. My network cable is not plugged in. Guess that's why `eth0` is not ready.
And yet another way - this may be the most useful - is to only show the most recent output by using `tail`. This is good for when you are trying to do SomeThing and it's not working as expected. Try running the command before doing SomeThing:

```
dmesg | tail -10
```

and then do SomeThing and run the command again. This way you can see the operating system kernel messages associated with what happened when you did SomeThing. The `-10` limits it to the last 10 lines.

For example, how do I know my Raspberry Pi is even seeing my USB Thermal Printer when I plug it in? Well, run the command before plugging in the printer:

```
dmesg | tail -10
```

Nothing exciting, just boot messages about Bluetooth. What we are really interested in is what happens after plugging the printer in:

```
dmesg | tail -10
```

Cool. It saw a new USB device being attached, recognized it as pl2303 and attached it to ttyUSB0. That is all as expected. But if you got the same thing as the first output again - then nothing happened and something is wrong. Or maybe there would be messages that are flagged more as "error", etc.

**Note:**

- The numbers relate to time when message was logged. You can generally ignore them for basic checks like this.
You can change `-10` to some other value to increase/decrease the number of lines shown.

**Is my I2C device there?**

A lot of breakout boards use I2C to communicate with the Raspberry Pi. Each I2C device has a unique address. A quick way to check that the Raspberry Pi can see the connected device is to scan for the addresses of everything attached. To do that, use:

```
i2cdetect -y 1
```

The numbers that show up are the addresses, in hex. For the example above, something with the address 0x48 is seen. To find out what address your device should have, check the product page, or The List:

[The I2C Address List](#)

**Make Sure Power Is Good**

Raspberry Pi's have gotten more power hungry over the years, with the current high end models recommending a whopping 2.5 amps! The actual current consumption will vary depending on what the Raspberry Pi is doing, but you need to accommodate the worst case scenario. Maintaining a good voltage of 5V is also important. Some power supplies will drop in voltage as current draw goes up - even if they claim they are rated for that current.

**How do I know if I have a good power supply?**

It actually can be difficult to know. At a minimum, look for the label on the supply and check its voltage and current rating.
The power supply above has an output of 5.25 volts and is rated for 2 amps.

If you're using a (earlier) Pi 4, use the [Official Pi Power Supply](https://www.adafruit.com/product/4985) (opens in new tab). Original manufactured Pi 4's had a mistake on the USB connector which causes proper "USB Power Delivery (PD)" supplies to get confused and not power the Pi 4. You can tell which version you have by checking this guide. (opens in new tab)

If you have an older Pi, try using the [official Pi Power supply](https://www.adafruit.com/product/4985) (opens in new tab) or [USB Micro B to USB C adapter](https://www.adafruit.com/product/3926) (opens in new tab) attached to a standard micro USB power supply (opens in new tab).

Make sure USB cable is good.

Another important factor for power is the USB cable being used to deliver it to the Raspberry Pi. Cheap charge only cables may have conductors that can only carry a couple 100 milliamps. Even if you have a good power supply, these cables will fail to adequately deliver it to the Raspberry Pi.

How do I know if I have a good USB cable?

This is even more difficult to check. Ideally, the cable itself will have a printed label that will indicate the gauge of the wires used.
The units will probably be AWG (American Wire Gauge). For the cable above, the wires are 28 AWG. For Raspberry Pi usage, that's probably about as thin as you want to go.

The thinner the cable, the higher the AWG number.

If you can't find a label on your USB cable, the only other option is to try other cables.

Use an updated operating system image.

The Raspberry Pi Foundation typically roles out a new release of their operating system with each new Raspberry Pi model. These updated releases are needed to be able to boot the latest models. So make sure you are running latest software with latest hardware.

You should get your operating system images from here:

Raspberry Pi Download Page

If you are just starting out, it's best to steer clear of the Third Party images and use the main Raspbian distribution. Installation onto and SD card has become very easy:

Raspberry Pi Software Installation
Use a text editor, not a word processor, for configuration files

If you are doing setup and configuration by directly editing text files, as shown in this guide, then use a simple text editor, like Notepad, and not a full featured word processor, like Pages, Word, etc. A “text editor” edits text and nothing else, whereas a "word processor" allows for all kinds of formatting, like making text bold, inserting images, etc. A word processor can add additional content to the file which can make it appear unreadable when the operating system tries to access it.

<table>
<thead>
<tr>
<th>Long Flashes</th>
<th>Short Flashes</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>Generic failure to boot</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>start.elf not found</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>Kernel image not found</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>SDRAM failure</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>Insufficient SDRAM</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>In HALT state</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>Partition not FAT</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>Failed to read from partition</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Extended partition not FAT</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>File signature/hash mismatch - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>SPI EEPROM error - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>SPI EEPROM is write protected - Pi 4</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>I2C error - Pi 4</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>Unsupported board type</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>Fatal firmware error</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>Power failure type A</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>Power failure type B</td>
</tr>
</tbody>
</table>

FAQ

The Power Over Ethernet connector is interfering with my HAT/Bonnet/etc.

The PoE connector and the main GPIO header connector are the same height. So in most cases, you should be OK. But if you are finding you want some extra space between them, you can use a socket riser (PID 4079). Add that in between the Pi and the HAT/Bonnet for some extra head room.

Resources

Where to go for more information, help, or other resources.
The Adafruit Forums

Adafruit does not make the Raspberry Pi, so we can't cover everything. But you can always ask for help here.

The Raspberry Pi Foundation

These are the ones that make the Raspberry Pi. Go here for tons of information.

Most importantly, go here for getting the latest operating system images.

They also have a help forum.

GPIO Header Pinouts

What pins are what? This page is an excellent go to reference for the GPIO header pinouts.