Troubleshooting KMS HDMI output

Raspberry Pi Ltd

2023-02-10: githash: c65fe9c-clean

Colophon

build-date: 2023-02-10
build-version: githash: c65fe9c-clean

Legal Disclaimer Notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME ("RESOURCES") ARE PROVIDED BY RASPBERRY PI LTD ("RPL") "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW IN NO EVENT SHALL RPL BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THE RESOURCES, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

RPL reserves the right to make any enhancements, improvements, corrections or any other modifications to the RESOURCES or any products described in them at any time and without further notice.

The RESOURCES are intended for skilled users with suitable levels of design knowledge. Users are solely responsible for their selection and use of the RESOURCES and any application of the products described in them. User agrees to indemnify and hold RPL harmless against all liabilities, costs, damages or other losses arising out of their use of the RESOURCES.

RPL grants users permission to use the RESOURCES solely in conjunction with the Raspberry Pi products. All other use of the RESOURCES is prohibited. No licence is granted to any other RPL or other third party intellectual property right.

HIGH RISK ACTIVITIES. Raspberry Pi products are not designed, manufactured or intended for use in hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, weapons systems or safety-critical applications (including life support systems and other medical devices), in which the failure of the products could lead directly to death, personal injury or severe physical or environmental damage ("High Risk Activities"). RPL specifically disclaims any express or implied warranty of fitness for High Risk Activities and accepts no liability for use or inclusions of Raspberry Pi products in High Risk Activities.

Raspberry Pi products are provided subject to RPL's Standard Terms. RPL's provision of the RESOURCES does not expand or otherwise modify RPL's Standard Terms including but not limited to the disclaimers and warranties expressed in them.

Document version history

Release	Date	Description
1.0	3 Jan 2023	• Initial release

Scope of document

This document applies to the following Raspberry Pi products:

Pi Zero			Pi 1				Pi 2		Pi 3			Pi 4	Pi 400	CM1	CM3	CM4	Pico
Zero	W	H	A	B	A+	B+	A	B	B	A+	B+	All	All	All	All	All	All
*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	

Introduction

With the introduction of the KMS (Kernel Mode Setting) graphics driver, Raspberry Pi Ltd is moving away from legacy firmware control of the video output system and towards a more open source graphics system. However, this has come with its own set of challenges. This document is intended to help with any issues that might arise when moving to the new system.

This whitepaper assumes that Raspberry Pi is running Raspberry Pi OS, and is fully up to date with the latest firmware and kernels.

Terminology

DRM: Direct Rendering Manager, a subsystem of the Linux kernel used to communicate with graphics processing units (GPUs). Used in partnership with FKMS and KMS.

DVI: A predecessor to HDMI, but without the audio capabilities. HDMI to DVI cables and adapters are available to connect a Raspberry Pi device to a DVI-equipped display.

EDID: Extended Display Identification Data. A metadata format for display devices to describe their capabilities to a video source. The EDID data structure includes manufacturer name and serial number, product type, physical display size, and the timings supported by the display, along with some less useful data. Some displays can have defective EDID blocks, which can cause problems if those defects are not handled by the display system.

FKMS (vc4-fkms-v3d): Fake Kernel Mode Setting. While the firmware still controls the low-level hardware (for example, the High-Definition Multimedia Interface (HDMI) ports, the Display Serial Interface (DSI), etc), standard Linux libraries are used in the kernel itself. FKMS is used by default in Buster, but is now deprecated in favour of KMS in Bullseye.

HDMI: High-Definition Multimedia Interface is a proprietary audio/video interface for transmitting uncompressed video data, and compressed or uncompressed digital audio data.

HPD: Hotplug detect. A physical wire that is asserted by a connected display device to show it is present.

KMS: Kernel Mode Setting; see https://www.kernel.org/doc/html/latest/gpu/drm-kms.html for more details. On Raspberry Pi, `vc4-kms-v3d` is a driver that implements KMS, and is often referred to as "the KMS driver".

Legacy graphics stack: A graphics stack wholly implemented in the VideoCore firmware blob exposed by a Linux framebuffer driver. The legacy graphics stack has been used in the majority of Raspberry Pi Ltd devices until recently; it is now gradually being replaced by (F)KMS/DRM.

The HDMI system and the graphics drivers

Raspberry Pi devices use the HDMI standard, which is very common on modern LCD monitors and televisions, for video output. Raspberry Pi 3 (including Raspberry Pi 3B+) and earlier devices have a single HDMI port, which is capable of 1920 × 1200 @60Hz output using a full-size HDMI connector. Raspberry Pi 4 has two micro HDMI ports, and is capable of 4K output on both ports. Depending on setup, the HDMI 0 port on Raspberry Pi 4 is capable of up to 4kp60, but when using two 4K output devices you are limited to p30 on both devices.

The graphics software stack, irrespective of version, is responsible for interrogating attached HDMI devices for their properties, and setting up the HDMI system appropriately. Legacy and FKMS stacks both use firmware in the VideoCore graphics processor to check for HDMI presence and properties. By contrast, KMS uses an entirely open source, ARM-side implementation. This means the code bases for the two systems are entirely different, and in some circumstances this can result in different behaviour between the two approaches.

HDMI and DVI devices identify themselves to the source device using a piece of metadata called an EDID block. This is read by the source device from the display device via an I2C connection, and this is entirely transparent to the end user as it is done by the graphics stack. The EDID block contains a great deal of information, but it is primarily used to specify which resolutions the display supports, so Raspberry Pi can be set up to output an appropriate resolution.

How HDMI is dealt with during booting

When first powered on, Raspberry Pi goes through a number of stages, known as boot stages:

1. The first-stage, ROM-based bootloader starts up the VideoCore GPU.
2. Second-stage bootloader (this is `bootcode.bin` on the SD card on devices prior to Raspberry Pi 4, and in SPI EEPROM on Raspberry Pi 4):
    a. On Raspberry Pi 4, the second-stage bootloader will start up the HDMI system, interrogate the display for possible modes, then set up the display appropriately. At this point the display is used to provide basic diagnostic data.
    b. The bootloader diagnostic display (07 Dec 2022 onwards) will display the status of any attached displays (whether Hotplug Detect (HPD) is present, and whether an EDID block was recovered from the display).
3. The VideoCore firmware (`start.elf`) is loaded and run. This will take over control of the HDMI system, read the EDID block from any attached displays, and show the rainbow screen on those displays.
4. The Linux kernel boots
    a. During kernel boot, KMS will take over control of the HDMI system from the firmware. Once again the EDID block is read from any attached displays, and this information is used to set up the Linux console and desktop.

Possible problems and symptoms

The most common failure symptom experienced when moving to KMS is an initially good boot, with the bootloader screen and then the rainbow screen appearing, followed after a few seconds by the display going black and not coming back on. The point at which the display goes black is in fact the point during the kernel booting process when the KMS driver takes over running the display from the firmware. The Raspberry Pi is currently running in all respects except for the HDMI output, so if SSH is enabled then you should be able to log in to the device by that route. The green SD card access LED will usually flicker occasionally.

It is also possible that you will see no HDMI output at all; no bootloader display, and no rainbow screen. This can usually be attributed to a hardware fault.

Diagnosing the fault

No HDMI output at all

It is possible that the device has not booted at all, but this is outside of the remit of this white paper.

Assuming that the observed behaviour is a display problem, the lack of HDMI output during any part of the booting process is usually due to a hardware fault. There are a number of possible options:

- Defective HDMI cable
  - Try a new cable. Some cables, especially very cheap ones, may not contain all the required communications lines (e.g. hotplug) for Raspberry Pi to successfully detect the display.
- Defective HDMI port on Raspberry Pi
  - If you are using a Raspberry Pi 4, try the other HDMI port.
- Defective HDMI port on the monitor
  - Sometimes the HDMI port on a monitor or TV can wear out. Try a different port if the device has one.
  - Rarely, a display device may only provide EDID data when turned on, or when the correct port is selected. To check, make sure that the device is on and that the correct input port is selected.
- Display device not asserting the hotplug detect line

Initial output, then screen goes black

If the display comes up but then goes off during Linux kernel boot, there are a number of possible causes, and these are usually related to a problem reading the EDID from the display device.

As can be seen from the section above dealing with the boot sequence, the EDID is read at a number of different points during the boot process, and each of these reads is done by a different piece of software. The final read, when KMS takes over, is carried out by unaltered upstream Linux kernel code, and this does not handle defective EDID formats as well as the earlier firmware software. This is why the display can stop working correctly once KMS takes over.

There are number of ways to confirm whether KMS is failing to read the EDID, and two of these are as follows.

Check the bootloader diagnostic screen (Raspberry Pi 4 only)

NOTE

Bootloader diagnostics require a recent bootloader. You can upgrade to the latest version using these instructions: https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#updating-the-bootloader

Remove the SD card and reboot the Raspberry Pi. Press ESC on the `Install OS` screen, and the diagnostic screen should appear on the display device. There should be a line on the display that starts with `display:` — for example:

```
display: DISP0: HDMI HPD=1 EDID=ok #2 DISP1:   HPD=0 EDID=none #0
```

This output from a Raspberry Pi 4 shows that the system detected an HDMI display on HDMI port 0, the hotplug detect is asserted, and the EDID was read OK. Nothing was found on HDMI port 1.

Check whether the KMS system detected an EDID

To check this you will need to log in to the Raspberry Pi device over SSH from a different computer. SSH can be enabled when creating an SD card image with Raspberry Pi Imager, using the Advanced Settings options. Enabling SSH on an SD card that has already been imaged is a little more complicated: you will need to use another computer to add a file named `ssh` to the boot partition. Replace the SD card in the original Raspberry Pi and power it up. This should enable SSH, with an IP address allocated by DHCP.

Once logged in, type the following at the terminal prompt to display the contents of any EDID detected (you may need to change `HDMI-A-1` to `HDMI-A-2` depending on which HDMI port on the Raspberry Pi the display device is connected to):

```
cat /sys/class/drm/card?-HDMI-A-1/edid
```

If there are no folders named `card?-HDMI-A-1` or similar, then is likely that no EDID could be read from the display device.

NOTE

In the case where the EDID is read successfully, there is a useful virtual file in the same folder, called `modes`, which when displayed shows all the possible modes the EDID claims the device supports.

Mitigations

Hotplug detect failure

If both the firmware and KMS fail to find an attached monitor, it could be a hotplug detection failure — i.e., the Raspberry Pi does not know a device has been plugged in, so it doesn't check for an EDID. This could be caused by a bad cable, or a display device that does not assert hotplug correctly.

You can force a hotplug detect by altering the kernel command line file (`cmdline.txt`) that is stored in the boot partition of a Raspberry Pi OS SD card. You can edit this file on another system, using whatever editor you prefer. Add the following to the end of the `cmdline.txt` file:

```
video=HDMI-A-1:1280x720@60D
```

If you are using the second HDMI port, replace `HDMI-A-1` with `HDMI-A-2`. You can also specify a different resolution and frame rate, but make sure you choose ones that the display device supports.

NOTE

Documentation on the kernel command line settings for video can be found here: https://www.kernel.org/doc/Documentation/fb/modedb.txt

WARNING

Older graphics stacks supported the use of a `config.txt` entry to set hotplug detect, but at the time of writing this does not work with KMS. It may be supported in future firmware releases. The `config.txt` entry is `hdmi_force_hotplug`, and you can specify the specific HDMI port that the hotplug applies to using either `hdmi_force_hotplug:0=1` or `hdmi_force_hotplug:1=1`. Note that the nomenclature for KMS refers to the HDMI ports as 1 and 2, while Raspberry Pi uses 0 and 1.

EDID problems

A minority of display devices are incapable of returning an EDID if they are turned off, or when the wrong AV input is selected. This can be an issue when the Raspberry Pi and the display devices are on the same power strip, and the Raspberry Pi device boots faster than the display. With devices like this, you may need to provide an EDID manually.

Even more unusually, some display devices have EDID blocks that are badly formatted and cannot be parsed by the KMS EDID system. In these circumstances, it may be possible to read an EDID from a device with similar resolution and use that.

In either case, the following instructions set out how to read an EDID from a display device and configure KMS to use it, instead of KMS trying to interrogate the device directly.

Copying an EDID to a file

Creating a file containing EDID metadata from scratch is not usually feasible, and using an existing one is much easier. It is generally possible to obtain an EDID from a display device and store it on the Raspberry Pi's SD card so it can be used by KMS instead of getting an EDID from the display device. The easiest option here is to ensure that the display device is up and running and on the correct AV input, and that the Raspberry Pi has started up the HDMI system correctly. From the

terminal, you can now copy the EDID to a file with the following command:

```
sudo cp /sys/class/drm/card?-HDMI-A-1/edid /lib/firmware/myedid.dat
```

If for some reason the EDID is not present, you can boot the device in a non-KMS mode that does succeed in booting to the desktop or console, then copy the EDID that the firmware will (hopefully) successfully read to a file.

1. Boot to legacy graphics mode.
    a. Edit `config.txt` in the boot partition, making sure to run your editor using `sudo`, and change the line that says `dtoverlay=vc4-kms-v3d` to `#dtoverlay=vc4-kms-v3d`.
    b. Reboot.
2. The desktop or login console should now appear.
    a. Using the terminal, copy the EDID from the attached display device to a file with the following command:

```
tvservice -d myedid.dat
sudo mv myedid.dat /lib/firmware/
```

Using a file-based EDID instead of interrogating the display device

Edit `/boot/cmdline.txt`, making sure to run your editor using `sudo`, and add the following to the kernel command line:

```
drm.edid_firmware=myedid.dat
```

You can apply the EDID to a specific HDMI port as follows:

```
drm.edid_firmware=HDMI-A-1:myedid.dat
```

If necessary, boot back into KMS mode by doing the following:

1. Edit `config.txt` in boot partition, making sure to run your editor using `sudo`, and change the line that says `#dtoverlay=vc4-kms-v3d` to `dtoverlay=vc4-kms-v3d`.
2. Reboot.

NOTE

If you use a file-based EDID, but still have problems with hotplug, you can force hotplug detection by adding the following to the kernel command line: `video=HDMI-A-1:D`.

Raspberry Pi