
Using a DPI Display on the Raspberry Pi

Raspberry Pi Ltd

2022-04-29: githash: ba7441c-clean

Colophon

© 2020-2022 Raspberry Pi Ltd (formerly Raspberry Pi (Trading) Ltd.)

This documentation is licensed under a Creative Commons [Attribution-NoDerivatives 4.0 International](#) (CC BY-ND).

build-date: 2022-04-29

build-version: githash: ba7441c-clean

Legal Disclaimer Notice

TECHNICAL AND RELIABILITY DATA FOR RASPBERRY PI PRODUCTS (INCLUDING DATASHEETS) AS MODIFIED FROM TIME TO TIME (THE 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	10 January 2022	Initial release
1.1	27 April 2022	Copy edit, public release

Scope of document

This document applies to the following {pi=prefix} products:

Pi 0			Pi 1		Pi 2		Pi 3	Pi 4	Pi 400	CM 1	CM 3	CM 4	Pico
0	W	H	A	B	A	B	B	All	All	All	All	All	All
*	*	*	*	*	*	*	*	*	*	*	*		

Introduction

Display Parallel Interface (DPI) displays can be connected to Raspberry Pi devices via the 40-pin general-purpose input/output (GPIO) connector as an alternative to using the dedicated Display Serial Interface (DSI) or High-Definition Multimedia Interface (HDMI) ports. Many third-party DPI displays have been made available to take advantage of this. The Buster (and earlier) Raspberry Pi operating system (OS) and the legacy display stack used Raspberry Pi-specific parameters in `config.txt` to configure DPI displays. With the move to Bullseye and its use of the Kernel Mode Setting (KMS) graphics driver by default, these `config.txt` entries are no longer relevant as all control of the display pipeline has shifted to the Linux kernel.

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

Terminology

DPI: Display Parallel interface. A specification for interfacing with display devices, typically liquid crystal display (LCD) panels, that are driven through a set of parallel data lines.

Legacy graphics stack: A graphics stack wholly implemented in the VideoCore firmware blob with a shim application programming interface (API) exposed to the kernel. This is what has been used on the majority of Raspberry Pi Ltd's Pi devices since launch, but is gradually being replaced by KMS/DRM.

vc4-kms-v3d: The full KMS driver for Raspberry Pi devices. Controls the entire display process, including talking to the hardware directly with no firmware interaction.

FKMS: Fake Kernel Mode Setting. While the firmware still controls the low-level hardware (for example the HDMI ports, DSI, DPI, etc.), standard Linux libraries are used in the kernel itself.

KMS: Kernel Mode Setting API.

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.

SoC: System on a chip. The main processor on Raspberry Pi boards, which varies according to model.

DT: Device Tree, a mechanism for defining the hardware characteristics of a device.

Using DPI displays

Simple panels

For simple DPI panels that need no configuration, moving to KMS is relatively straightforward.

The new `vc4-kms-dpi-generic` overlay allows the properties and timings of the panel to be specified with `dtoverlay` / `dtparam` lines in `config.txt`. As the help text describes, it has a number of parameters:

Params:	clock-frequency	Display clock frequency (Hz)
Ê	hactive	Horizontal active pixels
Ê	hfp	Horizontal front porch
Ê	hsync	Horizontal sync pulse width
Ê	hbp	Horizontal back porch
Ê	vactive	Vertical active lines
Ê	vfp	Vertical front porch
Ê	vsync	Vertical sync pulse width
Ê	vbp	Vertical back porch
Ê	hsync-invert	Horizontal sync active low
Ê	vsync-invert	Vertical sync active low
Ê	de-invert	Data Enable active low
Ê	pixel-clock-invert	Negative edge pixel clock
Ê	width-mm	Define the screen width in mm
Ê	height-mm	Define the screen height in mm
Ê	rgb565	Change to RGB565 output on GPIOs 0-19
Ê	rgb666-padhi	Change to RGB666 output on GPIOs 0-9, 12-17, and 20-25
Ê	rgb888	Change to RGB888 output on GPIOs 0-27
Ê	bus-format	Override the bus format for a MEDIA_BUS_FMT_* value. NB also overridden by rgbXXX overrides.
Ê	backlight-gpio	Defines a GPIO to be used for backlight control (default of none).

The `dpi_timings=` line from the legacy `config.txt` system has the following format:

```
dpi_timings=<hactive> <h_sync_polarity> <hfp> <hsync> <hbp> <vactive>
<v_sync_polarity> <vfp> <vsync> <vbp> <n/a> <n/a> <n/a> <n/a> <n/a> <clock-
frequency> <n/a>
```

Using the timings example of the Pimoroni HyperPixel4, the old `config.txt` line

```
dpi_timings=480 0 10 16 59 800 0 15 113 15 0 0 0 60 0 32000000 6
```

converts to

```
dtoverlay=vc4-kms-dpi -generic
dtparam=hactive=480,hfp=10,hsync=16,hbp=59
dtparam=active=800,vfp=15,vsync=113,vbp=15
dtparam=clock-frequency=32000000
```

The legacy `dpi_output_format` line is a bitmasked field to configure various properties of the DPI block, as documented in <https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#controlling-output-format>.

The bits from that field map to `dtparam` settings as follows:

Bits	Description	Options	Default	KMS support
0D3	Output format	rgb565, rgb666, rgb666-padhi, rgb888 (not all the options are mapped)	RGB666 (old mode 5)	partial
4D7	RGB order	Hex value, set in conjunction with output format: 0x1013 for bgr888, 0x101e for bgr666-padhi, 0x101f for bgr666; other orderings are not supported	RGB	partial
8	Output enable mode			no
9	Invert pixel clock	pixclk-invert	off	yes
12	HSync disable			no
13	VSynC disable			no
14	Output enable disable			no
16	HSync polarity	hsync-invert	off	yes
17	VSynC polarity	vsync-invert	off	yes
18	Output enable invert	de-invert	off	yes
20	HSync phase			no
21	VSynC phase			no

The `config.txt` entry

```
dpi_output_format=0x7f216
```

as used on the HyperPixel4 can be translated to the additional `dtparam` lines as follows:

```
dtparam=hsync-invert, vsync-invert, pixclk-invert
dtparam=rgb666-padhi
```

Note: This use of phase with invert actually counteract each other.

The `dtparams` parameters `width-mm` and `height-mm` allow the setting of the physical size of the display advertised to userspace.

The final parameter of note is the backlight control GPIO. The `dtoverlay` option `backlight-gpio` allows a specific GPIO to be assigned to any backlight control implemented on the display. This defaults to `none`.

More advanced timing

Where the `vc4-kms-dpi-generic` overlay does not allow for all options of timings and syncs, and your panel requires no initialisation sequence, it is possible to add a custom panel timing definition to the `panel-simple.c` driver (<https://github.com/torvalds/linux/blob/master/drivers/gpu/drm/panel/panel-simple.c>). This provides access to all the extra options that are not supported by the generic overlay.

The definition requires:

- ✚ A `struct drm_display_mode` to define the timings. The `flags` field can express the polarity of the HSync and VSync pulses if needed, or `DRM_MODE_FLAG_CSYNC` with `DRM_MODE_FLAG_PCSYNC` or `DRM_MODE_FLAG_NCSYNC` can be used for composite sync options (this used to be 'output enable mode' in `dpi_output_format`).
- ✚ A `struct panel_desc` to define the panel size, link to the mode, and specify any additional flags required.
- ✚ An entry in `platform_of_match` to link a compatible string to the `struct panel_desc`.

An overlay can then reference the new compatible string rather than requiring the timings to be specified in the device tree.

If adding compatible strings, follow the basic guidance of making them 'vendor,device', where vendor should come from <https://github.com/torvalds/linux/blob/master/Documentation/devicetree/bindings/vendor-prefixes.yaml>. Ideally, the [device tree binding documentation](#) should also be updated.

Panel-specific drivers

With more complex panels which need to be sent initialisation commands, the correct solution is to write a panel-specific driver and configure it with a specific `dtoverlay`.

This is what is actually required for the HyperPixel4 as it needs a set of Serial Peripheral Interface (SPI) commands to initialise it. That is further complicated by the fact that the panel SPI lines are not wired directly to one of the hardware SPI controllers, so it needs to use the `spi-gpio` module to bit-bash SPI on a generic GPIO. This has been done with the `panel driver` and the `dtoverlay` (<https://github.com/6by9/linux/blob/rpi-5.15.y-hyperpixel4/arch/arm/boot/dts/overlays/vc4-kms-dpi-hyperpixel4-overlay.dts>).

As this is a dedicated panel driver, the timing configuration is in the driver instead of device tree! See the function `ili9806e_480x800_mode` for more details.

Panel drivers have four main functions:

- ✚ `prepare`: Used to power up and configure. Video is not enabled at this point.
- ✚ `enable`: Enable the display. Video is enabled.
- ✚ `disable`: Disable the display. Video is still running.
- ✚ `unprepare`: Power down the display. Video is disabled before this call.

As can be seen in the `panel-ili9806e` driver, `prepare` sends all the SPI commands to configure the panel mode (taken from their older `configuration script`), and `enable` sends the `DISPLAY_ON` command. Conversely, `disable` sends the `DISPLAY_OFF` command, and `unprepare` puts the panel to sleep.

A backlight is defined in the overlay using the `gpio-backlight` driver, and then associated with the panel driver. Doing this means that the DRM framework handles enabling and disabling of the backlight rather than the panel driver having to do it. `gpio-backlight` allows a GPIO to turn a backlight on/off. There is also `pwm-backlight` which allows the use of a pulse-width modulation (PWM) output to give finer control of backlight intensity! See `cutiepi-panel-overlay.dts` for an example of using this.



Raspberry Pi

Raspberry Pi is a trademark of the Raspberry Pi Foundation

Raspberry Pi Ltd