

IMX296-130 BW Camera

From Waveshare Wiki

Jump to: navigation, search

Quick Test

Hardware Connection

To test the Raspberry Pi camera, you need to connect the Raspberry Pi to an HDMI screen or DSI screen

The packages of the CSI (camera) and DSI (monitor) ports on the Raspberry Pi motherboard are similar, so be careful not to connect them incorrectly when wiring. The CSI interface is located between the audio jack and the HDMI port. The camera interface of the Raspberry Pi Zero series is next to the power interface. If you are using a computing module, it depends on the layout of the baseplate.

- Connect to Raspberry Pi 5

Place the metal surface of the cable towards the wired network port and insert it into the port. Pi5 has two CSI interfaces, and either interface can be connected

IMX296-130 BW Camera



(<https://www.waveshare.com/imx296-130-bw-camera.htm>)

1.6MP, IMX296, 130°, CSI



(/wiki/File:IMX296-130-BW-

Camera-details-5-2.jpg)

- Connect to the Raspberry Pi Zero series

Put the metal side of the cable downwards and connect it to the camera interface



(/wiki/File:IMX296-130-BW-

Camera-details-5-3.jpg)

- Other Raspberry Pi motherboards

Put the metal side of the cable downwards the HDMI interface and connect it to the camera interface

500px (/w/index.php?title=Special:Upload&wpDestFile=IMX296-130-BW-Camera-5.jpg)

Test Camera

Software Configuration

If you are using the latest Raspberry Pi Camera Module 3 or Raspberry Pi Global Shutter Camera, you need to run the following command to update the system (Internet connection required).

```
sudo apt-get update -y
sudo apt-get upgrade -y
```

If only one camera is called, connect the camera to the CAM1 port.

If you are using a camera that is not the official Raspberry Pi, you need to configure a config.txt file separately.

If you are using the latest Bookworm system, you need to configure /boot/firmware/config.txt.

```
sudo nano /boot/config.txt
#If it is the bookworm system
sudo nano /boot/firmware/config.txt
```

Find the camera-auto-detect=1 statement and change it to camera_auto_detect=0

At the end of the file, add the following setting statements according to the camera model

Model	Setting statement
OV9281	dtoverlay=ov9281
IMX290/IMX327	dtoverlay=imx290,clock-frequency=37125000
IMX378	dtoverlay=imx378
IMX219	dtoverlay=imx219
IMX415	dtoverlay=imx415,clk-37125
IMX462	dtoverlay=imx462
IMX477	dtoverlay=imx477
IMX708	dtoverlay=imx708
IMX296	dtoverlay=imx296

Note: To use the IMX290 on the Raspberry Pi 5, you also need to add the json file to the command directory to use it. The operation is as follows:

```
sudo wget https://files.waveshare.com/wiki/common/Imx290.zip
sudo unzip Imx290.zip
sudo cp imx290.json /usr/share/libcamera/ipa/rpi/pisp
```

Binocular camera configuration

- At present, both the CM4 baseplate and the Raspberry Pi 5 support access to two cameras
- If the user needs to access two camera calls at the same time, cam0 and cam1 can be added after the corresponding camera configuration statement to specify the camera.
 - For example, if IMX219 is connected to CAM0 interface, and at the same time OV5647 camera interface is connected to CAM1

```
dtoverlay=imx219,cam0
dtoverlay=ov5647,cam1
```

Test Camera (Bullseye System)

Quick Call

Applicable for OV5647, official IMX219, official IMX477.

Open the Raspberry Pi terminal and turn on the camera preview:

```
sudo libcamera-hello -t 0
```

If you want to close the preview window, you can use Ctrl-C to terminate the program.

Test Camera (Bookworm System)

Quick Call

Applicable for OV5647, official IMX219, official IMX477.

Open the Raspberry Pi terminal and turn on the camera preview:

```
sudo rpicas-hello -t 0
```

If you want to close the preview window, you can use Ctrl-C to terminate the program.

Preface

To check what version of the system you are using, run `sudo cat /etc/os-release` to see if there is any information about the following two images, and then select.

- The Raspberry Pi OS Bookworm changed the camera capture application from `libcamera-*` to `rpicas-*`, which currently allows users to continue using the old `libcamera`, but `libcamera` will be deprecated in the future, so please use `rpicas` as soon as possible.
- If you are using the Raspberry Pi OS Bullseye system, please scroll down the page to use this tutorial's `libcamera-*` section.

RPicas

When running the latest version of Raspberry Pi OS, `rpicas-apps` already has five basic features installed. In this case, the official Raspberry Pi camera will also be detected and automatically enabled.

You can check if everything is working by inputting the following:

```
rpicas-hello
```

You should see a camera preview window for about five seconds.

Note: If you're running on a Bullseye Raspberry Pi 3 and earlier, you need to re-enable Glamor to ensure that the X Windows hardware acceleration preview window works properly. Input the command `sudo raspi-config` in the terminal window, then select Advanced Options, Glamor, and Yes. Exit and restart your Raspberry Pi. By default, Raspberry

Pi 3 and earlier devices running Bullseye may not be using the correct display driver. Please refer to the `/boot/firmware/config.txt` file and make sure that `dtoverlay=vc4-fkms-v3d` or `dtoverlay=vc4-kms-v3d` is currently active. If you need to change this setting, please restart.

rpicam-hello

It is equivalent to the camera's "hello world", it starts the camera preview stream and displays it on the screen, which can be stopped by clicking the window's Close button or by using `ctrl^C` in the terminal.

```
rpicam-hello -t 0
```

Tuning file

The `libcamera` for Raspberry Pi has tuning files for each type of camera module. The parameters in the file are passed to the algorithm and hardware to produce the best quality image. `libcamera` can only automatically determine the image sensor being used, not the entire module, even if the entire module affects the "tuning". As a result, it is sometimes necessary to override the default tuning file for a particular sensor.

For example, a sensor without an infrared filter (NoIR) version requires a different AWB (white balance) setting than the standard version, so an IMX219 NoIR used with a Pi 4 or earlier device should operate as follows:

```
rpicam-hello --tuning-file /usr/share/libcamera/ipa/rpi/vc4/imx219_noir.json
```

The Raspberry Pi 5 uses different tuning files in different folders, so here you will use:

```
rpicam-hello --tuning-file /usr/share/libcamera/ipa/rpi/pisp/imx219_noir.json
```

This also means that users can copy existing tuning files and modify them according to their preferences, as long as the parameter `--tuning-file` points to the new version.

The `--tuning-file` parameter is applicable to all `rpicam-apps` just like other command-line options.

rpicam-jpeg

`rpicam-jpeg` is a simple static image capture application.

To capture a full-resolution JPEG image, use the following command. This will display a preview for approximately five seconds, then capture the full-resolution JPEG image to the file `test.jpg`.

```
rpicam-jpeg -o test.jpg
```

The `-t <duration>` option is used to change the duration of the preview display, and the `--width` and `--height` options will alter the resolution of the captured static images. For example:

```
rpicam-jpeg -o test.jpg -t 2000 --width 640 --height 480
```

Exposure control

All of this `rpicam`-apps allow the user to run the camera with a fixed shutter speed and gain. Capture an image with an exposure time of 20ms and a gain of 1.5x. This gain will be used as an analog gain within the sensor until it reaches the maximum analog gain allowed by the core sensor driver. After that, the rest will be used as a digital gain

```
rpicam-jpeg -o test.jpg -t 2000 --shutter 20000 --gain 1.5
```

The AEC/AGC algorithm for the Raspberry Pi enables application-defined exposure compensation, allowing images to be made darker or brighter through a specified number of stops.

```
rpicam-jpeg --ev -0.5 -o darker.jpg  
rpicam-jpeg --ev 0 -o normal.jpg  
rpicam-jpeg --ev 0.5 -o brighter.jpg
```

Digital gain

The digital gain is applied by the ISP, not by the sensor. The digital gain will always be very close to 1.0 unless:

- The total gain requested (via the option `--gain` or via the exposure profile in the camera adjustment) exceeds the gain that can be used as an analog gain within the sensor. Only the extra gain required will be used as a digital gain.
- One of the color gains is less than 1 (note that the color gain is also used as a digital gain). In this case, the published digital gain will stabilize at $1 / \min(\text{red gain, blue gain})$. This means that one of the color channels (instead of the green channel) has a single-digit gain applied.
- AEC/AGC is changing. When AEC/AGC moves, the digital gain typically changes to some extent in an attempt to eliminate any fluctuations, but it quickly returns to its normal value.

rpicam-still

It emulates many features of the original app `raspistill`.

```
rpicam-still -o test.jpg
```

Encoder

rpicam-still allows files to be saved in a number of different formats. It supports PNG and BMP encoding. It also allows the file to be saved as a binary dump of RGB or YUV pixels with no encoding or file format. In the latter case, the application that reads the file must be aware of its own pixel arrangement.

```
rpicam-still -e png -o test.png
rpicam-still -e bmp -o test.bmp
rpicam-still -e rgb -o test.data
rpicam-still -e yuv420 -o test.data
```

Note: The format in which the image is saved depends on the `-e` (equivalent to encoding) option and is not automatically selected based on the output file name.

Raw image capture

The raw image is an image that is produced directly by an image sensor, before any processing of it by the ISP (Image Signal Processor) or any CPU core. For color image sensors, these are usually Bayer format images. Note that the original image is very different from the processed but unencoded RGB or YUV images we saw before.

Get the raw image:

```
rpicam-still --raw --output test.jpg
```

Here, the `-r` option (also known as raw) indicates capturing the raw image and JPEG. In fact, the original image is the raw image that generates a JPEG. The original images are saved in DNG (Adobe Digital Negative) format and are compatible with many standard applications such as draw or RawTherapee. The original image is saved to a file with the same name but an extension of `.ng`, thus becoming `test.dng`.

These DNG files contain metadata related to image capture, including black level, white balance information, and the color matrix used by the ISP to generate JPEGs. This makes these DNG files more convenient to use for "manual" original conversion with some of the above tools in the future. Use `exiftool` to display all metadata encoded into a DNG file:

```

File Name           : test.dng
Directory          : .
File Size          : 24 MB
File Modification Date/Time : 2021:08:17 16:36:18+01:00
File Access Date/Time   : 2021:08:17 16:36:18+01:00
File Inode Change Date/Time : 2021:08:17 16:36:18+01:00
File Permissions     : rw-r--r--
File Type          : DNG
File Type Extension  : dng
MIME Type          : image/x-adobe-dng
Exif Byte Order     : Little-endian (Intel, II)
Make              : Raspberry Pi
Camera Model Name   : /base/soc/i2c0mux/i2c@1/imx477@1a
Orientation        : Horizontal (normal)
Software          : rpicas-still
Subfile Type       : Full-resolution Image
Image Width        : 4056
Image Height       : 3040
Bits Per Sample    : 16
Compression        : Uncompressed
Photometric Interpretation : Color Filter Array
Samples Per Pixel  : 1
Planar Configuration : Chunky
CFA Repeat Pattern Dim : 2 2
CFA Pattern 2      : 2 1 1 0
Black Level Repeat Dim : 2 2
Black Level        : 256 256 256 256
White Level        : 4095
DNG Version        : 1.1.0.0
DNG Backward Version : 1.0.0.0
Unique Camera Model : /base/soc/i2c0mux/i2c@1/imx477@1a
Color Matrix 1     : 0.8545269369 -0.2382823821 -0.09044229197 -0.1890484985
1.063961506 0.1062747385 -0.01334283455 0.1440163847 0.2593136724
As Shot Neutral    : 0.4754476844 1 0.413686484
Calibration Illuminant 1 : D65
Strip Offsets      : 0
Strip Byte Counts   : 0
Exposure Time      : 1/20
ISO                : 400
CFA Pattern        : [Blue,Green][Green,Red]
Image Size         : 4056x3040
Megapixels         : 12.3
Shutter Speed      : 1/20

```

We have noticed that there is only one calibration light source (determined by the AWB algorithm, although it is always labeled as "D65"), and dividing the ISO number by 100 gives the analog gain being used.

Ultra-long exposure

In order to capture long-exposure images, disable AEC/AGC and AWB, as these algorithms will force the user to wait many frames while converging.

The way to disable them is to provide explicit values. Additionally, the `--immediate` option can be used to skip the entire preview phase that is captured.

Therefore, to perform an exposure capture of 100 seconds, use:

```
rpicam-still -o long_exposure.jpg --shutter 100000000 --gain 1 --awbgains 1,1 --immediate
```

For reference, the maximum exposure times of the three official Raspberry Pi cameras can be found in this table (<https://www.raspberrypi.com/documentation/accessories/camera.html#hardware-specification>).

rpicam-vid

`rpicam-vid` can help us capture video on our Raspberry Pi device. `Rpicam-vid` displays a preview window and writes the encoded bitstream to the specified output. This will produce an unpacked video bitstream that is not packaged in any container format (such as an mp4 file).

- `rpicam-vid` uses H.264 encoding

For example, the following command writes a 10-second video to a file named `test.h264`:

```
rpicam-vid -t 10s -o test.h264
```

You can use VLC and other video players to play the result files:

```
VLC test.h264
```

On the Raspberry Pi 5, you can output directly to the MP4 container format by specifying the MP4 file extension of the output file:

```
rpicam-vid -t 10s -o test.mp4
```

Encoder

`rpicam-vid` supports dynamic JPEG as well as uncompressed and unformatted YUV420:

```
rpicam-vid -t 10000 --codec mjpeg -o test.mjpeg  
rpicam-vid -t 10000 --codec yuv420 -o test.data
```

The codec option determines the output format, not the extension of the output file.

The segment option splits the output file into segments-sized chunks (in milliseconds). By specifying extremely short segments (1 millisecond), this allows for the convenient decomposition of a moving JPEG stream into individual JPEG files. For example, the following command combines a 1 millisecond segment with a counter in the output filename to generate a new filename for each segment:

```
rpicam-vid -t 10000 --codec mjpeg --segment 1 -o test%05d.jpeg
```

Capture high frame rate video

To minimize frame loss for high frame rate (> 60fps) video, try the following configuration adjustments:

- Set the target level of H.264 to 4.2 with the parameter `--level 4.2`
- Disable software color denoising processing by setting the denoise option to `cdn_off`.
- Disable the display window for `nopreview` to release additional CPU cycles.
- Set `force_turbo=1` in `/boot/firmware/config.txt` to ensure that the CPU clock is not throttled during video capture. For more information, see the `force_turbo` (https://www.raspberrypi.com/documentation/computers/config_txt.html#force_turbo) documentation.
- Adjust the ISP output resolution parameter to `--width 1280 --height 720` or lower to achieve the frame rate target.
- On the Raspberry Pi 4, you can overclock the GPU to improve performance by adding a frequency of `gpu_freq=550` or higher in `/boot/firmware/config.txt`. For detailed information, please refer to the `Overclocking` (https://www.raspberrypi.com/documentation/computers/config_txt.html#overclocking) documentation.

The following command demonstrates how to implement a 1280×720 120fps video:

```
rpicam-vid --level 4.2 --framerate 120 --width 1280 --height 720 --save-pts timestamp.pts  
-o video.264 -t 10000 --denoise cdn_off -n
```

Integration of Libav with rpicam-vid

Rpicam-vid can encode audio and video streams using the ffmpeg/libav codec backend. You can save these streams to a file, or stream them over the network.

To enable the libav backend, pass `libav` to the codec option:

```
rpicam-vid --codec libav --libav-format avi --libav-audio --output example.avi
```

UDP

To use a Raspberry Pi as a server for streaming video over UDP, use the following command, replacing the < IP -addr> placeholder with the IP address of the client or multicast address, and replacing the <port> placeholder with the port you wish to use for streaming:

```
rpicam-vid -t 0 --inline -o udp://<ip-addr>:<port>
```

Use a Raspberry Pi as a client to view video streams over UDP, using the following command, replace the <port> placeholder with the port you want to stream:

```
vlc udp://@:<port> :demux=h264
```

Alternatively, use ffplay on the client side to stream with the following command:

```
ffplay udp://<ip-addr-of-server>:<port> -fflags nobuffer -flags low_delay -framedrop
```

TCP

Video can also be transmitted over TCP. Use Raspberry Pi as a server:

```
rpicam-vid -t 0 --inline --listen -o tcp://0.0.0.0:<port>
```

Use the Raspberry Pi as the client to view the video stream over TCP, use the following command:

```
vlc tcp/h264://<ip-addr-of-server>:<port>
```

Alternatively, use the ffplay stream at 30 frames per second on the client side with the following command:

```
ffplay tcp://<ip-addr-of-server>:<port> -vf "setpts=N/30" -fflags nobuffer -flags low_delay -framedrop
```

RTSP

To transfer video via RTSP using VLC, using the Raspberry Pi as the server, use the following command:

```
rpicam-vid -t 0 --inline -o - | cvlc stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/stream1}' :demux=h264
```

To view the video stream on the RTSP using the Raspberry Pi as a client, use the following command:

```
ffmpeg rtsp://<ip-addr-of-server>:8554/stream1 -vf "setpts=N/30" -fflags nobuffer -flags low_delay -framedrop
```

Or on the client side use the following command to stream with VLC:

```
vlc rtsp://<ip-addr-of-server>:8554/stream1
```

If you need to close the preview window on the server, use the nopreview command.

Use inline flags to enforce stream header information into each inner frame, which helps the client understand the stream when the beginning is missed.

rpicam-raw

rpicam-raw records the video directly from the sensor as the original Bayer frame. It doesn't show the preview window. To record a two-second raw clip to a file called test.raw, run the following command:

```
rpicam-raw -t 2000 -o test.raw
```

RPICAM-RAW outputs raw frames without any format information. The application prints the pixel format and image size to the terminal window to help the user parse the pixel data. By default, rpicam-raw outputs raw frames in a single, potentially very large file. Use the segment option to direct each raw frame to a separate file, using the %05d directive to make each frame filename unique:

```
rpicam-raw -t 2000 --segment 1 -o test%05d.raw
```

Using a fast storage device, rpicam-raw can write 18MB frames from a 12-megapixel HQ camera at a speed of 10fps to the disk. rpicam-raw is unable to format the output frame as a DNG file; To do this, use the rpicam-still option at a frame rate lower than 10 to avoid frame drops:

```
rpicam-raw -t 5000 --width 4056 --height 3040 -o test.raw --framerate 8
```

For more information on the original format, see mode (https://www.raspberrypi.com/documentation/computers/camera_software.html#mode) documentation.

rpicam-detect

Note: The Raspberry Pi operating system does not include rpicam-detect. If you already have TensorFlow Lite installed, you can build rpicam-detect. For more information, see the instructions on building rpicam-apps in build (<https://www.raspberrypi.com/documentation/>

computers/camera_software.html#build-libcamera-and-rpicam-apps). Don't forget to pass `-DENABLE_TFLITE=1` when running `cmake`.

`rpicam-detect` displays a preview window and monitors the content using a Google MobileNet v1 SSD (Single Shot Detector) neural network that has been trained to recognize about 80 classes of objects using the Coco dataset. `Rpicam-detect` can recognize people, cars, cats and many other objects.

Whenever `rpicam-detect` detects a target object, it captures a full-resolution JPEG. Then return to monitoring preview mode.

For general information about model usage, please refer to the TensorFlow Lite Object Detector (https://www.raspberrypi.com/documentation/computers/camera_software.html#object_detect_tf-stage) section. For example, when you are out, you can keep an eye on your cat:

```
rpicam-detect -t 0 -o cat%04d.jpg --lores-width 400 --lores-height 300 --post-process-file object_detect_tf.json --object cat
```

rpicam Parameter Settings

- `--help -h` prints all the options, along with a brief description of each option

```
rpicam-hello -h
```

- `--version` outputs the version strings for `libcamera` and `rpicam-apps`

```
rpicam-hello --version
```

Example output:

```
rpicam-apps build: ca559f46a97a 27-09-2021 (14:10:24)
libcamera build: v0.0.0+3058-c29143f7
```

- `--list-cameras` lists the cameras connected to the Raspberry Pi and their available sensor modes

```
rpicam-hello --list-cameras
```

The identifier for the sensor mode has the following form:

```
S<Bayer order><Bit-depth>_<Optional packing> : <Resolution list>
```

Cropping is specified in the native sensor pixels (even in pixel binning mode) as (<x>, <y>)/<Width> × <Height>. (x, y) specifies the position of the width × height clipping window in the sensor array.

For example, the following output shows information for an IMX219 sensor with index 0 and an IMX477 sensor with index 1:

```
Available cameras
-----
0 : imx219 [3280x2464] (/base/soc/i2c0mux/i2c@1/imx219@10)
  Modes: 'SRGGB10_CSI2P' : 640x480 [206.65 fps - (1000, 752)/1280x960 crop]
        1640x1232 [41.85 fps - (0, 0)/3280x2464 crop]
        1920x1080 [47.57 fps - (680, 692)/1920x1080 crop]
        3280x2464 [21.19 fps - (0, 0)/3280x2464 crop]
  'SRGGB8' : 640x480 [206.65 fps - (1000, 752)/1280x960 crop]
        1640x1232 [41.85 fps - (0, 0)/3280x2464 crop]
        1920x1080 [47.57 fps - (680, 692)/1920x1080 crop]
        3280x2464 [21.19 fps - (0, 0)/3280x2464 crop]
1 : imx477 [4056x3040] (/base/soc/i2c0mux/i2c@1/imx477@1a)
  Modes: 'SRGGB10_CSI2P' : 1332x990 [120.05 fps - (696, 528)/2664x1980 crop]
        'SRGGB12_CSI2P' : 2028x1080 [50.03 fps - (0, 440)/4056x2160 crop]
        2028x1520 [40.01 fps - (0, 0)/4056x3040 crop]
        4056x3040 [10.00 fps - (0, 0)/4056x3040 crop]
```

- `--camera` selects the camera you want to use. Specify an index from the list of available cameras.

```
rpicam-hello --list-cameras 0
rpicam-hello --list-cameras 1
```

- `--config -c` specifies a file that contains the command parameter options and values. Typically, a file named `example_configuration.txt` specifies options and values as key-value pairs, with each option on a separate line.

```
timeout=99000
verbose=
```

Notice: Omission of prefixes `--` for parameters is typically used in command lines. For flags that are missing values, such as `verbose` in the example above, a trailing `=` must be included. Then you can run the following command to specify a timeout of 99,000 milliseconds and detailed output:

```
rpicam-hello --config example_configuration.txt
```

- `--time -t`, the default delay of 5000 milliseconds

```
rpicam-hello -t
```

Specifies how long the application runs before shutting down. This applies to both the video recording and preview windows. When capturing a still image, the application displays a preview window with a timeout millisecond before outputting the captured image.

```
rpicam-hello -t 0
```

- `--preview` sets the position (x,y coordinates) and size (w,h dimensions) of the desktop or DRM preview window. There is no impact on the resolution or aspect ratio of the image requested from the camera

Pass the preview window dimensions in the following comma-separated form: x,y,w,h

```
rpicam-hello --preview 100,100,500,500
```

- `--fullscreen -f` forces the preview window to use the entire screen, with no borders or title bars. Scale the image to fit the entire screen. Values are not accepted.

```
rpicam-hello -f
```

- `--qt-preview` uses the Qt preview window, which consumes more resources than other options, but supports X window forwarding. Not compatible with full-screen flags. Values are not accepted.

```
rpicam-hello --qt-preview
```

- `--nopreview` makes the application not display a preview window. Values are not accepted.

```
rpicam-hello --nopreview
```

- `--info-text`

Default values: "#%frame (%fps fps) exp %exp ag %ag dg %dg"

When running in a desktop environment, set the provided string as the title of the preview window. The following image metadata substitutions are supported:

Command Description

%frame	Frame sequence number
%fps	Instantaneous frame rate
%exp	Shutter speed at which an image is captured, in ms
%ag	Image analog gain controlled by sensor chip
%dg	Image number gain controlled by ISP
%rg	Gain of the red component of each pixel point

%bg	Gain of the blue component of each pixel point
%focus	The corner point measure of the image, the larger the value, the clearer the image
%lp	Diopter of the current lens (distance in 1/meter)
%afstate	Autofocus status (idle, scanning, focused, failed)

```
rpicam-hello --info-test "Focus measure: %focus"
```

- --width
- --height

Each parameter accepts a number that defines the size of the image displayed in the preview window in pixels.

For rpicam-still, rpicam-jpeg, and rpicam-vid, specify the output resolution.

For rpicam-raw, specify the original frame resolution. For a camera with a 2×2 bin readout mode, specify a resolution that is equal to or less than the bin mode to capture 2×2 bin original frames.

For rpicam-hello, there is no effect.

Record a 1080p video

```
rpicam-vid -o test.h264 --width 1920 --height 1080
```

Capture a JPEG at a resolution of 2028×1520. If used with an HQ camera, the 2×2 bin mode is used, so the original file (test. ng) contains 2028×1520 original Bayer images.

```
rpicam-still -r -o test.jpg --width 2028 --height 1520
```

- --viewfinder-width
- --viewfinder-height

Each parameter accepts a number that defines the size of the image displayed in the preview window in pixels. The size of the preview window is not affected, as the image is resized to fit. Captured still images or videos are not affected.

```
rpicam-still --viewfinder-width 1920 --viewfinder-height 1080
```

- --mode allows the camera mode to be specified in the following colon-separated format: <width>:<height>:<bit-depth>:<packing> and if the provided values do not match exactly, the system will select the closest available option for the sensor. You can use the packed(P) or unpacked(U) packaging format, which affects the format of the stored video and static images, but does not affect the format of the frames transmitted to the preview window.

Bit-depth and packing are optional. By default, Bit-depth is 12, and Packing is set to P (packed).

For information on the bit depth, resolution, and packing options available for the sensors, please refer to [list-cameras](#).

As shown below:

- ■ 4056:3040:12:P - 4056×3040 resolution, 12 bits/pixel, packed.
- ■ 1632:1224:10 - 1632×1224 resolution, 10 bits/pixel.
- ■ 2592:1944:10:U - 2592×1944 resolution, 10 bits/pixel, unpacked.
- ■ 3264:2448 - 3264×2448 resolution.
- --viewfinder-mode is the same as the mode option, but it applies to the data passed to the preview window. For more information, see mode (https://www.raspberrypi.com/documentation/computers/camera_software.html#mode) documentation.
- --lores-width and --lores-height

Provide a second low-resolution image stream from the camera, scaled down to a specified size. Each accepts a number to define the dimensions of a low-resolution stream (in pixels). Available in preview and video modes. Static capture is not provided. For RPICAM-vid, disable additional color denoising processing. It is useful for image analysis combined with Image post-processing (https://www.raspberrypi.com/documentation/computers/camera_software.html#post-processing-with-rpicam-apps).

```
rpicam-hello --lores-width 224 --lores-height 224
```

- --hflip flips the image horizontally. Values are not accepted.

```
rpicam-hello --hflip -t 0
```

- --vflip flips the image vertically. Values are not accepted.

```
rpicam-hello --vflip -t 0
```

- --rotation rotates the image extracted from the sensor. Only values of 0 or 180 are accepted.

```
rpicam-hello --rotation 0
```

- --roi crops the image extracted from the entire sensor domain. Accepts four decimal values, ranged 0 to 1, in the following format: <x>,<y>,<w>,<h>. Each of these values represents the percentage of available width and height as a decimal between 0 and 1.

These values define the following proportions:

<x>: X coordinates to skip before extracting an image

<y>: Y coordinates to skip before extracting an image

<w>: image width to extract

<h>: image height to extract

The default is 0,0,1,1 (starting with the first X coordinate and the first Y coordinate, using 100% of the image width and 100% of the image height).

Examples:

`rpicam-hello --roi 0.25, 0.25, 0.5, 0.5` selects half of the total number of pixels cropped from the center of the image (skips the first 25% of the X coordinates and the first 25% of the Y coordinates, uses 50% of the total width of the image and 50% of the total height of the image).

`rpicam-hello --roi 0,0,0.25,0.25` selects a quarter of the total number of pixels cropped from the top left corner of the image (skips the first 0% of the X coordinate and the first 0% of the Y coordinate, uses 25% of the width of the image and 25% of the height of the image).

- `--HDR` default: Off, run the camera in HDR mode. If no value is passed, it is assumed to be auto. Accept one of the following values:
 - off - disables HDR.
 - auto - enables HDR on supported devices. If available, use the sensor's built-in HDR mode. If the sensor does not have a built-in HDR mode, the onboard HDR mode is used (if available).
 - single-exp enables HDR on supported devices. If available, use the sensor's built-in HDR mode. If the sensor does not have a built-in HDR mode, the onboard HDR mode is used (if available).

```
rpicam-hello --hdr
```

Use the onboard HDR mode, if available, even if the sensor has a built-in HDR mode. If the onboard HDR mode is not available, HDR is disabled.

Raspberry Pi 5 and higher versions of devices have the onboard HDR mode.

To check the HDR mode built into the sensor, add this option to the list of cameras.

Camera Control Options

The following options control the image processing and algorithms that affect the image quality of the camera.

- sharpness

Sets the image clarity. Values from the following ranges are accepted:

- - 0.0 refers to not applying sharpening
 - Values greater than 0.0 but less than 1.0 apply less sharpening than the default value
 - 1.0 applies the default sharpening amount
 - Values greater than 1.0 apply additional sharpening

```
rpicam-hello --sharpness 0.0
```

- contrast

Specifies the image contrast. Values from the following ranges are accepted:

- - 0.0 applies minimum contrast ratio
 - Values greater than 0.0 but less than 1.0 apply a contrast that is less than the default value
 - 1.0 applies the default contrast ratio
 - Values greater than 1.0 apply additional contrast

```
rpicam-hello --contrast 0.0
```

- brightness

Specifies the image brightness, which is added as an offset of all pixels in the output image.

Values from the following ranges are accepted:

- - -1.0 refers to minimum brightness (black)
 - 0.0 applies standard brightness
 - 1.0 applies maximum brightness (white)

For more uses, refer to `ev` (https://www.raspberrypi.com/documentation/computers/camera_software.html#ev).

```
rpicam-hello --brightness 1.0
```

- saturation

Specifies the image color saturation. Values from the following ranges are accepted:

- - 0.0 applies minimum saturation (grayscale)
 - Values greater than 0.0 but less than 1.0 apply saturation less than the default value
 - 1.0 applies the default saturation
 - Values greater than 1.0 apply additional saturation

```
rpicam-hello --saturation 0.6
```

- ev

Specifies the exposure value (EV) (https://en.wikipedia.org/wiki/Exposure_value)

compensation for the image. A numeric value is accepted that is passed along the following spectrum to the target value of the automatic exposure/gain control (AEC/AGC) processing algorithm:

- - -10.0 applies the minimum target value
 - 0.0 applies standard target value
 - 10.0 applies the maximum target value

```
rpicam-hello --ev 10.0
```

- shutter

Specifies the exposure time using the shutter, measured in microseconds. When you use this option, the gain can still be varied. If the camera's frame rate is too high, it doesn't allow the specified exposure time (for example, with a frame rate of 1 fps and an exposure time of 10,000 microseconds), the sensor will use the maximum exposure time allowed by the frame rate.

For a list of minimum and maximum shutter times for official cameras, see camera hardware documentation (<https://www.raspberrypi.com/documentation/accessories/camera.html#hardware-specification>). Values higher than the maximum will result in undefined behavior.

```
rpicam-hello --shutter 10000
```

- gain

The effect of analoggain is the same as gain

Sets the combined analog and digital gain. When the sensor drive can provide the required gain, only analog gain is used. When the analog gain reaches its maximum, the ISP applies the digital gain. Accepts a numeric value.

For a list of analogue gain limits, for official cameras, see the camera hardware documentation (<https://www.raspberrypi.com/documentation/accessories/camera.html#hardware-specification>).

Sometimes, digital gain can exceed 1.0 even when the analogue gain limit is not exceeded.

This can occur in the following situations:

Either of the colour gains drops below 1.0, which will cause the digital gain to settle to $1.0/\min(\text{red_gain}, \text{blue_gain})$. This keeps the total digital gain applied to any colour channel above 1.0 to avoid discolouration artefacts.

Slight variances during Automatic Exposure/Gain Control (AEC/AGC) changes.

```
rpicam-hello --gain 0.8
```

- metering default value: centre

Sets the metering mode of the Automatic Exposure/Gain Control (AEC/AGC) algorithm.

Accepts the following values:

- centre - centre weighted metering
- spot - spot metering
- average - average or whole frame metering
- custom - custom metering mode defined in the camera tuning file

For more information on defining a custom metering mode, and adjusting region weights in existing metering modes, see the Raspberry Tuning guide for the Raspberry Pi cameras and libcamera (<https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf>).

```
rpicam-hello --metering centre
```

- exposure

Sets the exposure profile. Changing the exposure profile should not affect the image exposure. Instead, different modes adjust gain settings to achieve the same net result.

Accepts the following values:

- sport: short exposure, larger gains
- normal: normal exposure, normal gains
- long: long exposure, smaller gains

You can edit exposure profiles using tuning files. For more information, see the Tuning guide for the Raspberry Pi cameras and libcamera (<https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf>).

```
rpicam-hello --exposure sport
```

- awb

Sets the exposure profile. Changing the exposure profile should not affect the image exposure. Instead, different modes adjust gain settings to achieve the same final result.

Accepts the following values: Available white balance modes:

Mode	Color temperature
auto	2500K ~ 8000K
incandescent	2500K ~ 3000K
tungsten	3000K ~ 3500K
fluorescent	4000K ~ 4700K
indoor	3000K ~ 5000K
daylight	5500K ~ 6500 K
cloudy	7000K ~ 8500K
custom	A custom range defined in the tuning file

These values are only approximate: values could vary according to the camera tuning.

No mode fully disables AWB. Instead, you can fix colour gains with awbgains (https://www.raspberrypi.com/documentation/computers/camera_software.html#awbgains).

For more information on AWB modes, including how to define a custom one, see the Tuning guide for the Raspberry Pi cameras and libcamera (<https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf>).

```
rpicam-hello --awb auto
```

■ awbgains

Sets a fixed red and blue gain value to be used instead of an Auto White Balance (AWB) algorithm. Set non-zero values to disable AWB. Accepts comma-separated numeric input in the following format: <red_gain> , <blue_gain>

```
rpicam-jpeg -o test.jpg --awbgains 1.5,2.0
```

■ denoise

Default value: auto

Sets the denoising mode. Accepts the following values:

- auto: Enables standard spatial denoise. Uses extra-fast colour denoise for video, and high-quality colour denoise for images. Enables no extra colour denoise in the preview window.
- off: Disables spatial and colour denoise.
- cdn_off: Disables colour denoise.
- cdn_fast: Uses fast colour denoise.
- cdn_fast: Uses high-quality colour denoise. Not appropriate for video/viewfinder due to reduced throughput.

Even fast colour denoise can lower framerates. High quality colour denoise significantly lowers framerates.

```
rpicam-hello --denoise off
```

■ tuning-file

Specifies the camera tuning file. The tuning file allows you to control many aspects of image processing, including the Automatic Exposure/Gain Control (AEC/AGC), Auto White Balance (AWB), colour shading correction, colour processing, denoising and more. Accepts a tuning file path as input. For more information about tuning files, see [Tuning Files \(https://www.raspberrypi.com/documentation/computers/camera_software.html#tuning-files\)](https://www.raspberrypi.com/documentation/computers/camera_software.html#tuning-files).

■ autofocus-mode

Default value: default Specifies the autofocus mode. Accepts the following values:

- default: puts the camera into continuous autofocus mode unless lens-position or autofocus-on-capture override the mode to manual
- manual: does not move the lens at all unless manually configured with lens-position
- auto: only moves the lens for an autofocus sweep when the camera starts or just before capture if autofocus-on-capture is also used
- continuous: adjusts the lens position automatically as the scene changes

This option is only supported for certain camera modules.

```
rpicam-hello --autofocus-mode auto
```

- autofocus-range

Default value: normal

Specifies the autofocus range. Accepts the following values:

- normal: focuses from reasonably close to infinity
- macro: focuses only on close objects, including the closest focal distances supported by the camera
- full: focus on the entire range, from the very closest objects to infinity

This option is only supported for certain camera modules.

```
rpicam-hello autofocus-range normal
```

- autofocus-speed

Default value: normal

Specifies the autofocus speed. Accepts the following values:

- normal: changes the lens position at normal speed
- fast: changes the lens position quickly

This option is only supported for certain camera modules.

```
rpicam-hello --autofocus-speed normal
```

- autofocus-window

Specifies the autofocus window within the full field of the sensor. Accepts four decimal values, ranged 0 to 1, in the following format: <x>,<y>,<w>,<h>. Each of these values represents the percentage of available width and height as a decimal between 0 and 1.

These values define the following proportions:

<x>: X coordinates to skip before applying autofocus

<y>: Y coordinates to skip before applying autofocus

<w>:autofocus area width

<h>:autofocus area height

The default value uses the middle third of the output image in both dimensions (1/9 of the total image area).

Examples:

```
rpicam-hello--autofocus-window 0.25,0.25,0.5,0.5
```

selects exactly half of the total number of pixels cropped from the centre of the image (skips the first 25% of X coordinates, skips the first 25% of Y coordinates, uses 50% of the total image width, uses 50% of the total image height).

```
rpicam-hello--autofocus-window 0,0,0.25,0.25
```

selects exactly a quarter of the total number of pixels cropped from the top left of the image (skips the first 0% of X coordinates, skips the first 0% of Y coordinates, uses 25% of the image width, uses 25% of the image height).

This option is only supported for certain camera modules.

- lens-position

Default value: default Moves the lens to a fixed focal distance, normally given in dioptres (units of 1 / distance in metres). Accepts the following spectrum of values:

- 0.0: moves the lens to the "infinity" position
- Any other number: moves the lens to the 1 / number position. For example, the value 2.0 would focus at approximately 0.5m
- normal: move the lens to a default position which corresponds to the hyperfocal position of the lens

Lens calibration is imperfect, so different camera modules of the same model may vary.

- verbose

Alias: -v

Default value: 1 Sets the verbosity level. Accepts the following values:

- 0: no output
- 1: normal output
- 2: verbose output

```
rpicam-hello --verbose 1
```

For more details, click here (https://www.raspberrypi.com/documentation/computers/camera_software.html#advanced-rpicam-apps) for reference.

Resources

Documents

- RPi Camera User Manual (<https://files.waveshare.com/upload/6/61/RPi-Camera-User-Manual.pdf>)

Related Links

- Raspberry Pi Documentation (https://www.waveshare.com/wiki/Raspberry_Pi_Documentation)

FAQ

Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket, Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.

Working Time: 9 AM - 6 PM GMT+8
(Monday to Friday)

Submit Now (<https://service.waveshare.com/>)

*Retrieved from "https://www.waveshare.com/w/index.php?title=IMX296-130_BW_Camera&oldid=106940
(https://www.waveshare.com/w/index.php?title=IMX296-130_BW_Camera&oldid=106940)"*