IMX219-160 Camera

From Waveshare Wiki
Jump to: navigation, search

Introduction

IMX219 Camera, 800 megapixels and 160 FOV. Compatible with Jetson nano. You can also use it with CM3/CM3+ expansion boards like Raspberry Pi Compute Module IO board, Compute Module IO Board Plus, Compute Module POE Board of Waveshare, and the StereoPi board.

More (https://www.waveshare.com/imx219-160-camera.htm)

IMX219-160 Camera

(https://www.waveshare.com/imx219-160-camera.htm?sku=16662)

IMX219 sensor, FOV 160, compatible with jetson nano

Parameters

- 8,000,000 pixels
- Photosensitive chip: IMX219
- Resolution: 3280 x 2464
- Camera parameters:
  - CMOS size: 1/4 inch
  - Aperture (F): 2.35
  - Focal Length: 3.15mm
  - Diagonal Field of View (FOV): 160° degrees
  - Distortion: <14.3%
  - Lens size: 6.5mm x 6.5mm
- 4 screw holes:
  - Available in a fixed position
  - Support 3.3V external power supply

- Dimensions: 25mm × 24mm

Test with Jetson Nano

Hardware connection

- Insert the camera cable with the metal side facing the heatsink into the camera port on the Jetson Nano development kit.
    - Start the Jetson Nano.
- Test camera.
    - Open the terminal (press Ctrl+ALT+T shortcut on the keyboard to open the terminal), and enter the following command to test the camera.

```
DISPLAY=:0.0 nvgstcapture-1.0
```

- Test the dual camera.
    - If you need to test dual cameras, you can add sensor-id=x to select the camera. x can be 0 or 1.

```
#test video0
DISPLAY=:0.0 nvgstcapture-1.0 --sensor-id=0
#test video1
DISPLAY=:0.0 nvgstcapture-1.0 --sensor-id=1
```

- If the camera shooting effect is reddish, you can follow the steps below:

1. Download the camera-override.isp file and extract it to a specific folder:

```
wget https://files.waveshare.com/upload/e/eb/Camera_overrides.tar.gz
tar zxvf Camera_overrides.tar.gz
sudo cp camera_overrides.isp /var/nvidia/nvcam/settings/
```

2. Install files:

```
sudo chmod 664 /var/nvidia/nvcam/settings/camera_overrides.isp
sudo chown root:root /var/nvidia/nvcam/settings/camera_overrides.isp
```

【Notice】

- 12 of NV12 is a number, not a letter.

- The test screen is output to HDMI or DP screen, so when testing, first connect the screen to Jetson Nano.

- Opencv calls the camera

If you want to use the opencv library to call the camera, you can refer to the official jetcam program (https://github.com/NVIDIA-AI-IOT/jetcam)
Or you can use the following method:

```
wget https://files.waveshare.com/upload/2/2e/Simple_camera.zip
unzip Simple_camera.zip
sudo python3  simple_camera.py
```

Test with Compute Module

The IMX219 series camera can be used in the same way as other Raspberry Pi cameras, if you use Raspberry Pi 5, you need to use the supplied 22PIN cable to connect to the IMX219 series camera.

libcamera/libcamera Usage

Introduction

After the Bullseye version, the underlying Raspberry Pi driver for the Raspberry Pi image has been switched from Raspicam to libcamera. Libcamera is an open-source software stack (referred to as a driver later for ease of understanding) that is convenient for third-party porting and developing their own camera drivers. As of December 11, 2023, the official pycamera2 library has been provided for libcamera, making it easier for users to call Python demos.

Call Camera

The libcamera software stack provides six commands for users to preview and test the camera interface.

libcamera-hello

This is a simple "hello world" program that previews the camera and displays the camera image on the screen.
Example

```
libcamera-hello
```

This command will preview the camera on the screen for about 5 seconds. The user can use the "-t <duration>" parameter to set the preview time, where the unit of <duration> is milliseconds. If it is set to 0, it will keep previewing all the time. For example:

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

Tuning File

The libcamera driver of the Raspberry Pi will call a tuning file for different camera modules. The tuning file provides various parameters. When calling the camera, libcamera will call the parameters in the tuning file, and process the image in combination with the algorithm. The final output is the preview screen. Since the libcamera driver can only automatically receive the signal of the chip, the final display effect of the camera will also be affected by the entire module. The use of the tuning file is to flexibly handle the cameras of different modules and adjust to improve the image quality.
If the output image of the camera is not ideal after using the default tuning file, the user can adjust the image by calling the custom tuning file. For example, if you are using the official NOIR version of the camera, the NOIR camera may require different white balance parameters compared with the regular Raspberry Pi Camera V2. In this case, you can switch by calling the tuning file.

```
libcamera-hello --tuning-file /usr/share/libcamera/ipa/raspberrypi/imx219_noir.json
```

Users can copy the default tuning files and modify them according to their needs.
Note: The use of tuning files applies to other libcamera commands, and will not be introduced in subsequent commands.

Preview Window

Most libcamera commands will display a preview window on the screen. Users can customize the title information of the preview window through the --info-text parameter, and can also call some camera parameters through %directives and display them on the window.
For example, if you use HQ Camera: You can display the focal length of the camera on the window through --info-txe "%focus".

```
libcamera-hello --info-text "focus %focus".
```

Note: For more information on parameter settings, please refer to the following chapters.

libcamera-jpeg

libcamera-jpeg is a simple static picture shooting program, different from the complex functions of libcamera-still, libcamera-jpeg code is more concise and has many of the same functions to complete picture shooting.

Take a JPEG image of full pixel

```
libcamera-jpeg -o test.jpg
```

This shooting command will display a preview serial port for about 5 seconds, and then shoot a full-pixel JPEG image and save it as test.jpg.
Users can set the preview time through the -t parameter and can set the resolution of the captured image through --width and --height. E.g.:

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

Exposure control

All libcamera commands allow the user to set the shutter time and gain themselves, such as:

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

This command will capture an image with 20ms exposure and camera gain set to 1.5x. The gain parameter set will first set the analog gain parameter inside the photosensitive chip. If the set gain exceeds the maximum built-in analog gain value of the driver, the maximum analog gain of the chip will be set first, and then the remaining gain multiples will be used as digital gains to take effect.
Note: The digital gain is realized by ISP (image signal processing), not directly adjusting the built-in register of the chip. Under normal circumstances, the default digital gain is close to 1.0, unless there are the following three situations.

1. Overall gain parameter requirements, that is, when the analog gain cannot meet the set gain parameter requirements, the digital gain will be needed for compensation.
2. One of the color gains is less than 1 (the color gain is achieved by digital gain), in this case, the final gain is stabilized at 1/min(red_gain, blue_gain), that is, a uniform digital gain is applied, and is the gain value for one of the color channels (not the green channel).
3. AEC/AGC was modified. If there is a change in AEC/AGC, the digital gain will also change to a certain extent to eliminate any fluctuations, this change will be quickly restored to the "normal" value.

The Raspberry Pi's AEC/AGX algorithm allows the program to specify exposure compensation, which is to adjust the brightness of the image by setting the aperture value, for example:

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

libcamera-still

libcamera-still and libcamera-jpeg are very similar, the difference is that libcamera inherits more functions of raspistill. As before, users can take a picture with the following command.

Test Command

```
libcamera-still -o test.jpg
```

Encoder

libcamea-still supports image files in different formats, can support png and bmp encoding, and also supports saving binary dumps of RGB or YUV pixels as files without encoding or in any image format. If you save RGB or YUV data directly, the program must understand the pixel arrangement of the file when reading such files.

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

Note: The format of image saving is controlled by the -e parameter. If the -e parameter is not called, it will be saved in the format of the output file name by default.

Raw Image Capture

The raw image is the image output by the direct image sensor without any ISP or CPU processing. For color camera sensors, the output format of the raw image is generally Bayer. Note that the raw image is different from the bit-encoded RGB and YUV images we said earlier, and RGB and YUV are also ISP-processed images.

A command to take a raw image:

```
libcamera-still -r -o test.jpg
```

The raw image is generally saved in DNG (Adobe Digital Negative) format, which is compatible with most standard programs, such as dcraw or RawTherapee. The raw image will be saved as a file of the same name with the .dng suffix, for example, if you run the above command, it will be saved as a test.dng, and generate a jpeg file at the same time. The DNG file contains metadata related to image acquisition, such as white balance data, ISP color matrix, etc. The following is the metadata encoding information displayed by the exiftool:

```
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                        : libcamera-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
```

Long Exposure

If we want to take a super long exposure picture, we need to disable AEC/AGC and white balance, otherwise, these algorithms will cause the picture to wait for a lot of frame data when it converges. Disabling these algorithms requires another explicit value to be set. Additionally, the user can skip the preview process with the --immediate setting.
Here is the command to take an image with an exposure of 100 seconds:

```
libcamera-still -o long_exposure.jpg --shutter 100000000 --gain 1 --awbgains 1,1 --immedi
ate
```

Remarks: Reference table for the longest exposure time of several official cameras.

Module	Maximum exposure time (s)
V1 (OV5647)	6
V2 (IMX219)	11.76
V3 (IMX708)	112
HQ (IMX477)	670

libcamera-vid

libcamera-vid is a video recording demo that uses the Raspberry Pi hardware H.264 encoder by default. After the program runs, a preview window will be displayed on the screen, and simultaneously the bitstream encoding will be output to the specified file. For example, record a 10s video.

```
libcamera-vid -t 10000 -o test.h264
```

If you want to view the video, you can use vlc to play it.

```
vlc test.h264
```

Note: The recorded video stream is unpacked. Users can use --save-pts to set the output timestamp to facilitate the subsequent conversion of the bit stream to other video formats.

```
libcamera-vid -o test.h264 --save-pts timestamps.txt
```

If you want to output the mkv file, you can use the following command:

```
mkvmerge -o test.mkv --timecodes 0:timestamps.txt test.h264
```

Encoder

Raspberry Pi supports JPEG format and YUV420 without compression and format:

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

The --codec option sets the output format, not the output file extension.
Use the --segment parameter to split the output file into segments (unit is ms), which is suitable for JPEG files that need to split the JPEG video stream into separate short (about 1ms) JPEG files.

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

UDP Video Streaming Transmission

UDP can be used for video streaming transmission, and the Raspberry Pi runs (server):

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

Among them, <ip-addr> needs to be replaced with the actual client IP address or multicast address. On the client (client), enter the following commands to obtain and display the video stream (you can use one of the two commands):

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

Note: The port needs to be the same as the one you set on the Raspberry Pi.

TCP Video Streaming Transmission

You can use TCP for video streaming Transmission, and the Raspberry Pi runs (server):

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

The client runs:

```
vlc tcp/h264://<ip-addr-of-server>:<port> #Select one of the two commands
ffplay tcp://<ip-addr-of-server>:<port> -vf "setpts=N/30" -fflags nobuffer -flags low_del
ay -framedrop
```

RTSP Video Streaming Transmission

On the Raspberry Pi, vlc is usually used to process the RTSP video stream:

```
libcamera-vid -t 0 --inline -o - | cvlc stream:///dev/stdin --sout '#rtp{sdp=rtsp://:855
4/stream1}' :demux=h264
```

On the playback side, you can run any of the following commands:

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

In all preview commands, if you want to turn off the preview window on the Raspberry Pi, you can use the parameter -n (--nopreview) to set it. Also, pay attention to the setting of the --inline parameter. Changing the setting will force the header information of the video stream to be included in each I (intra) frame. This setting allows the client to correctly parse the video stream even if the video header is lost.

High Frame Rate Mode

If you use the libcamera-vid command to record high frame rate video (generally higher than 60fps) while reducing frame loss, you need to pay attention to the following points:

1. The target level of H.264 needs to be set to 4.2, which can be set with the **--level 4.2** parameter.
2. The color noise reduction function must be turned off, which can be set with the **--denoise cdn_off** parameter.
3. If the set frame rate is higher than 100fps, close the preview window to release more CPU resources and avoid frame loss, which can be set with the **-n** parameter.
4. It is recommended to add the setting force_turbo=1 in the /boot/config.txt file to ensure that the CPU clock will not be limited in the video stream.
5. Adjust the ISP output resolution, use **-width 1280 --height 720** to set the resolution, or set it to a lower resolution, depending on the camera model.
6. If you are using Pi 4 or higher-performance model, you can add the setting gpu_freq=550 or higher in the /boot/config.txt file to overclock the motherboard GPU to achieve higher performance.

For example, record 1280x720 120fps video.

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

libcamera-raw

Libcamera-raw is similar to a video recording demo. The difference is, libcamera-raw records the Bayer format data output by the direct sensor, that is, the raw image data. Libcamera-raw doesn't show a preview window. For example, record a 2-second piece of raw data.

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

The demo will directly dump the raw frame without format information, the program will directly print the pixel format and image size on the terminal, and the user can view the pixel data according to the output data.
By default, the program will save the raw frame as a file, the file is usually large, and the user can divide the file by the --segement parameter.

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

If the memory is large (such as using SSD), libcamera-raw can write the official HQ Camera data (about 18MB per frame) to the hard disk at a speed of about 10 frames per second. To achieve this speed, the demo writes the unformatted raw frames, there is no way to save them as DNG files like libcamera-still does. If you want to ensure that there are no dropped frames, you can use --framerate to reduce the frame rate.

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

Common Command Setting Options

Common command setting options apply to all libcamera commands:

```
--help, -h
```

Print program help information, you can print the available setting options for each program command, and then exit.

```
--version
```

Print the software version, print the software version of libcamera and libcamera-app, then exit.

```
--list-cameras
```

Display the recognized supported cameras. For example:

```
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]
```

According to the printed information, the IMX219 camera has a suffix of 0, and the IM new 477 camera has a suffix of 1. When calling the camera, you can specify the corresponding suffix.

```
--camera
```

Specify the camera, and the corresponding suffix can refer to the print information of the command **--list-camera**.
For example: libcamera-hello -c config.txt
In the setting file, set parameters one line at a time, in the format of key=value:

```
timeout=99000
verbose=
```

```
--config,       -c
```

Under normal circumstances, we can directly set the camera parameters through commands. Here we use the --config parameter to specify the setting file and directly read the setting parameters in the file to set the camera preview effect.

```
--timeout, -t
```

The "-t" option sets the runtime of the libcamera demo. If the video recording command is run, the timeout option sets the recording duration. If the image capture command is run, the timeout sets the preview time before the image is captured and output.

If "timeout" is not set when running the libcamera demo, the default timeout value is 5000 (5 seconds). If the timeout is set to 0, the demo will continue to run.
Example: *libcamera-hello -t 0*

```
--preview, -p
```

"-p" sets the size and position of the preview window (the qualified settings are valid in both X and DRM version windows), and the format is --preview <x. y, w, h> where "x", "y" sets the preview window coordinates, "w" and "h" set the width and length of the preview window. The settings of the preview serial port will not affect the resolution and aspect ratio of the camera image preview. The demo will scale the preview image to display in the preview window and adapt it according to the original image aspect ratio.
Example: *libcamera-hello -p 100,100,500,500*

```
--fullscreen, -f
```

The "-f" option sets the preview window full screen, the preview window and the border in full-screen mode. Like "-p", it does not affect the resolution and aspect ratio, and will automatically adapt.
Example: *libcamera-still -f -o test.jpg*

```
--qt-preview
```

Using the preview window based on the QT framework, this setting is not recommended under normal circumstances, because the preview demo will not use zero-copy buffer sharing and GPU acceleration, which will occupy more resources. The QT preview window supports X forwarding (the default preview program does not).
The Qt preview serial port does not support the "--fullscreen" setting option. If the user wants to use the Qt preview, it is recommended to keep a small preview window to avoid excessive resource usage and affecting the normal operation of the system.
Example: *libcamera-hello --qt-preview*

```
--nopreview, -n
```

Images are not previewed. This setting will turn off the image preview function.
Example: *libcamera-hello -n*

```
--info-text
```

Set the title and information display of the preview window (only available in the X graphics window) using the format --info-text <string>. When calling this option, multiple parameters can be set, and the parameters are usually called in the % command format. The demo will call the corresponding value in the graphics metadata according to the instructions.
If no window info is specified, the default --info-text is set to *"#%frame (%fps fps) exp %exp ag %ag dg %dg"* .
Example: *libcamera-hello --info-test "Focus measure: %focus*
Available parameters:

Instructions	Instructions
%frame	Frame sequence number
%fps	Instantaneous frame rate
%exp	Shutter speed when capturing the image, in ms
%ag	Image analog gain controlled by the sensor chip
%dg	Image value gain controlled by ISP
%rg	Gain of the red component of each pixel
%bg	Gain of the blue component of each pixel
%focus	Corner detection of the image, the larger the value, the clearer the image
%lp	Diopter of the current lens (1/distance in meters)
%afstate	Autofocus state (idle, scanning, focused, failed)

```
--width
--height
```

These two parameters set the width and height of the image respectively. For libcamera-still, libcamera-jpeg, and libcamera-vid commands, these two parameters can set the resolution of the output image/video.
If the libcamera-raw command is used, these two parameters will affect the size of the obtained metadata frame. The camera has a 2 x 2 block reading mode. If the set resolution is smaller than the split mode, the camera will obtain the metadata frame according to the 2 x 2 block size.
libcamera-hello cannot specify the resolution.
Example:
libcamera-vid -o test.h264 --width 1920 --height 1080 Record a 1080p video
libcamera-still -r -o test.jpg --width 2028 --height 1520 Take a 2028 x 1520 JPEG image

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

This setting option is also used to set the resolution of the image, the difference is only the image size of the preview. It does not affect the final output image or video resolution.The size of the preview image will not affect the size of the preview window and it will be adapted according to the window.
Example: *libcamera-hello --viewfinder-width 640 --viewfinder-height 480*

```
--rawfull
```

This setting forces the sensor chip to activate --width and --height settings to output still images and video in full-resolution reading mode. This setting libcamera-hello is invalid. With this setting, the framerate will be affected. In full-resolution mode, frame reading will be slower.
Example: *libcamera-raw -t 2000 --segment 1 --rawfull -o test%03d.raw*
The example command will capture multiple Metadata frames in full-resolution mode. If you are using an HQ camera, the size of each frame is 18MB, and if --rawfull is not set, the HQ camera defaults to 2 x 2 mode, and the data size of each frame is only 4.5MB.

```
--mode
```

This parameter is more general than rawfull. It is used to set the camera mode. When using it, you need to specify the width, height, bit depth and packing mode, and separate them with colons. The set value does not have to be completely accurate, the system will automatically match the closest value, and the bit depth and packing mode can be set (the default is 12 and P means packing).

- **4056:3040:12:P** - 4056x3040 resolution, 12 bits per pixel, packing. Packing means that the raw image data will be packed in the buffer. In this case, two pixels will only occupy 3 bytes, which can save memory.
- **1632:1224:10** - 1632x1224 resolution, 10 bits per pixel, packing by default. In 10-bit packing mode, 4-pixel data will occupy 5 bytes.
- **2592:1944:10:U** - 2592x1944 resolution, 10 bits per pixel, no packing. In the case of unpacking, each pixel will occupy 2 bytes of memory, in this case, the highest 6 bits will be set to 0.
- **3262:2448** - 3264x2448 resolution, 12 bits and packing mode are used by default. However, if the camera model, such as Camera V2 (IMX219), does not support 12-bit mode, the system will automatically select 10-bit mode.

```
--viewfinder-mode       #Specify sensor mode, given as <width>:<height>:<bit-depth>:<pack
ing>
```

The --mode parameter is used to set the camera mode when recording video and shooting still images. If you want to set it when previewing, you can use the --viewfinder-mode parameter.

```
--lores-width
--lores-height
```

These two options set low-resolution images. The low-resolution data stream compresses the image, causing the aspect ratio of the image to change. When using libcamera-vid to record video, if a low resolution is set, functions such as color image denoising will be disabled.
Example: *libcamera-hello --lores-width 224 --lores-height 224*
Note that low-resolution settings are often used in conjunction with image postprocessing, otherwise it has little effect.

```
--hflip #Flip the image horizontally
--vflip #Flip the image vertically
--rotation #Flip the image horizontally or vertically according to the given angle <angle
>
```

These three options are used to flip the image. The parameters of --rotation currently only support 0 and 180, which are equivalent to --hflip and --vflip.
Example: *libcamera-hello --vflip --hflip*

```
--roi #Crop image <x, y, w, h>
```

The --roi parameter allows the user to crop the image area they want according to the coordinates from the complete image provided by the sensor, that is digital scaling, note that the coordinate values should be within the valid range. For example --roi 0, 0, 1, 1 is an invalid instruction.
Example: *libcamera-hello --roi 0.25,0.25,0.5,0.5*
The example command will crop 1/4 of the image from the center of the image.

```
--hdr #Run the camera in HDR mode (supported cameras only)
```

The --hdr parameter is used to set the wide dynamic mode of the camera. This setting will only take effect if the camera supports a wide dynamic range. You can use --list-camera to see if the camera supports hdr mode.

```
--sharpness #Set the sharpness of the image <number>
```

Adjust the sharpness of the image by the value of <number>. If set to 0, sharpening will be applied. If you set a value above 1.0, an extra sharpening amount will be used.
Example: *libcamera-still -o test.jpg --sharpness 2.0*

```
--contrast #Set image contrast <number>
```

Example: *libcamera-still -o test.jpg --contrast 1.5*

```
--brightness #Set image brightness <number>
```

The setting range is -1.0 ~ 1.0
Example: *libcamera-still -o test.jpg --brightness 0.2*

```
--saturation #Set image color saturation <number>
```

Example: *libcamera-still -o test.jpg --saturation 0.8*

```
--ev #Set EV compensation <number>
```

Set the EV compensation of the image in units of aperture, the setting range is -10 ~ 10, the default value is 0. The program runs using the AEC/AGC algorithm.
Example: *libcamera-still -o test.jpg --ev 0.3*

```
--shutter #Set the exposure time, the unit is ms <number>
```

Note: If the frame rate of the camera is too fast, it may not work according to the set shutter time. If this happens, you can try to use --framerate to reduce the frame rate.
Example: *libcamera-hello --shutter 30000*

```
--gain #Set gain value (combination of numerical gain and analog gain) <number>
--analoggain #--gain synonym
```

--analoggain is the same as --gain, the use of analoggain is only for compatibility with raspicam programs.

```
--metering #Set metering mode <string>
```

Set the metering mode of the AEC/AGC algorithm, the available parameters are:

- centre - Center metering (default)
- spot - spot metering
- average - average or full frame metering
- custom - custom metering mode, can be set via tuning file

Example: *libcamera-still -o test.jpg --metering spot*

```
--exposure #Set exposure profile <string>
```

The exposure mode can be set to normal or sport. The report profile for these two modes does not affect the overall exposure of the image, but in the case of sport mode, the program will shorten the exposure time and increase the gain to achieve the same exposure effect.

- sport: Short exposure, larger gains
- normal: Normal expoeure, normal gains
- long: Long exposure, smaller gains

Example: *libcamera-still -o test.jpg --exposure sport*

```
--awb #Set white balance mode <string>
```

Available white balance modes:

Mode	Color Temperature
auto	2500K ~ 8000K
incadescent	2500K ~ 3000K
tungsten	3000K ~3500K
fluorescent	4000K ~ 4700K
indoor	3000K ~ 5000K
daylight	5500K ~ 6500K
cloudy	7000K ~ 8500K
custom	Custom range, set via tuning file

Example: *libamera-still -o test.jpg --awb tungsten*

```
--awbgains #Set a fixed color gain <number,number>
```

Set red and blue gain.
Example: *libcamera-still -o test.jpg --awbgains 1.5, 2.0*

```
--denoise #Set denoising mode <string>
```

Supported denoising modes:

- auto - Default mode, use standard spatial denoising. If it is a video, fast color denoising will be used, and high-quality color denoising will be used when taking still images. Previewing images will not use any color denoising.
- off - Turn off spatial denoising and color denoising.
- cdn_off - Turn off color denoising.
- cdn_fast - Use fast color denoising.
- cdn_hq - Use high-quality color denoising, not suitable for video recording.

Example: *libcamera-vid -o test.h264 --denoise cdn_off*

```
--tuning-file #Specify camera tuning file <string>
```

For more instructions on tuning files, you can refer to official tutorial (https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf).
Example: *libcamera-hello --tuning-file ~/my~camera-tuning.json*

```
--autofocus-mode #Specify the autofocus mode <string>
```

Set the autofocus mode.

- default - By default, the camera will use continuous autofocus mode, unless --lens-position or --autofocus-on-capture manual focus is set.
- manual - Manual focus mode, the focus position can be set by --lens-position.
- auto - Focusing will only be done once when the camera is turned on, and will not be adjusted in other cases. (If you use the libcamera-still command, only when --autofocus-on-capture is used, it will focus once before taking a photo).
- continuous - The camera will automatically adjust the focus position according to the scene changes.

```
--autofocus-range #Specify the autofocus range <string>
```

Set the autofocus range.

- normal -- The default mode, from nearest to infinity.
- macro - The macro mode, only focus on nearby objects.
- full - The full distance mode, adjusted to infinity for the closest object.

```
--autofocus-speed #Specify the autofocus speed <string>
```

Set the focus speed.

- normal - The default item, normal speed.
- fast - The fast focus mode.

```
--autofocus-window   --autofocus-window
```

To display the focus window, you need to set x, y, width and height, and the coordinate value setting is based on the ratio of the image. For example **--autofocus-window 0.25,0.25,0.5,0.5** would set a window that is half the size of the image and centered.

```
--lens-position #Set the lens to a given position <string>
```

Set the focus position.

- 0.0 -- Set the focus position to infinity.
- number -- Set the focus position to 1/number, number is any value you set, for example, if you set 2, it means that it will focus on the position of 0.5m.
- default -- Focus on the default position relative to the hyperfocal distance of the lens.

```
--output, -o #Output file name <string>
```

Set the filename of the output image or video. In addition to setting the file name, you can also specify the output udp or tcp server address to output the image to the server. If you are interested, you can check the relevant setting instructions of the subsequent tcp and udp.
Example: *libcamera-vid -t 100000 -o test.h264*

```
--wrap #Wrap the output file counter <number>
```

Example: *libcamera-vid -t 0 --codec mjpeg --segment 1 --wrap 100 -o image%d.jpg*

```
--flush #Flush the output file immediately
```

The --flush parameter will immediately update each frame of the image to the hard disk at the same time as it is written, reducing latency.
Example: *libcamera-vid -t 10000 --flush -o test.h264*

Still Image Shooting Setting Parameters

```
--quality, -q           #Set JPEG image quality <0 ~ 100>
--exif, -x              #Add extra EXIF flags
--timelapse             #Time interval of time-lapse photography, the unit is ms
--framestart            #Start value of frame count
--datetime              #Name the output file with date format
--timestamp             #Name the output file with the system timestamp
-- restart              #Set the JPEG restart interval
--keypress, -k          #Set the Enter key photo mode
--signal, -s            #Set the signal to trigger the photo mode
--thumb                 #Set thumbnail parameters <w:h:q>
--ebcoding, -e          #Set the image encoding type: jpg/png/bmp/rgb/yuv420
--raw, -r               #Save raw image
--latest                #Associate symbols to the latest saved file
--autofocus-on-capture  #Set to do a focus action before taking a photo
```

Video Recording Image Setting Parameters

```
--quality, -q   #Set JPEG commands <0 - 100>
--bitrate, -b   #Set H.264 bitrate
--intra, -g     #Set the internal frame period (only supports H.264)
--profile       #Set H.264 configuration
--level         #Set H.264 level
--codec         #Set encoding type h264 / mjpeg / yuv420
--keypress, -k  #Set Enter key to pause and record
--signal, -s    #Set signal pause and record
--initial       #Start the program in the recording or paused state
--split         #Split video and save to another file
--segment       #Split video into multiple video segments
--circular      #Write video to the circular buffer
--inline        #Write header in each I frame (only supports H.264)
--listen        #Wait for a TCP connection
--frames        #Set the number of frames recorded
```

- For more camera setup instructions, please refer to Official Camera Documentation (https://www.raspberrypi.com/documentation/accessories/camera.html#libcamera-and-libcamera-apps).

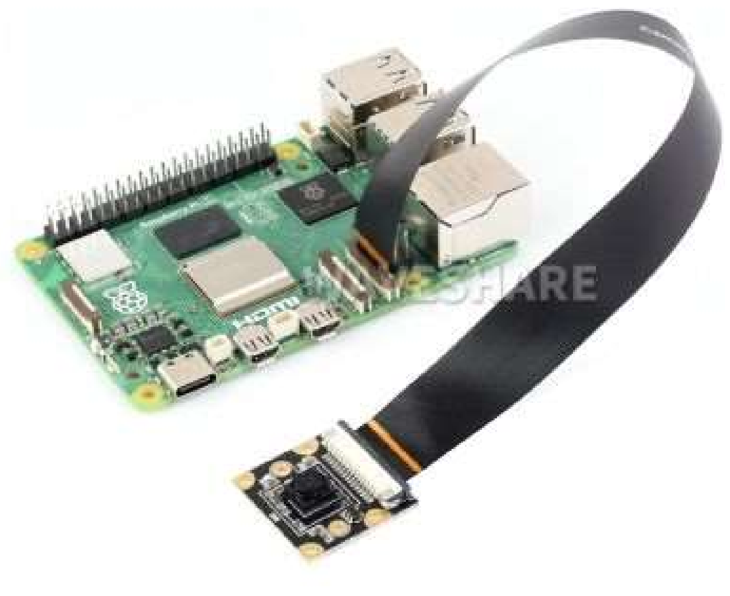
Raspberry Pi User Guide

Hardware Connection

To test the Camera, you need to connect a HDMI display or a DIS display for previewing.
The connectors of the DSI interface (display) and the CSI interface (camera) look the same, please take care of it when you connect the camera. The CSI interface is placed between the Audio jack and the HDMI port. The CSI connector of Pi zero series is beside the Power interface. If you use the Compute Module, please check the actual place of the carrier board.

- Connect to Raspberry Pi 5

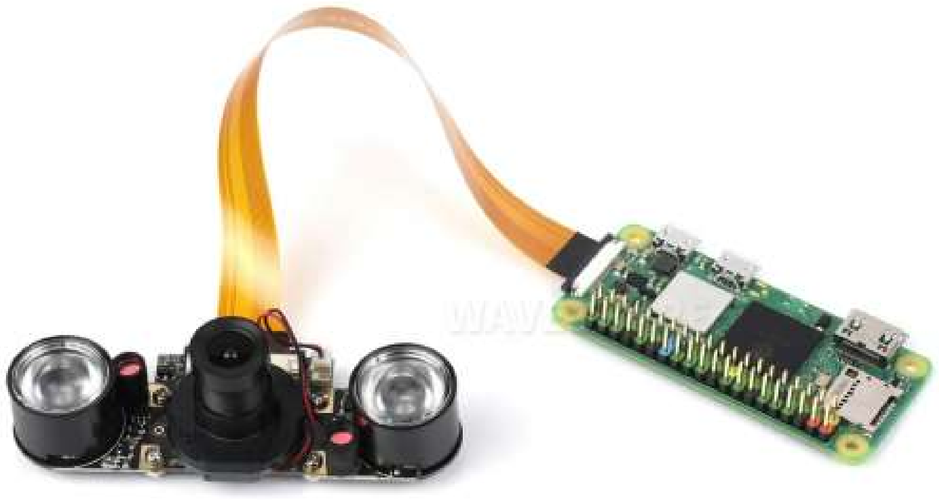
Place the metal surface of the FPC cable towards the wired network port, then connect to the CSI port. The Pi 5 has two CSI ports; either port can be used for connection.



(/wiki/File:Pi5-IMX219-77-details-5.jpg)

- Connect to Raspberry Pi Zero Series

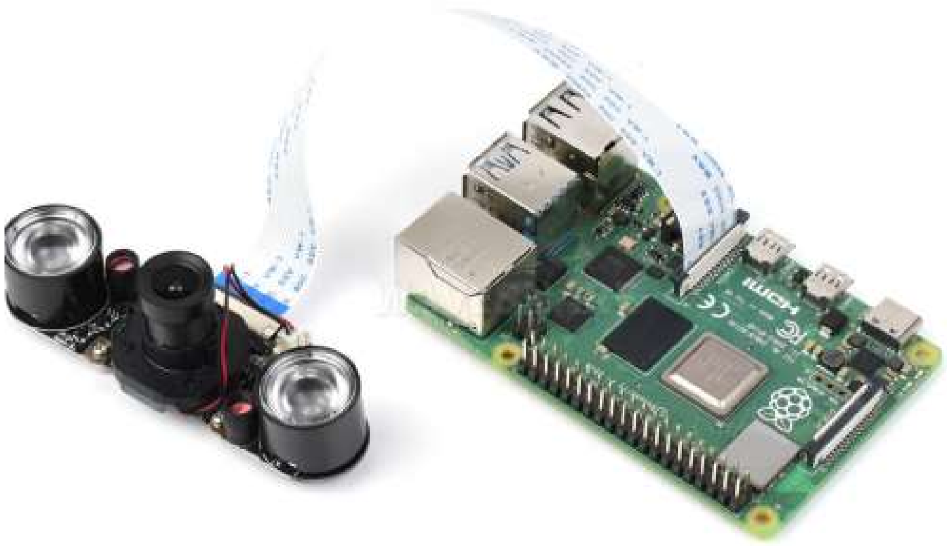
Place the metal surface of the cable downward, then connect it to the CSI port.



(/wiki/File:RPi_Camera_Connect_Zero.png)

- Connect to other Raspberry Pi boards

Place the metal surface of the cable towards the HDMI port and connect to the CSI port.



(/wiki/File:RPi_Camera_Connect_Pi4.png)

About the model

Photosensitive Chip Model	Raspberry Pi Board Model Supported	Driver Type Supported
OV5647	All Raspberry Pi boards	libcamera/Raspicam
OV9281	All Raspberry Pi boards	libcamera
IMX219 (Official Raspberry Pi)	All Raspberry Pi boards	libcamera/Raspicam
IMX219 (Third party)	Raspberry Pi Compute Module	libcamera
IMX290/IMX327	All Raspberry Pi boards	libcamera
IMX378	All Raspberry Pi boards	libcamera
IMX477 (Official Raspberry Pi)	All Raspberry Pi boards	libcamera/Raspicam
IMX477 (Third party)	Raspberry Pi Compute Module	libcamera
IMX519	All Raspberry Pi boards	libcamera (driver required)
IMX708 (Raspberry Pi Camera Module 3)	All Raspberry Pi boards	libcamera
IMX296 (Raspberry Pi Global Camera)	All Raspberry Pi boards	libcamera

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 commands to update the system (network connection is required).

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

If only one camera is invoked, connect the camera to the CAM1 port.
If you do not use an official Raspberry Pi camera, you need to configure the "config.txt " file.
If you use the latest Bookworm system, you need to configure /boot/firmware/config.txt.

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

Find "camera-auto-detect=1" and modify it to "camera_auto_detect=0".

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

Model	Set Statement
OV9281	dtoverlay=ov9281

IMX290/IMX327	dtoverlay=imx290, clock-frequency=37125000
IMX378	dtoverlay=imx378
IMX219	dtoverlay=imx219
IMX477	dtoverlay=imx477
IMX708	dtoverlay=imx708

Note: To connect IMX290 to Raspberry Pi 5, it is necessary to add a json file to the command directory. The operation is as follows:

```
sudo wget https://www.waveshare.net/w/upload/7/7a/Imx290.zip
sudo unzip Imx290.zip
sudo cp imx290.json /usr/share/libcamera/ipa/rpi/pisp
```

Binocular Camera Configuration

- Currently, both the CM4 carrier board and Raspberry Pi 5 support the connection of two cameras.
- If you want to simultaneously connect to two cameras, you can designate the cameras by adding 'cam0' and 'cam1' after the corresponding camera configuration statements.
  - For example, the imx219 is connected to the cam0 interface and the ov5647 camera is connected to the cam1 interface.

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

Test Camera Commands

Enter the Raspberry Pi terminal and enable the camera to preview:

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

If you want to close the preview window, you can directly press the keys "Alt + F4", or click "X" to close. Also, you can return to the terminal interface and press "Ctrl + c" to end the demo.
Note: If using "Camera Module 3", the auto-focus function is enabled.

Test Binocular Camera

- When testing the binocular camera, you need to add "--camera" to specify the camera. If you do not add this parameter, "cam0" is specified by default.

```
sudo libcamera-hello -t 0 --camera 0
sudo libcamera-hello -t 0 --camera 1
```

Introduction

- Check your system version with the command "sudo cat /etc/os-release", find the following related information related to the two images, and then select one:
  - As the Raspberry Pi OS Bookworm system modifies the camera capture application from "libcamera-*" to "rpicam-*", it is recommended to use rpicam because libcamera may be obsoleted in the future even if it is accessible now.
  - If you use Raspberry Pi OS Bullseye system, you can refer to the following libcamera-* tutorial.

Rpicam

When running the latest version of Raspberry Pi OS, **rpicam-apps** installed five basic functionalities. In this case, the official Raspberry Pi camera will be detected and auto-on. You can check whether everything is sound by inputting the following content:

```
rpicam-hello
```

You will see a camera preview window for five seconds.
Note: If running on Raspberry Pi 3 and earlier versions with Bullseye, you need to re-enable Glamor to make the X Windows hardware-accelerated preview window work properly. Enter **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. Refer to the **/boot/firmware/config.txt** file and ensure **dtoverlay=vc4-fkms-v3d** or **dtoverlay=vc4-kms-v3d** is currently active. If you need to change this setting, restart your Raspberry Pi.

rpicam-hello

Equivalent to a camera's "hello world," it starts the camera preview stream and displays it on the screen. You can stop the preview by clicking the window's close button or using Ctrl+C in the terminal.

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

Tunning File

Raspberry Pi's libcamera has tuning files for each different type of camera module. The parameters in these files are passed to the algorithms and hardware to produce the best quality images. Libcamera can automatically determine the image sensor being used, but it cannot automatically determine the entire module, even though the whole module can affect the "tuning." Therefore, it is sometimes necessary to override the default tuning file for a specific sensor.
For example, the NoIR version of the sensor requires different AWB (auto white balance) settings compared to the standard version. Therefore, the IMX219 NoIR used with Pi 4 or earlier devices should be run as follows:

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

Raspberry Pi 5 uses different tuning files in different folders, here you can use:

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

This also means users can copy existing tuning files and make changes according to their preferences, as long as the --tuning-file parameter points to the new version. The --tuning-file parameter, like other command line options, is also applicable to all rpicam-apps.

rpicam-jpeg

rpicam-jpeg is a simple still image capture application.
To capture a full-resolution JPEG image, use the following command. This will display a preview for approximately five seconds and then capture the full-resolution JPEG image to a file named test.jpg:

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

The -t <duration> option can be used to change the length of time the preview is displayed, and the --width and --height options will change the resolution of the captured still image. For example:

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

Exposure Control

rpicam-apps allows users to operate the camera at a fixed shutter speed and gain. Capture images with an exposure time of 20ms and a gain of 1.5x, where the gain serves as analog gain within the sensor until reaching the maximum analog gain allowed by the sensor driver demo, with the remain gain used as digital gain.

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

Raspberry Pi's AEC/AGC algorithm allows applications to specify exposure compensation, enabling the image to be darkened or brightened by a given 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

Digital gain is applied by ISP rather than the sensor. Its value always keep approximate to 1.0, unless:

- The requested total gain (through the --gain option or via the exposure profile in the camera tuning) exceeds the gain that can be used as analog gain within the sensor. Only the required additional gain will be used as digital gain.
- One of the color gains is less than 1 (note that color gains are also used as digital gains). In this case, the reported digital gain will stabilize at 1/min (red gain, blue gain). This means that one of the color channels (other than the green channel) has unit digital gain applied.
- AEC/AGC is changing. When AEC/AGC adjusts, digital gain usually fluctuates to try to eliminate any instability, but it quickly returns to normal values.

rpicam-still

Simulates many of the original features of the raspistill application.

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

Editor

rpicam-still allows files to be saved in many different formats. It supports png and bmp encoding. It also allows saving files as binary storage of RGB or YUV pixels, with no encoding or file format. In the latter case, the application reading the file must learn 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 that 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 filename.

Raw Image Capture

Raw images are directly generated by the image sensor. Before processing with ISP or any CPU kernel, the images generally are in Bayer format for color image sensor. Please note that the raw image is very different from the processed but uncoded RGB or YUV images we have seen before.
Obtain raw image:

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

Here, "-r" (——raw) represents the captured raw image and JPEG. In fact, raw images are the raw images from which JPEGs are generated. Raw images are saved in DNG (Adobe Digital Negative) format and are compatible with many standard applications such as draw or RawTherapee. Raw images are saved to files with the same name but with the extension . ng file, so it is "test.dng".
These DNG files contain metadata related to image capture, including black levels, white balance information, and the color matrix used by the ISP to generate JPEGs. This makes these DNG files more convenient for later "manual" raw conversions using some of the tools mentioned above. Use exiftool to display all metadata encoded into DNG files:

```
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                        : rpicam-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 note that there is only one calibrated source (the one determined by the AWB algorithm, although it is always labeled "D65"), and the ISO number is divided by 100 to give the analog gain being used.

Ultra-long exposure

To capture long exposure images, disable AEC/AGC and AWB as these algorithms will force the user to wait for many frames at convergence.
The way to disable them is to provide explicit values. In addition, the entire preview phase of the capture can be skipped using the --immediate option.
Thus, to perform a 100-second exposure capture, use the ;-immediate option.

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

For reference, the maximum exposure times for 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 devices. Rpicam-vid displays a preview window and writes the encoded bitstream to the specified output. This produces an unpacked video bitstream, which is not packaged in any type of container (e.g. mp4 file) format.

- rpicam-vid uses H.264 encode.

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 file.

```
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 and 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. segment option splits the output file into segment-sized chunks (in milliseconds). By specifying very short (1 millisecond) segments, this makes it easy to break up a motion JPEG stream into individual JPEG files. For example, the following command combines 1 millisecond segments 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 video

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

- Set the H.264 target level to 4.2 with the parameter --level 4.2.
- Disable software color noise reduction processing by setting denoise option to cdn_off.
- Disable nopreview's display window to free up some extra 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 force_turbo (https://www.raspberrypi.com/documentation/computers/config_txt.html#force_turbo) document.
- Adjust the output resolution parameter of ISP as "--width 1280 --height 720" or lower to realize the target frame rate.
- For Raspberry Pi 4, you can add "gpu_freq=550" or higher frame rate at /boot/firmware/config.txt to overclock the GPU for better performance. For more details, please refer to overclock (https://www.raspberrypi.com/documentation/computers/config_txt.html#overclocking) document.

The following commands demonstrate how to realize 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
```

Integrates Libav and picam-vid

Rpicam-vid can encode audio and video streams using ffmpeg/libav codec backends. 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 stream video over UDP using the Raspberry Pi as a server, replace the < IP -addr> placeholder with the IP address of the client or multicast address and the <port> placeholder with the port you want to use for streaming using the following commands.

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

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

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

Or stream using ffplay on the client with the following command:

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

TCP

Video can also be transferred over TCP. Using a Raspberry Pi as a server:

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

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

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

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

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

RTSP

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

```
rpicam-vid -t 0 --inline -o - | cvlc stream:///dev/stdin --sout '#rtp{sdp=rtsp://:8554/st
ream1}' :demux=h264
```

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

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

Or stream using VLC on the client with the following command:

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

If you need to close the preview window on the server, use the nopreview command.
The use of inline flags forces stream header information into each inner frame, which helps the client understand the stream if it misses the beginning.

rpicam-raw

rpicam-raw records video as raw Bayer frames directly from the sensor. It does not display a preview window. To record two seconds of raw clips to a file named test.raw, execute the following command:

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

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

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

With a fast storage device, rpicam-raw can write 18MB of 12-megapixel HQ camera frames to disk at 10fps. rpicam-raw is unable to format the output frames as DNG files; to achieve this, use the rpicam-still to avoid dropped frames with a frame rate option of less than 10:

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

For more details about raw format, you can refer to mode (https://www.raspberrypi.com/documentation/computers/camera_software.html#mode).

rpicam-detect

Note: Raspberry Pi operating system does not include rpicam-detect. If you have installed TensorFlow Lite, you can build rpicam-detect. For more information, you can refer to rpicam-apps (https://www.raspberrypi.com/documentation/computers/camera_software.html#build-libcamera-and-rpicam-apps). Do not forget to pass -DENABLE_TFLITE=1 when running cmake.
rpicam-detect displays a preview window, and use Google MobileNet v1 SSD (Single Shot Detector) neural network to monitor the content. After training the neural network, you can use Coco data set to identify about 80 class objects. Rpicam-detect can recognize human, car, cat and so on.
Whenever rpicam-detect detects a target object, it captures a full resolution JPEG. then returns to monitor preview mode.
For more information about model usage, please refer to TensorFlow Lite object detector (https://www.raspberrypi.com/documentation/computers/camera_software.html#object_detect_tf-stage). For example, when you hang out, you can monitor 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 setting

- --help -h prints all options and briefly introduces each option.

```
rpicam-hello -h
```

- --version output the string of libcamera and rpicam-apps.

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

Output example:

```
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 available sensor modes.

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

Sensor mode identifiers have the following form:

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

Cropping is specified in the native sensor pixel (even in pixel segmentation mode) as (<x>, <y>)/<Width>×<Height>. (x, y) specifies the position of the width × height crop window in the sensor array.
For example, the following output displays the information of the IMX219 sensor with index=0 and the IMX477 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 to be used. Specify an index from the list of available cameras.

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

- --config -c, specify a file, which includes the command parameter option and value. Generally, the file is named as "example_configuration.txt" file, specify options and values as key-value pairs, one option per line:

```
timeout=99000
verbose=
```

Note: Omit the prefix -- which is normally used on the command line with arguments. For flags with missing values, such as verbose in the example above, the trailing = must be included.
Then you can run the following command to specify a timeout of 99000 milliseconds and detailed output:

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

- --time -t, default delay of 5000 milliseconds.

```
rpicam-hello -t
```

Specifies how long the application will run before closing. This applies to video recording and preview windows. When capturing a still image, the application displays a preview window with a timeout of milliseconds before capturing the output image.

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

- --preview sets the position (x,y coordinates) and size (w,h dimensions) of the desktop or DRM preview window. Does not affect the resolution or aspect ratio of the image requested from the camera.

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

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

- --fullscreen -f, Force the preview window to use the entire screen with no border or title bar. Scales the image to fit the entire screen. Do not accept values.

```
rpicam-hello -f
```

- --qt-preview uses the Qt preview window, which consumes more resources than the other options, but supports X-window forwarding. Incompatible with the fullscreen flag. Do not accept values.

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

- --nopreview Causes the application not to display the preview window. Do not accept the value.

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

- --info-text

Default value: "#%frame (%fps fps) exp %exp ag %ag dg %dg"
When run in a desktop environment, the supplied string is set to the title of the preview window. The following image metadata replacements are supported:

Command	Description
%frame	frame sequence number
%fps	instantaneous frame rate (IFR)
%exp	Shutter speed in ms when capturing an image
%ag	Image analog gain controlled by photoreceptor chip
%dg	Numerical image gain controlled by ISP
%rg	Gain of the red component per pixel point
%bg	Gain of the blue component per pixel point
%focus	The corner metrics of an image, with larger values indicating a clearer image.
%lp	The current lens diopter (distance in meters^-1)
%afstate	Auto-focus (idle, scanning, focused, failed)

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

- --width
- --height

Each accepts a single number to define the size (in pixels) of the captured image.
For rpicam-still, rpicam-jpeg and rpicam-vid, specify the resolution.
For rpicam-raw, specify the resolution of the raw frame. For cameras with 2×2 bin readout mode, specify a resolution equal to or less than the bin mode to capture 2×2 bin raw frames.
It does not work for rpicam-hello.
Record 1080p video:

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

Capture JPEGs at 2028×1520 resolution. if used with an HQ camera, 2×2 bin mode is used, so the raw file (test. ng) contains the original 2028×1520 Bayer image.

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

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

Each parameter can accept a number that defines the size of the image displayed in the preview window in pixels. It does not affect the size of the preview window, as the image will be resized to fit. It does not affect the captured still images or videos.

```
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>, If the values provided do not match exactly, the system will select the closest available option for the sensor. Packed(P) or unpacked(U) can be used, affecting the format of the stored video and still images, but not the format of the frames passed to the preview window.

Bit-depth and packing are optional. Bit-depth default is 12, Packing default is P(packed).
See list-cameras for information on the bit depths, resolutions, and packing options available for the sensor.
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 same as the mode option, but it applies to the data passed to the preview window. For more information, you can refer to mode (https://www.raspberrypi.com/documentation/computers/camera_software.html#mode) document.
- --lores-width and --lores-height

Provides a second stream of low-resolution images from the camera, scaled down to the specified dimensions. Each accepts a number to define the dimension (in pixels) of the low-resolution stream. Can be used in preview and video modes. Does not provide static capture. For rpicam-vid, disable additional color denoising. It is useful for image post-processing (https://www.raspberrypi.com/documentation/computers/camera_software.html#post-processing-with-rpicam-apps) nudge analysis.

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

- --hflip flip the image, Flips the image horizontally. Does not accept values.

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

- --vflip Flips the image vertically. Does not accept values.

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

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

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

- --roi crops the image extracted from the sensor's full domain. Four decimal values ranging from 0 to 1 are accepted in the following format <x>, <y>, <w>, and <h>. Each of these values is expressed as a decimal between 0 and 1 as a percentage of the available width and height.

These values define the following ratios:
<x>: The x-coordinate to skip before extracting the image.
<y>: The y-coordinate to skip before extracting the image.
<w>: The width of the image to be extracted.
<h>: The height of the image to be extracted.
Defaults to 0,0,1,1 (starting at the first X-coordinate and the first Y-coordinate, using 100% of the image width and using 100% of the image height).
For example:
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 (skip the first 25% of the X coordinates, skip the first 25% of the Y coordinates, use 50% of the total width of the image, and use 50% of the total height of the image).
rpicam-hello --roi 0,0,0.25,0.25 selects one-quarter of the total number of pixels cropped from the upper left corner of the image (skips the first 0% of the X coordinate, skips the first 0% of the Y coordinate, uses 25% of the image width, and uses 25% of the image height).

- --hdr default value: turn off, run the camera on HDR mode. If no value is passed, auto is assumed. accepts one of the following values.
  - off -Disable HDR.
  - auto -Enable HDR on supported devices. use the sensor's built-in HDR mode if available. If the sensor does not have a built-in HDR mode, use the onboard HDR mode (if available).
  - single-exp enable HDR on the supported device. If it is available, use the built-in HDR mode of the sensor. If there is no built-in HDR mode on the sensor, use the onboard HDR mode (if it is available).

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

Use the onboard HDR mode, if available, even if the sensor has a built-in HDR mode. If onboard HDR mode is not available, disable HDR.
Raspberry Pi 5 and later devices have an onboard HDR mode.
To check the build-in HDR mode of the sensor, you need to add this option to the camera list.

Camera Control Option

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

- sharpness

Sets the image sharpness. Accepts values in the following ranges.

- - 0.0 is No sharpening applied.
  - Values greater than 0.0 but less than 1.0 apply a sharpening amount less 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. Accepts values in the following ranges.

- - 0.0 applies the minimum contrast.
  - Values greater than 0.0 but less than 1.0 apply contrast less than the default value.
  - 1.0 applies the default contrast.
  - Values greater than 1.0 apply additional contrast.

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

- brightness

Specify the image brightness, added as an offset for all pixels in the output image. Accepts values in the following range:

- - -1.0 applies the minimum brightness (black).
  - 0.0 applies the standard brightness.
  - 1.0 applies the maximum brightness (white).

For more usage, it is recommended to refer to ev (https://www.raspberrypi.com/documentation/computers/camera_software.html#ev).

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

- saturation

Specifies the image color saturation. Accepts values in the following ranges.

-
  - 0.0 applies minimum saturation (grayscale).
  - Values greater than 0.0 but less than 1.0 apply less saturation than the default value.
  - 1.0 applies 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. Accepts a value that controls the target value passed to the automatic exposure/gain control (AEC/AGC) processing algorithm along the following spectrum.

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

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

- shutter

Specifies the exposure time in microseconds using the shutter. When you use this option, the gain can still vary. If the camera is running a frame rate so fast that it does not allow the specified exposure time (for example, a frame rate of 1fps 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 official camera minimum and maximum shutter times, please refer to Canera Hardware Specification (https://www.raspberrypi.com/documentation/accessories/camera.html#hardware-specification). Values higher than the maximum value result in undefined behavior.

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

- gain

The effect of analoggain and gain are the same.
Set the combined analog and digital gain. When the sensor driver can provide the required gain, only the analog gain is used. When the analog gain is maxed out, the ISP applies the digital gain. Accepts the value.
For the list of Analog Gain Limits for Official Cameras, please refer to Camera Hardware Document (https://www.raspberrypi.com/documentation/accessories/camera.html#hardware-specification).
Sometimes the digital gain exceeds 1.0 even when the analog gain limit is not exceeded. This can happen in the following situations.
Any of these color gains are below 1.0, which causes the digital gain to stabilize at 1.0/min (red gain, blue gain). This causes the total digital gain to be applied to any color channel above 1.0 to avoid color shift artifacts.
Slight differences when the automatic exposure/gain control (AEC/AGC) changes.

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

- metering default value:centre

Sets the metering mode for the automatic exposure/gain control (AEC/AGC) algorithm. The following values are accepted.

-
  - centre: Center-weighted measures
  - spot: spot metering for light
  - average: average or full-frame metering
  - custom: self-de: custom metering modes defined in the camera adjustment file.

For more information on defining custom metering modes and adjusting area weights in existing metering modes, please refer to Raspberry Pi camera and libcamera tuning guide (https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf).

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

- exposure

Set the exposure profile. Changing the exposure profile should not affect the image exposure. Instead, different modes adjust the gain settings to achieve the same net result. The following values are accepted:

- - sport: Short exposure, big gains
  - normal: Normal exposure, normal gain
  - long: Long exposure, small gains

You can use the tuning file to edit the exposure config file. For more details, please refer to Raspberry Pi camera and libcamera tuning guide (https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf).

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

- awb

Set the exposure config file. Changing the exposure config file should not affect the image exposure. Instead, different modes adjust the gain settings to achieve the same end result. The following values are accepted. Available white balance modes:

Mode	Color Temperature
auto	2500K ~ 8000K
incadescent	2500K ~ 3000K
tungsten	3000K ~3500K
fluorescent	4000K ~ 4700K
indoor	3000K ~ 5000K
daylight	5500K ~ 6500 K
cloudy	7000K ~ 8500K
custom	Custom ranges, set via tuning file

These values are only approximations: Values may vary depending on camera adjustments. There is no mode to completely disable AWB. However, you can use awbgains (https://www.raspberrypi.com/documentation/computers/camera_software.html#awbgains) to correct color gain.
For more information on AWB modes, including how to define custom modes. Please refer to Raspberry Pi camera and libcamera tuning guide (https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf).

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

- awbgains

Set a fixed red and blue gain value to replace the automatic white balance (AWB) algorithm. Set a non-zero value to disable AWB. Accepts comma-separated numeric input in the following format:<red_gain> and <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: Enable standard spatial noise reduction. Use super-fast color denoising for videos and high-quality color denoising for images. Do not enable additional color noise in the preview window.
  - off: Close space and color denoising.
  - cdn_off: Disable color noise.
  - cdn_fast: Use quick color denoising.
  - cdn_hq: Uses high quality color denoising. Not suitable for video/viewfinder due to reduced throughput.

Even fast color noise reduction reduces the frame rate. High-quality color noise reduction significantly reduces the frame rate.

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

- tuning-file

Specify the camera tuning file. Tuning files allow you to control many aspects of image processing, including automatic exposure/gain control (AEC/AGC), automatic white balance (AWB), color shading correction, color processing, denoising, and more. Accepts the tuning file path as input. For more details about tuning file, please refer to tuning file (https://www.raspberrypi.com/documentation/computers/camera_software.html#tuning-files).

- autofocus-mode

Default value: default Specifies the autofocus mode. The following values are accepted:

- - default: Put the camera in continuous autofocus mode unless the lens position or autofocus capture overrides manual mode
  - manual: Does not move the lens unless the lens position is manually configured.
  - auto: Only move the lens for autofocus scanning when the camera is started or before capturing, if the autofocus capture function is also used.
  - continuous: Automatically adjusts the lens position 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. The following values are accepted.

- - normal: Focus range is fairly close to infinity.
  - macro: Focus only on close objects, including the closest focal length supported by the camera.
  - full: Focuses the entire range, from the nearest object 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. The following values are accepted.

- - normal: Changing the lens position at normal speed.
  - fast: Quick change of lens position.

This option is only supported for certain camera modules.

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

- autofocus-window

Specifies the autofocus window over the full range of the sensor. Accepts four decimal values ranging from 0 to 1 in the following format: <x>, <y>, <w> and <h>. Each of these values is expressed as a decimal between 0 and 1 as a percentage of the available width and height.
These values define the following ratios:
<x>: The x-coordinate to skip before applying autofocus.
<y>: The y-coordinate to skip before applying autofocus.
<w>: AF area width.
<h>: AF area height.
The default value uses the middle third of the output image (1/9 of the total image area) in both dimensions.
For example:

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

Select half of the total number of pixels cropped from the center of the image (skip the first 25% of the X coordinates, skip the first 25% of the Y coordinates, use 50% of the total width of the image, use 50% of the total height of the image).

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

Select a quarter of the total number of pixels cropped from the upper left corner of the image (skip the first 0% of the X coordinate, skip the first 0% of the Y coordinate, use 25% of the width of the image, use 25% of the height of the image).
This option is only supported for certain camera modules.

- lens-position

Default value: default Moves the lens to a fixed focal length, usually expressed in diopter (in units of 1/meter distance). The following range of values is accepted.

- - 0.0: Move the camera to the "infinity" position.
  - Any other number:Move the lens to the 1 / number position. For example, a value of 2.0 will focus at approximately 0.5m.
  - normal: Move the lens to the default position corresponding to the lens hyperfocal position.

Lens calibration is not perfect, so it may vary from camera module to camera module for the same model.

- verbose

Nickname: -v
Default value: 1 Sets the level of detail. Accepts the following values:

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

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

For more details, you can refer to here (https://www.raspberrypi.com/documentation/computers/camera_software.html#advanced-rpicam-apps).

libcamera/libcamera Usage

Introduction

After the Bullseye version, the underlying Raspberry Pi driver for the Raspberry Pi image has been switched from Raspicam to libcamera. Libcamera is an open-source software stack (referred to as a driver later for ease of understanding) that is convenient for third-party porting and developing their own camera drivers. As of December 11, 2023, the official pycamera2 library has been provided for libcamera, making it easier for users to call Python demos.

Call Camera

The libcamera software stack provides six commands for users to preview and test the camera interface.

libcamera-hello

This is a simple "hello world" program that previews the camera and displays the camera image on the screen.

Example

```
libcamera-hello
```

This command will preview the camera on the screen for about 5 seconds. The user can use the "-t <duration>" parameter to set the preview time, where the unit of <duration> is milliseconds. If it is set to 0, it will keep previewing all the time. For example:

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

Tuning File

The libcamera driver of the Raspberry Pi will call a tuning file for different camera modules. The tuning file provides various parameters. When calling the camera, libcamera will call the parameters in the tuning file, and process the image in combination with the algorithm. The final output is the preview screen. Since the libcamera driver can only automatically receive the signal of the chip, the final display effect of the camera will also be affected by the entire module. The use of the tuning file is to flexibly handle the cameras of different modules and adjust to improve the image quality.

If the output image of the camera is not ideal after using the default tuning file, the user can adjust the image by calling the custom tuning file. For example, if you are using the official NOIR version of the camera, the NOIR camera may require different white balance parameters compared with the regular Raspberry Pi Camera V2. In this case, you can switch by calling the tuning file.

```
libcamera-hello --tuning-file /usr/share/libcamera/ipa/raspberrypi/imx219_noir.json
```

Users can copy the default tuning files and modify them according to their needs.
Note: The use of tuning files applies to other libcamera commands, and will not be introduced in subsequent commands.

Preview Window
Most libcamera commands will display a preview window on the screen. Users can customize the title information of the preview window through the --info-text parameter, and can also call some camera parameters through %directives and display them on the window.
For example, if you use HQ Camera: You can display the focal length of the camera on the window through --info-txe "%focus".

```
libcamera-hello --info-text "focus %focus".
```

Note: For more information on parameter settings, please refer to the following chapters.

libcamera-jpeg

libcamera-jpeg is a simple static picture shooting program, different from the complex functions of libcamera-still, libcamera-jpeg code is more concise and has many of the same functions to complete picture shooting.
Take a JPEG image of full pixel

```
libcamera-jpeg -o test.jpg
```

This shooting command will display a preview serial port for about 5 seconds, and then shoot a full-pixel JPEG image and save it as test.jpg.
Users can set the preview time through the -t parameter and can set the resolution of the captured image through --width and --height. E.g.:

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

Exposure control

All libcamera commands allow the user to set the shutter time and gain themselves, such as:

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

This command will capture an image with 20ms exposure and camera gain set to 1.5x. The gain parameter set will first set the analog gain parameter inside the photosensitive chip. If the set gain exceeds the maximum built-in analog gain value of the driver, the maximum analog gain of the chip will be set first, and then the remaining gain multiples will be used as digital gains to take effect.

Note: The digital gain is realized by ISP (image signal processing), not directly adjusting the built-in register of the chip. Under normal circumstances, the default digital gain is close to 1.0, unless there are the following three situations.

1. Overall gain parameter requirements, that is, when the analog gain cannot meet the set gain parameter requirements, the digital gain will be needed for compensation.
2. One of the color gains is less than 1 (the color gain is achieved by digital gain), in this case, the final gain is stabilized at 1/min(red_gain, blue_gain), that is, a uniform digital gain is applied, and is the gain value for one of the color channels (not the green channel).
3. AEC/AGC was modified. If there is a change in AEC/AGC, the digital gain will also change to a certain extent to eliminate any fluctuations, this change will be quickly restored to the "normal" value.

The Raspberry Pi's AEC/AGX algorithm allows the program to specify exposure compensation, which is to adjust the brightness of the image by setting the aperture value, for example:

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

libcamera-still

libcamera-still and libcamera-jpeg are very similar, the difference is that libcamera inherits more functions of raspistill. As before, users can take a picture with the following command.

Test Command

```
libcamera-still -o test.jpg
```

Encoder

libcamea-still supports image files in different formats, can support png and bmp encoding, and also supports saving binary dumps of RGB or YUV pixels as files without encoding or in any image format. If you save RGB or YUV data directly, the program must understand the pixel arrangement of the file when reading such files.

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

Note: The format of image saving is controlled by the -e parameter. If the -e parameter is not called, it will be saved in the format of the output file name by default.

Raw Image Capture

The raw image is the image output by the direct image sensor without any ISP or CPU processing. For color camera sensors, the output format of the raw image is generally Bayer. Note that the raw image is different from the bit-encoded RGB and YUV images we said earlier, and RGB and YUV are also ISP-processed images.

A command to take a raw image:

```
libcamera-still -r -o test.jpg
```

The raw image is generally saved in DNG (Adobe Digital Negative) format, which is compatible with most standard programs, such as dcraw or RawTherapee. The raw image will be saved as a file of the same name with the .dng suffix, for example, if you run the above command, it will be saved as a test.dng, and generate a jpeg file at the same time. The DNG file contains metadata related to image acquisition, such as white balance data, ISP color matrix, etc. The following is the metadata encoding information displayed by the exiftool:

```
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                        : libcamera-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
```

Long Exposure

If we want to take a super long exposure picture, we need to disable AEC/AGC and white balance, otherwise, these algorithms will cause the picture to wait for a lot of frame data

when it converges. Disabling these algorithms requires another explicit value to be set. Additionally, the user can skip the preview process with the --immediate setting.
Here is the command to take an image with an exposure of 100 seconds:

```
libcamera-still -o long_exposure.jpg --shutter 100000000 --gain 1 --awbgains 1,1 --immedi
ate
```

Remarks: Reference table for the longest exposure time of several official cameras.

Module	Maximum exposure time (s)
V1 (OV5647)	6
V2 (IMX219)	11.76
V3 (IMX708)	112
HQ (IMX477)	670

libcamera-vid

libcamera-vid is a video recording demo that uses the Raspberry Pi hardware H.264 encoder by default. After the program runs, a preview window will be displayed on the screen, and simultaneously the bitstream encoding will be output to the specified file. For example, record a 10s video.

```
libcamera-vid -t 10000 -o test.h264
```

If you want to view the video, you can use vlc to play it.

```
vlc test.h264
```

Note: The recorded video stream is unpacked. Users can use --save-pts to set the output timestamp to facilitate the subsequent conversion of the bit stream to other video formats.

```
libcamera-vid -o test.h264 --save-pts timestamps.txt
```

If you want to output the mkv file, you can use the following command:

```
mkvmerge -o test.mkv --timecodes 0:timestamps.txt test.h264
```

Encoder

Raspberry Pi supports JPEG format and YUV420 without compression and format:

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

The --codec option sets the output format, not the output file extension.
Use the --segment parameter to split the output file into segments (unit is ms), which is suitable for JPEG files that need to split the JPEG video stream into separate short (about 1ms) JPEG files.

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

UDP Video Streaming Transmission

UDP can be used for video streaming transmission, and the Raspberry Pi runs (server):

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

Among them, <ip-addr> needs to be replaced with the actual client IP address or multicast address. On the client (client), enter the following commands to obtain and display the video stream (you can use one of the two commands):

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

Note: The port needs to be the same as the one you set on the Raspberry Pi.

TCP Video Streaming Transmission

You can use TCP for video streaming Transmission, and the Raspberry Pi runs (server):

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

The client runs:

```
vlc tcp/h264://<ip-addr-of-server>:<port> #Select one of the two commands
ffplay tcp://<ip-addr-of-server>:<port> -vf "setpts=N/30" -fflags nobuffer -flags low_del
ay -framedrop
```

RTSP Video Streaming Transmission

On the Raspberry Pi, vlc is usually used to process the RTSP video stream:

```
libcamera-vid -t 0 --inline -o - | cvlc stream:///dev/stdin --sout '#rtp{sdp=rtsp://:855
4/stream1}' :demux=h264
```

On the playback side, you can run any of the following commands:

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

In all preview commands, if you want to turn off the preview window on the Raspberry Pi, you can use the parameter -n (--nopreview) to set it. Also, pay attention to the setting of the --inline parameter. Changing the setting will force the header information of the video stream to be included in each I (intra) frame. This setting allows the client to correctly parse the video stream even if the video header is lost.

High Frame Rate Mode

If you use the libcamera-vid command to record high frame rate video (generally higher than 60fps) while reducing frame loss, you need to pay attention to the following points:

1. The target level of H.264 needs to be set to 4.2, which can be set with the **--level 4.2** parameter.
2. The color noise reduction function must be turned off, which can be set with the **--denoise cdn_off** parameter.
3. If the set frame rate is higher than 100fps, close the preview window to release more CPU resources and avoid frame loss, which can be set with the **-n** parameter.
4. It is recommended to add the setting force_turbo=1 in the /boot/config.txt file to ensure that the CPU clock will not be limited in the video stream.
5. Adjust the ISP output resolution, use **-width 1280 --height 720** to set the resolution, or set it to a lower resolution, depending on the camera model.
6. If you are using Pi 4 or higher-performance model, you can add the setting gpu_freq=550 or higher in the /boot/config.txt file to overclock the motherboard GPU to achieve higher performance.

For example, record 1280x720 120fps video.

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

libcamera-raw

Libcamera-raw is similar to a video recording demo. The difference is, libcamera-raw records the Bayer format data output by the direct sensor, that is, the raw image data. Libcamera-raw doesn't show a preview window. For example, record a 2-second piece of raw data.

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

The demo will directly dump the raw frame without format information, the program will directly print the pixel format and image size on the terminal, and the user can view the pixel data according to the output data.
By default, the program will save the raw frame as a file, the file is usually large, and the user can divide the file by the --segement parameter.

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

If the memory is large (such as using SSD), libcamera-raw can write the official HQ Camera data (about 18MB per frame) to the hard disk at a speed of about 10 frames per second. To achieve this speed, the demo writes the unformatted raw frames, there is no way to save them as DNG files like libcamera-still does. If you want to ensure that there are no dropped frames, you can use --framerate to reduce the frame rate.

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

Common Command Setting Options

Common command setting options apply to all libcamera commands:

```
--help, -h
```

Print program help information, you can print the available setting options for each program command, and then exit.

```
--version
```

Print the software version, print the software version of libcamera and libcamera-app, then exit.

```
--list-cameras
```

Display the recognized supported cameras. For example:

```
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]
```

According to the printed information, the IMX219 camera has a suffix of 0, and the IM new 477 camera has a suffix of 1. When calling the camera, you can specify the corresponding suffix.

```
--camera
```

Specify the camera, and the corresponding suffix can refer to the print information of the command **--list-camera**.
For example: libcamera-hello -c config.txt
In the setting file, set parameters one line at a time, in the format of key=value:

```
timeout=99000
verbose=
```

```
--config,       -c
```

Under normal circumstances, we can directly set the camera parameters through commands. Here we use the --config parameter to specify the setting file and directly read the setting parameters in the file to set the camera preview effect.

```
--timeout, -t
```

The "-t" option sets the runtime of the libcamera demo. If the video recording command is run, the timeout option sets the recording duration. If the image capture command is run, the timeout sets the preview time before the image is captured and output.

If "timeout" is not set when running the libcamera demo, the default timeout value is 5000 (5 seconds). If the timeout is set to 0, the demo will continue to run.
Example: *libcamera-hello -t 0*

```
--preview, -p
```

"-p" sets the size and position of the preview window (the qualified settings are valid in both X and DRM version windows), and the format is --preview <x. y, w, h> where "x", "y" sets the preview window coordinates, "w" and "h" set the width and length of the preview window. The settings of the preview serial port will not affect the resolution and aspect ratio of the camera image preview. The demo will scale the preview image to display in the preview window and adapt it according to the original image aspect ratio.
Example: *libcamera-hello -p 100,100,500,500*

```
--fullscreen, -f
```

The "-f" option sets the preview window full screen, the preview window and the border in full-screen mode. Like "-p", it does not affect the resolution and aspect ratio, and will automatically adapt.
Example: *libcamera-still -f -o test.jpg*

```
--qt-preview
```

Using the preview window based on the QT framework, this setting is not recommended under normal circumstances, because the preview demo will not use zero-copy buffer sharing and GPU acceleration, which will occupy more resources. The QT preview window supports X forwarding (the default preview program does not).
The Qt preview serial port does not support the "--fullscreen" setting option. If the user wants to use the Qt preview, it is recommended to keep a small preview window to avoid excessive resource usage and affecting the normal operation of the system.
Example: *libcamera-hello --qt-preview*

```
--nopreview, -n
```

Images are not previewed. This setting will turn off the image preview function.
Example: *libcamera-hello -n*

```
--info-text
```

Set the title and information display of the preview window (only available in the X graphics window) using the format --info-text <string>. When calling this option, multiple parameters can be set, and the parameters are usually called in the % command format. The demo will call the corresponding value in the graphics metadata according to the instructions.

If no window info is specified, the default --info-text is set to *"#%frame (%fps fps) exp %exp ag %ag dg %dg"* .

Example: *libcamera-hello --info-test "Focus measure: %focus*

Available parameters:

Instructions	Instructions
%frame	Frame sequence number
%fps	Instantaneous frame rate
%exp	Shutter speed when capturing the image, in ms
%ag	Image analog gain controlled by the sensor chip
%dg	Image value gain controlled by ISP
%rg	Gain of the red component of each pixel
%bg	Gain of the blue component of each pixel
%focus	Corner detection of the image, the larger the value, the clearer the image
%lp	Diopter of the current lens (1/distance in meters)
%afstate	Autofocus state (idle, scanning, focused, failed)

```
--width
--height
```

These two parameters set the width and height of the image respectively. For libcamera-still, libcamera-jpeg, and libcamera-vid commands, these two parameters can set the resolution of the output image/video.

If the libcamera-raw command is used, these two parameters will affect the size of the obtained metadata frame. The camera has a 2 x 2 block reading mode. If the set resolution is smaller than the split mode, the camera will obtain the metadata frame according to the 2 x 2 block size.

libcamera-hello cannot specify the resolution.

Example:

libcamera-vid -o test.h264 --width 1920 --height 1080 Record a 1080p video

libcamera-still -r -o test.jpg --width 2028 --height 1520 Take a 2028 x 1520 JPEG image

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

This setting option is also used to set the resolution of the image, the difference is only the image size of the preview. It does not affect the final output image or video resolution.The size of the preview image will not affect the size of the preview window and it will be adapted according to the window.
Example: *libcamera-hello --viewfinder-width 640 --viewfinder-height 480*

```
--rawfull
```

This setting forces the sensor chip to activate --width and --height settings to output still images and video in full-resolution reading mode. This setting libcamera-hello is invalid. With this setting, the framerate will be affected. In full-resolution mode, frame reading will be slower.
Example: *libcamera-raw -t 2000 --segment 1 --rawfull -o test%03d.raw*
The example command will capture multiple Metadata frames in full-resolution mode. If you are using an HQ camera, the size of each frame is 18MB, and if --rawfull is not set, the HQ camera defaults to 2 x 2 mode, and the data size of each frame is only 4.5MB.

```
--mode
```

This parameter is more general than rawfull. It is used to set the camera mode. When using it, you need to specify the width, height, bit depth and packing mode, and separate them with colons. The set value does not have to be completely accurate, the system will automatically match the closest value, and the bit depth and packing mode can be set (the default is 12 and P means packing).

- **4056:3040:12:P** - 4056x3040 resolution, 12 bits per pixel, packing. Packing means that the raw image data will be packed in the buffer. In this case, two pixels will only occupy 3 bytes, which can save memory.
- **1632:1224:10** - 1632x1224 resolution, 10 bits per pixel, packing by default. In 10-bit packing mode, 4-pixel data will occupy 5 bytes.
- **2592:1944:10:U** - 2592x1944 resolution, 10 bits per pixel, no packing. In the case of unpacking, each pixel will occupy 2 bytes of memory, in this case, the highest 6 bits will be set to 0.
- **3262:2448** - 3264x2448 resolution, 12 bits and packing mode are used by default. However, if the camera model, such as Camera V2 (IMX219), does not support 12-bit mode, the system will automatically select 10-bit mode.

```
--viewfinder-mode       #Specify sensor mode, given as <width>:<height>:<bit-depth>:<pack
ing>
```

The --mode parameter is used to set the camera mode when recording video and shooting still images. If you want to set it when previewing, you can use the --viewfinder-mode parameter.

```
--lores-width
--lores-height
```

These two options set low-resolution images. The low-resolution data stream compresses the image, causing the aspect ratio of the image to change. When using libcamera-vid to record video, if a low resolution is set, functions such as color image denoising will be disabled.
Example: *libcamera-hello --lores-width 224 --lores-height 224*
Note that low-resolution settings are often used in conjunction with image postprocessing, otherwise it has little effect.

```
--hflip #Flip the image horizontally
--vflip #Flip the image vertically
--rotation #Flip the image horizontally or vertically according to the given angle <angle
>
```

These three options are used to flip the image. The parameters of --rotation currently only support 0 and 180, which are equivalent to --hflip and --vflip.
Example: *libcamera-hello --vflip --hflip*

```
--roi #Crop image <x, y, w, h>
```

The --roi parameter allows the user to crop the image area they want according to the coordinates from the complete image provided by the sensor, that is digital scaling, note that the coordinate values should be within the valid range. For example --roi 0, 0, 1, 1 is an invalid instruction.
Example: *libcamera-hello --roi 0.25,0.25,0.5,0.5*
The example command will crop 1/4 of the image from the center of the image.

```
--hdr #Run the camera in HDR mode (supported cameras only)
```

The --hdr parameter is used to set the wide dynamic mode of the camera. This setting will only take effect if the camera supports a wide dynamic range. You can use --list-camera to see if the camera supports hdr mode.

```
--sharpness #Set the sharpness of the image <number>
```

Adjust the sharpness of the image by the value of <number>. If set to 0, sharpening will be applied. If you set a value above 1.0, an extra sharpening amount will be used.
Example: *libcamera-still -o test.jpg --sharpness 2.0*

```
--contrast #Set image contrast <number>
```

Example: *libcamera-still -o test.jpg --contrast 1.5*

```
--brightness #Set image brightness <number>
```

The setting range is -1.0 ~ 1.0
Example: *libcamera-still -o test.jpg --brightness 0.2*

```
--saturation #Set image color saturation <number>
```

Example: *libcamera-still -o test.jpg --saturation 0.8*

```
--ev #Set EV compensation <number>
```

Set the EV compensation of the image in units of aperture, the setting range is -10 ~ 10, the default value is 0. The program runs using the AEC/AGC algorithm.
Example: *libcamera-still -o test.jpg --ev 0.3*

```
--shutter #Set the exposure time, the unit is ms <number>
```

Note: If the frame rate of the camera is too fast, it may not work according to the set shutter time. If this happens, you can try to use --framerate to reduce the frame rate.
Example: *libcamera-hello --shutter 30000*

```
--gain #Set gain value (combination of numerical gain and analog gain) <number>
--analoggain #--gain synonym
```

--analoggain is the same as --gain, the use of analoggain is only for compatibility with raspicam programs.

```
--metering #Set metering mode <string>
```

Set the metering mode of the AEC/AGC algorithm, the available parameters are:

- centre - Center metering (default)
- spot - spot metering
- average - average or full frame metering
- custom - custom metering mode, can be set via tuning file

Example: *libcamera-still -o test.jpg --metering spot*

```
--exposure #Set exposure profile <string>
```

The exposure mode can be set to normal or sport. The report profile for these two modes does not affect the overall exposure of the image, but in the case of sport mode, the program will shorten the exposure time and increase the gain to achieve the same exposure effect.

- sport: Short exposure, larger gains
- normal: Normal expoeure, normal gains
- long: Long exposure, smaller gains

Example: *libcamera-still -o test.jpg --exposure sport*

```
--awb #Set white balance mode <string>
```

Available white balance modes:

Mode	Color Temperature
auto	2500K ~ 8000K
incadescent	2500K ~ 3000K
tungsten	3000K ~3500K
fluorescent	4000K ~ 4700K
indoor	3000K ~ 5000K
daylight	5500K ~ 6500K
cloudy	7000K ~ 8500K
custom	Custom range, set via tuning file

Example: *libamera-still -o test.jpg --awb tungsten*

```
--awbgains #Set a fixed color gain <number,number>
```

Set red and blue gain.
Example: *libcamera-still -o test.jpg --awbgains 1.5, 2.0*

```
--denoise #Set denoising mode <string>
```

Supported denoising modes:

- auto - Default mode, use standard spatial denoising. If it is a video, fast color denoising will be used, and high-quality color denoising will be used when taking still images. Previewing images will not use any color denoising.
- off - Turn off spatial denoising and color denoising.
- cdn_off - Turn off color denoising.
- cdn_fast - Use fast color denoising.
- cdn_hq - Use high-quality color denoising, not suitable for video recording.

Example: *libcamera-vid -o test.h264 --denoise cdn_off*

```
--tuning-file #Specify camera tuning file <string>
```

For more instructions on tuning files, you can refer to official tutorial (https://datasheets.raspberrypi.com/camera/raspberry-pi-camera-guide.pdf).
Example: *libcamera-hello --tuning-file ~/my~camera-tuning.json*

```
--autofocus-mode #Specify the autofocus mode <string>
```

Set the autofocus mode.

- default - By default, the camera will use continuous autofocus mode, unless --lens-position or --autofocus-on-capture manual focus is set.
- manual - Manual focus mode, the focus position can be set by --lens-position.
- auto - Focusing will only be done once when the camera is turned on, and will not be adjusted in other cases. (If you use the libcamera-still command, only when --autofocus-on-capture is used, it will focus once before taking a photo).
- continuous - The camera will automatically adjust the focus position according to the scene changes.

```
--autofocus-range #Specify the autofocus range <string>
```

Set the autofocus range.

- normal -- The default mode, from nearest to infinity.
- macro - The macro mode, only focus on nearby objects.
- full - The full distance mode, adjusted to infinity for the closest object.

```
--autofocus-speed #Specify the autofocus speed <string>
```

Set the focus speed.

- normal - The default item, normal speed.
- fast - The fast focus mode.

```
--autofocus-window   --autofocus-window
```

To display the focus window, you need to set x, y, width and height, and the coordinate value setting is based on the ratio of the image. For example **--autofocus-window 0.25,0.25,0.5,0.5** would set a window that is half the size of the image and centered.

```
--lens-position #Set the lens to a given position <string>
```

Set the focus position.

- 0.0 -- Set the focus position to infinity.
- number -- Set the focus position to 1/number, number is any value you set, for example, if you set 2, it means that it will focus on the position of 0.5m.
- default -- Focus on the default position relative to the hyperfocal distance of the lens.

```
--output, -o #Output file name <string>
```

Set the filename of the output image or video. In addition to setting the file name, you can also specify the output udp or tcp server address to output the image to the server. If you are interested, you can check the relevant setting instructions of the subsequent tcp and udp.
Example: *libcamera-vid -t 100000 -o test.h264*

```
--wrap #Wrap the output file counter <number>
```

Example: *libcamera-vid -t 0 --codec mjpeg --segment 1 --wrap 100 -o image%d.jpg*

```
--flush #Flush the output file immediately
```

The --flush parameter will immediately update each frame of the image to the hard disk at the same time as it is written, reducing latency.
Example: *libcamera-vid -t 10000 --flush -o test.h264*

Still Image Shooting Setting Parameters

```
--quality, -q           #Set JPEG image quality <0 ~ 100>
--exif, -x              #Add extra EXIF flags
--timelapse             #Time interval of time-lapse photography, the unit is ms
--framestart            #Start value of frame count
--datetime              #Name the output file with date format
--timestamp             #Name the output file with the system timestamp
-- restart              #Set the JPEG restart interval
--keypress, -k          #Set the Enter key photo mode
--signal, -s            #Set the signal to trigger the photo mode
--thumb                 #Set thumbnail parameters <w:h:q>
--ebcoding, -e          #Set the image encoding type: jpg/png/bmp/rgb/yuv420
--raw, -r               #Save raw image
--latest                #Associate symbols to the latest saved file
--autofocus-on-capture  #Set to do a focus action before taking a photo
```

Video Recording Image Setting Parameters

```
--quality, -q   #Set JPEG commands <0 - 100>
--bitrate, -b   #Set H.264 bitrate
--intra, -g     #Set the internal frame period (only supports H.264)
--profile       #Set H.264 configuration
--level         #Set H.264 level
--codec         #Set encoding type h264 / mjpeg / yuv420
--keypress, -k  #Set Enter key to pause and record
--signal, -s    #Set signal pause and record
--initial       #Start the program in the recording or paused state
--split         #Split video and save to another file
--segment       #Split video into multiple video segments
--circular      #Write video to the circular buffer
--inline        #Write header in each I frame (only supports H.264)
--listen        #Wait for a TCP connection
--frames        #Set the number of frames recorded
```

- For more camera setup instructions, please refer to Official Camera Documentation (https://www.raspberrypi.com/documentation/accessories/camera.html#libcamera-and-libcamera-apps).

Resources

Related resources

- Jetson Nano Developer Kit (/wiki/Jetson_Nano_Developer_Kit)

FAQ

Question:Why wouldn't the camera world with Cheese?

Answer:
The cheese tool could only work with the USB camera. The IMX219 series are CSI camera. Please use the provided test command.

Question:The camera could not work with opencv

Answer:

The CSI camera could work with the Gstreamer pipeline, please check the calling method.

Question:What is the operating temperature range of the IMX219 series?

Answer:

0-60°.

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=IMX219-160_Camera&oldid=88855 (https://www.waveshare.com/w/index.php?title=IMX219-160_Camera&oldid=88855)"