

i.MX 8 Camera Use Cases

Marco Franchi

Version 1.1, 2019-08-05

Table of Contents

1. Introduction	1
2. Obtaining a Linux L4.14.78_1.0.0-GA Image	2
2.1. i.MX BSP Releases	2
2.2. The Yocto Project Build	2
2.3. Changing DTB files on Linux	2
3. Camera daughter cards	3
3.1. MINISASTOCSI camera daughter card	3
3.2. MX8XMIPI4CAM2 adapter daughter card	3
3.3. MCIMXCAMERA1MP camera daughter card	4
4. i.MX 8MM EVK camera use cases	5
4.1. i.MX 8MM EVK Device Tree available	5
4.2. i.MX 8MM EVK camera use cases	6
5. i.MX 8MQ EVK camera use cases	7
5.1. i.MX 8MQ EVK Device Tree available	7
5.2. i.MX 8MQ EVK camera examples	8
6. i.MX 8QXP MEK camera use cases	10
6.1. i.MX 8QXP MEK Device Tree available	10
6.2. i.MX 8QXP MEK camera examples	11
7. i.MX 8QM MEK camera use cases	13
7.1. i.MX 8QM MEK Device Tree available	13
7.2. i.MX 8QM MEK camera examples	14
8. Advanced camera use cases topics	17
8.1. Multiple cameras output	17
8.2. Camera Zero-copy example	18
8.3. V4L2 Encode and Decode Extra Properties	20

1. Introduction

This document describes all the i.MX 8 MIPI-CSI use cases, showing the available cameras and daughter cards supported by the boards, the compatible Device Tree (DTS) files, and how to enable these different camera options on the i.MX 8 boards.

This guide is based on the L4.14.78_1.0.0-GA release.

The document covers the following topics:

- Obtaining a Linux L4.14.78_1.0.0-GA image release
- The i.MX 8 MIPI-CSI cameras and daughter cards
- i.MX 8MM EVK camera use cases
- i.MX 8MQ EVK camera use cases
- i.MX 8QXP MEK camera use cases
- i.MX 8QM MEK camera use cases
- Advanced GStreamer camera use cases

2. Obtaining a Linux L4.14.78_1.0.0-GA Image

There are two methods available to install the L4.14.78_1.0.0-GA release into the i.MX 8 boards:

2.1. i.MX BSP Releases

Prebuilt Images can be downloaded from the [NXP i.MX developer resources webpage](#). These images are composed by a Linux Kernel, ATF, SC firmware and U-Boot, and can be directly written to a SDCard. This SDCard can be then placed in the SDCard slot on the board and booted.

2.2. The Yocto Project Build

To build a Yocto Image from the source code, please refer to the "i.MX Yocto Project User's Guide" for detailed information.

2.3. Changing DTB files on Linux

To change the DTB files on Linux, follow the steps below. Please ensure you have an image loaded into an SDCard and that your board successfully boots U-Boot and Kernel:

Step 1: Reboot the board and press any key to stop the boot process at U-Boot.

Step 2: Type the desired DTB file following the example command line below:

```
=> setenv fdt_file 'fsl-imx8mm-evk.dtb'
```

Step 3: Save the environment variable:

```
=> saveenv
```

Step 4: Boot the Linux Kernel:

```
=> boot
```

3. Camera daughter cards

The i.MX 8 family has a set of cameras and daughter cards that can be purchased separately to improve the user experience. Refer to the next section for more information on how to use and what expected from these devices.

3.1. MINISASTOCSI camera daughter card

This accessory is a MIPI-CSI interface camera kit, based on the OmniVision OV5640, which is compatible with all the i.MX 8 boards released by the time of publishing of this document.

Supported resolutions:

- QXGA: 2592x1944@15fps;
- Full HD: 1920x1080@30fps;
- HD: 1280x720@60fps.



Figure 1. MINISASTOCSI camera daughter card

3.2. MX8XMIPI4CAM2 adapter daughter card

This accessory is a four camera adapter daughter card with direct board connectors for the OV10635 MIPI camera through a MAX9286 coaxial cable. This daughter card is supported by the i.MX 8QM MEK and the i.MX 8QXP MEK and only works with the OV10635 camera, which is sold separately.



Figure 2. MX8XMIPI4CAM2 adapter board

3.3. MCIMXCAMERA1MP camera daughter card

This accessory is a camera based on Ominivision OV10635 and MAX9286 coaxial cable, which can be used until four at same time in combination with the MX8XMIPI4CAM2 daughter card. This accessory is only compatible with the i.MX 8QM MEK and the i.MX 8QXP MEK.

Supported resolutions:

- WXGA: 1280x800@30fps;
- HD: 1280x720@30fps;
- WVGA: 752x480@30fps;
- VGA: 640x480@30fps;
- CIF: 352x288@30fps;
- QVGA: 320x240@30fps;

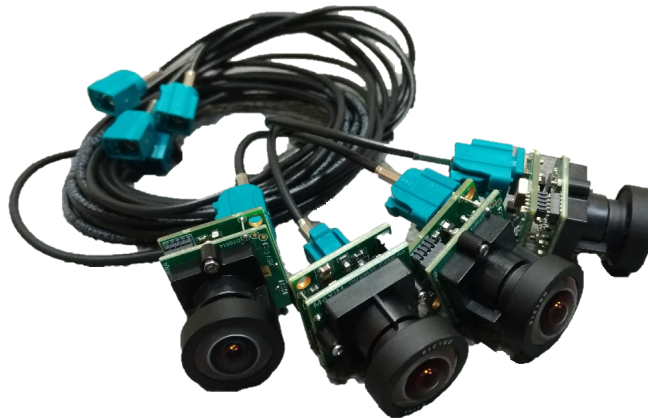


Figure 3. MCIMXCAMERA1MP camera daughter card

4. i.MX 8MM EVK camera use cases

This section describes the i.MX 8MM EVK MIPI-CSI0 camera use case. It covers the available Device Tree, daughter card supported, and an example on how to enable and use it in L4.14.78_1.0.0-GA.

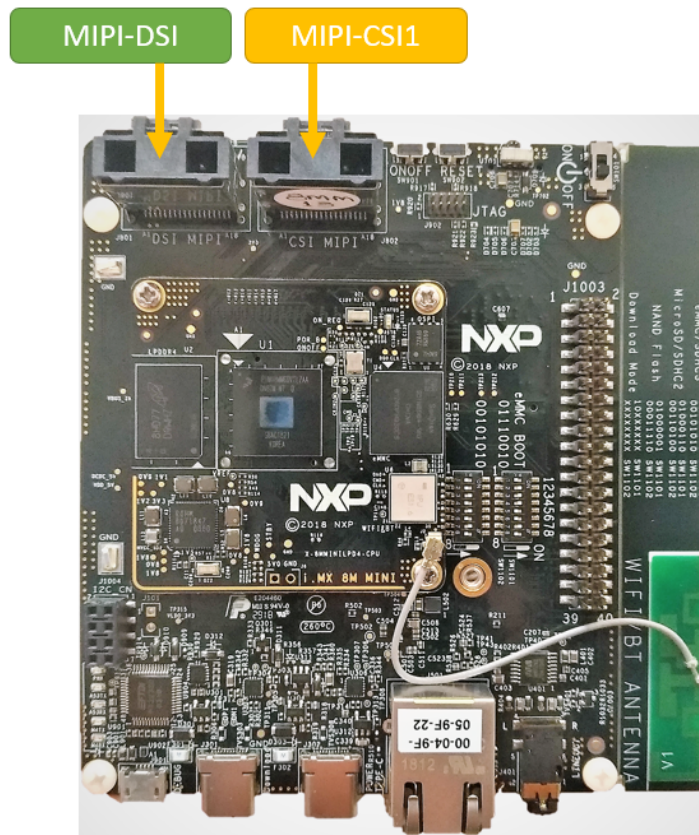


Figure 4. i.MX 8MM EVK mini-SAS connectors

4.1. i.MX 8MM EVK Device Tree available

The L4.14.78_1.0.0-GA release for i.MX 8MM EVK has just one Device Tree file available that affect the behavior of the camera connected to the MIPI-CSI connector. The table below describes this Device Tree option. Please, consult the [Obtaining a Linux L4.14.78_1.0.0-GA Image](#) chapter above to obtain instructions on how to change the Device Tree on U-boot.

Table 1. i.MX 8MM EVK B0 - DTB files available

DTB Name	mini-SAS Connector	Supported camera	Daughter card	Max Resolution
fsl-imx8mm-evk.dtb	MIPI-CSI	OV5640	MINISASTOCSI	2592x1944@15fps

4.2. i.MX 8MM EVK camera use cases

The i.MX 8MM EVK has its MIPI-CSI enabled by default. The user can connect an OV5640 camera through the MINISASTOCSI daughter card and reach up to 2592x1944@15fps resolution.

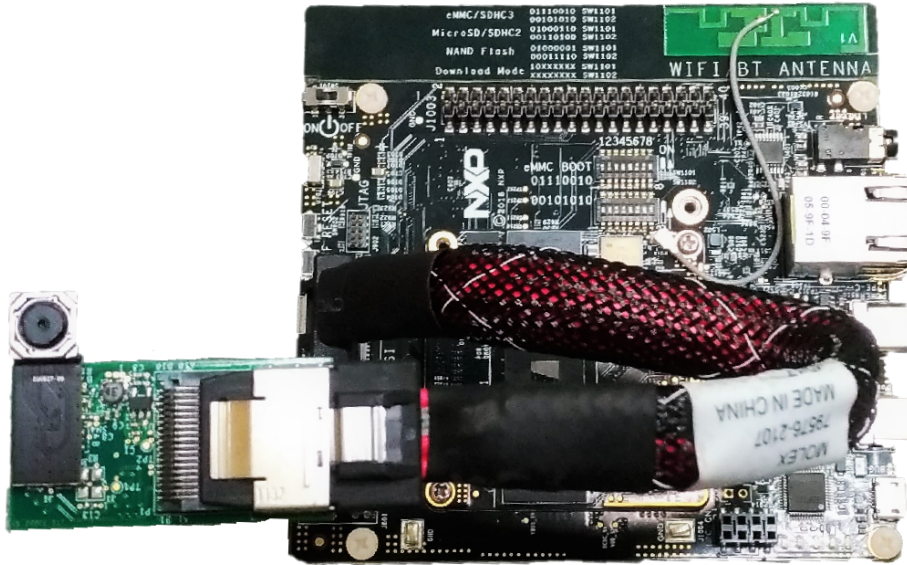


Figure 5. i.MX 8MM EVK MIPI OV5640 example

This camera will be allocated on `/dev/video0`. In order to test it, use the `fsl-imx8mm-evk.dtb` in combination with the pipelines below:

```
$ gst-launch1.0 v4l2src device=/dev/video0 ! autovideosink
```


5. i.MX 8MQ EVK camera use cases

This section describes the i.MX 8MQ EVK MIPI-CSI1 and MIPI-CSI2 camera use cases. It covers the available Device Tree options, daughter cards supported, and each use case example, including how to reproduce it in L4.14.78_1.0.0-GA.

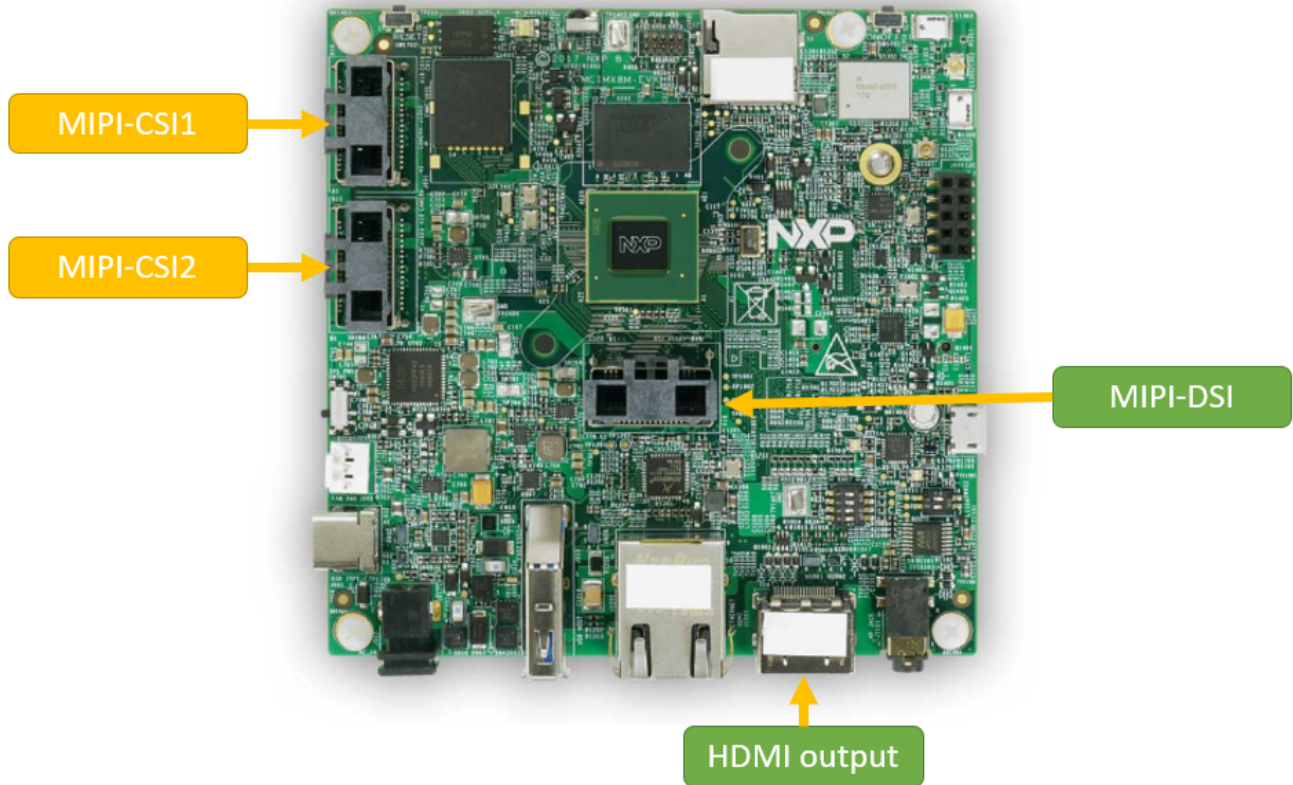


Figure 6. i.MX 8MQ EVK mini-SAS connectors

5.1. i.MX 8MQ EVK Device Tree available

The L4.14.78_1.0.0-GA release for i.MX 8MQ EVK has some Device Tree files available that affect the behavior of the cameras connected to the MIPI-CSI connectors. The table below describes all the Device Tree options to use/enable these i.MX 8MQ EVK camera use cases. Please, consult the [Obtaining a Linux L4.14.78_1.0.0-GA Image](#) chapter above to obtain instructions on how to change the Device Tree on U-boot.

Table 2. i.MX 8MQ EVK B0 - DTB files available

DTB Name	mini-SAS Connector	Supported camera	Daughter card	Max Resolution
fsl-imx8mq-evk.dtb/ fsl-imx8mq-evk-b3.dtb	MIPI-CSI0/CSI1	2xOV5640	2xMINISASTOCSI	2592x1944@15fps
fsl-imx8mq-evk-mipi-csi2.dtb	MIPI-CSI1	OV5640	MINISASTOCSI	2592x1944@15fps

5.2. i.MX 8MQ EVK camera examples

This section shows in details what can be expected when combined the DTS files and the cameras daughter cards available for i.MX 8MQ EVK.

5.2.1. Dual MIPI OV5640

The i.MX 8MQ EVK has both MIPI-CSI interfaces enabled by default. The user can connect two MINISASTOCSI daughter cards and reach up to 2592x1944@15fps resolution.

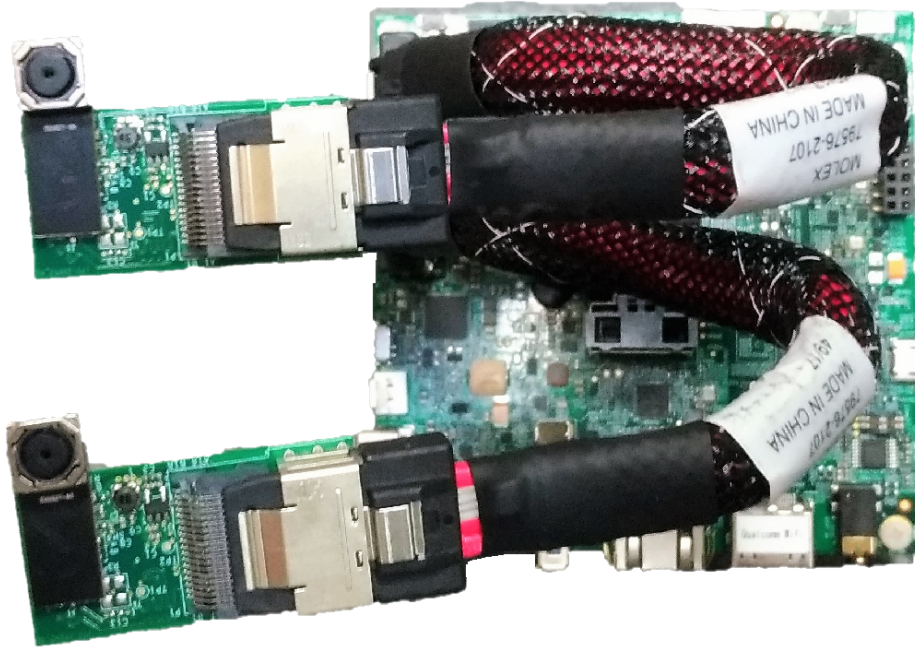


Figure 7. i.MX 8MQ EVK MIPI OV5640 example

These cameras will be allocated on `/dev/video0` and `/dev/video1`. To corroborate functionality use the `fsl-imx8mq-evk.dtb` and test by using the pipelines below:

```
$ gst-launch1.0 v4l2src device=/dev/video0 ! autovideosink
$ gst-launch1.0 v4l2src device=/dev/video1 ! autovideosink
```

NOTE

If you are using the i.MX 8 MQ EVK B3 or an old version, please, change the DTB file to `fsl-imx8mq-evk-b3.dtb` instead of `fsl-imx8mq-evk.dtb`.

5.2.2. MIPI OV5640 - Only on MIPI-CSI2

Another way to use a camera on the i.MX 8MQ EVK is enabling only the MIPI-CSI2 interface. For this, change the DTB file to `fsl-imx8mq-evk-mipi-csi2.dtb` and be sure to connect the camera to the MIPI-CSI2 interface.

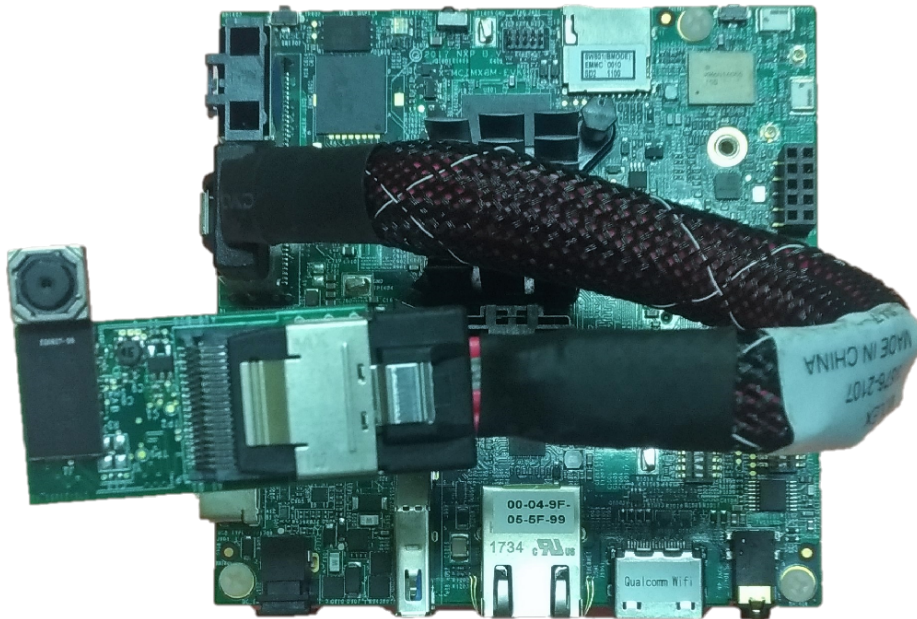


Figure 8. i.MX 8MQ EVK MIPI OV5640 example

By default, this camera will be allocated on `/dev/video0`. A way to test this is:

```
$ gst-launch1.0 v4l2src ! autovideosink
```

6. i.MX 8QXP MEK camera use cases

This section describes the i.MX 8QXP MEK MIPI-CSI0 camera use cases. It covers the available Device Tree options, daughter cards supported, and each use case example, including how to reproduce it in L4.14.78_1.0.0-GA.

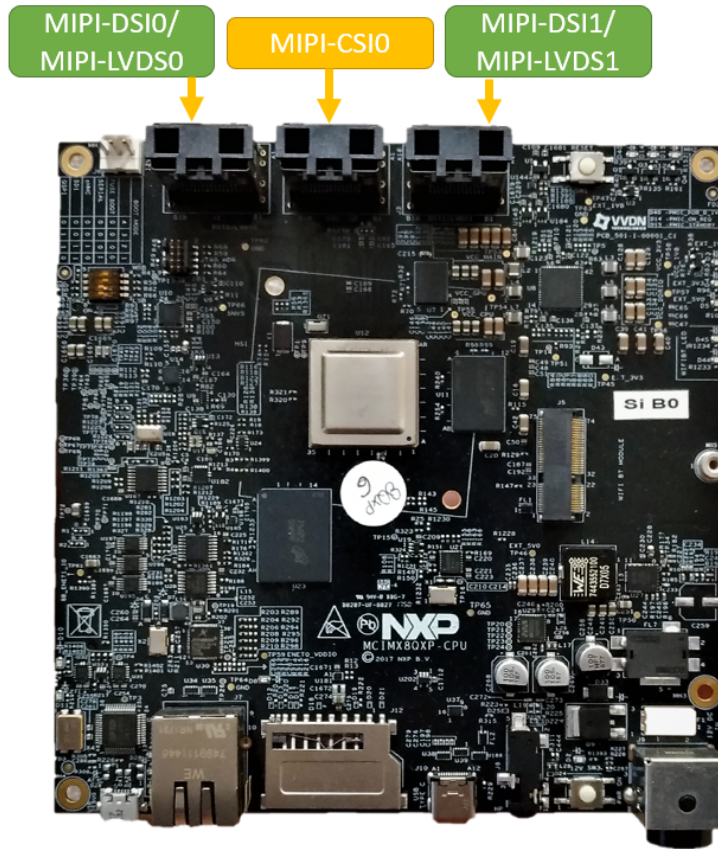


Figure 9. i.MX 8QXP MEK mini-SAS connectors

6.1. i.MX 8QXP MEK Device Tree available

The L4.14.78_1.0.0-GA release for i.MX 8QXP MEK has some Device Tree files available that affect the behavior of the cameras connected to the MIPI-CSI0 connector. The table below describes all the Device Tree options to use/enable these i.MX 8QXP MEK camera use cases. Please, consult the [Obtaining a Linux L4.14.78_1.0.0-GA Image](#) chapter above to obtain instructions on how to change the Device Tree on U-boot.

Table 3. i.MX 8QXP MEK B0 - DTB files available

DTB Name	mini-SAS Connector	Supported camera	Daughter card	Max Resolution
fsl-imx8qxp-mek.dtb	MIPI-CSI0	MCIMXCAMER A1MP	MX8XMIPI4CAM2	1280x800@30fps
fsl-imx8qxp-mek-mipi-ov5640.dtb	MIPI-CSI0	OV5640	MINISASTOCSI	2592x1944@15fps

6.2. i.MX 8QXP MEK camera examples

This section shows in details what can be expected when combined the DTS files and the cameras daughter cards available for the i.MX 8QXP MEK.

6.2.1. OV10635

Using the MX8XMIPI4CAM2 in combination with one MCIMXCAMERA1MP daughter card, the user can connect it to the MIPI-CSIO mini-SAS interface and enable it by using the default DTB `fsl-imx8qxp-mek.dtb`.

NOTE

The i.MX 8QXP MEK B0 has a known hardware issue (e50058), which limits the MIPI-CSI interface for single camera operation support only.

6.2.2. MIPI OV5640

Another way to use a camera on the i.MX 8QXP MEK is through the MINISASTOCSI daughter card, which includes an OV5640 camera. This camera can reach up to 2592x1944@15fps and in order to use it, you just need to connect the mini-SAS to the MIPI-CSI0 interface and change the DTB to `fsl-imx8qxp-mek-mipi-ov5640.dtb`.

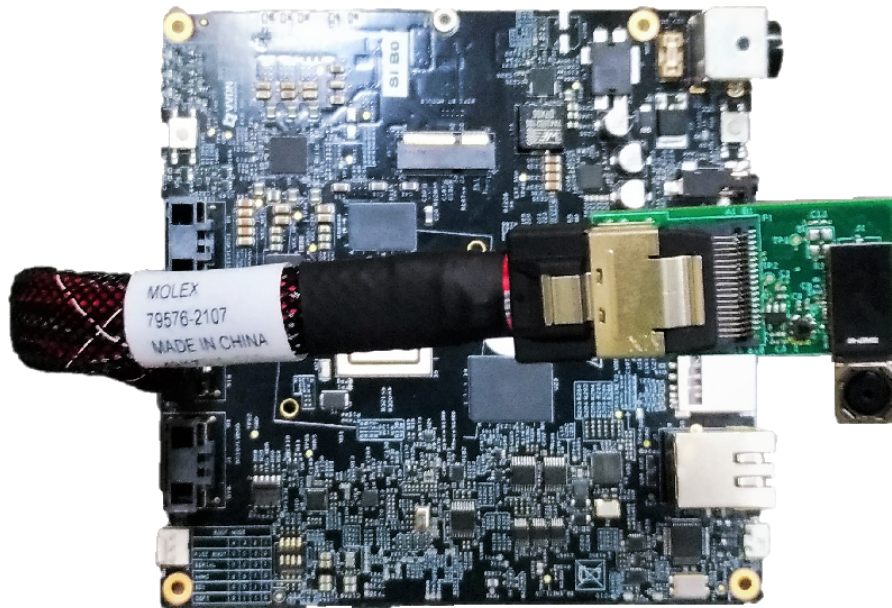


Figure 10. i.MX 8QXP MEK MIPI OV5640 example

By default, this camera will be allocated on `/dev/video0`. A way to test this is:

```
$ gst-launch1.0 v4l2src ! autovideosink
```

7. i.MX 8QM MEK camera use cases

This section describes the i.MX 8QM MEK MIPI-CSI0/CSI1 camera use cases. It covers the available Device Tree options, daughter cards supported, and each use case example, including how to reproduce it in L4.14.78_1.0.0-GA.

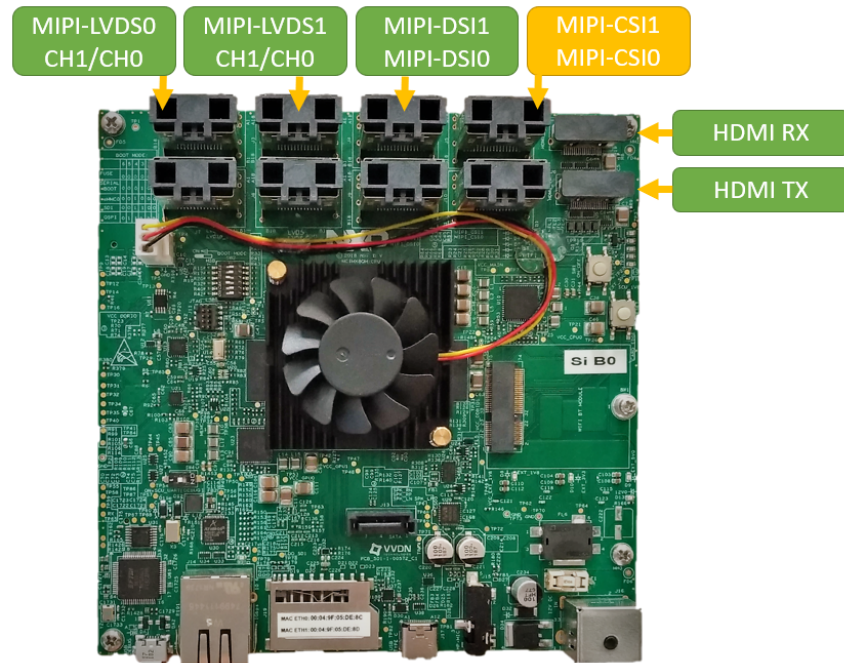


Figure 11. i.MX 8QM MEK mini-SAS connectors

7.1. i.MX 8QM MEK Device Tree available

The L4.14.78_1.0.0-GA release for i.MX 8QM MEK has some Device Tree files available that affect the behavior of the cameras connected to the MIPI-CSI connectors. The table below describes all the Device Tree options to use/enable these i.MX 8QM MEK camera use cases. Please, consult the [Obtaining a Linux L4.14.78_1.0.0-GA Image](#) chapter above to obtain instructions on how to change the Device Tree on U-boot.

Table 4. i.MX 8QM MEK B0 - DTB files available

DTB Name	mini-SAS Connector	Supported camera	Daughter card	Max Resolution
fsl-imx8qm-mek.dtb	MIPI-CSI0	MCIMXCAMER A1MP	MX8XMIPI4CAM2	1280x800@30fps
fsl-imx8qm-mek-mipi-ov5640.dtb	MIPI-CSI0	OV5640	MINISASTOCSI	2592x1944@15fps
fsl-imx8qm-mek-mipi-two-ov5640.dtb	MIPI-CSI0/CSI1	2xOV5640	2xMINISASTOCSI	2592x1944@15fps
fsl-imx8qm-mek-8cam.dtb	MIPI-CSI0/CSI1	2xMCIMXCAM ERA1MP	2xMX8XMIPI4CAM2	1280x800@30fps

7.2. i.MX 8QM MEK camera examples

This section shows in details what can be expected when combined the DTS files and the cameras daughter cards available for the i.MX 8QM MEK.

7.2.1. Four OV10635

Using the MX8XMIPI4CAM2 in combination with four MCIMXCAMERA1MP daughter cards, the user can connect it to the MIPI-CSI0 mini-SAS interface and enable this four cameras at the same time by using the default DTB `fs1-imx8qm-mek.dtb`.



Figure 12. i.MX 8QM MEK 4 cameras example

NOTE

Be sure to use the power from the board to turn on the MX8XMIPI4CAM2. For this, use a cable to connect the MX8XMIPI4CAM2 J15 jumper.

To keep the section short and avoid unnecessary duplication of information, please check the Advanced Topics chapter at the end of this document to see how to run the four cameras at the same time.

7.2.2. MIPI OV5640

Another way to use a camera on the i.MX 8QM MEK is through the MINISASTOCSI daughter card, which includes an OV5640 camera. This camera can reach up to 2592x1944@15fps and in order to use it, you just need to connect the mini-SAS to the MIPI-CSI0 interface and change the DTB to `fsl-imx8qm-mek-mipi-ov5640.dtb`.

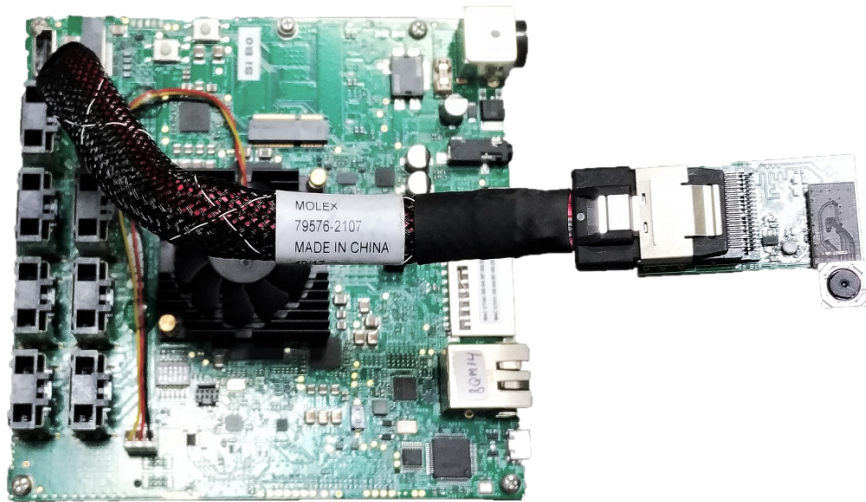


Figure 13. i.MX 8QM MEK MIPI OV5640 example

By default, this camera will be allocated on `/dev/video0`. A way to test this is:

```
$ gst-launch1.0 v4l2src ! autovideosink
```

7.2.3. Dual MIPI OV5640

The i.MX 8QM MEK has two MIPI-CSI interfaces: MIPI-CSI0 and MIPI-CSI1. You can enable both by changing the DTB to `'fsl-imx8qm-mek-mipi-two-ov5640.dtb'` and by connecting two MINISASTOCSI to it.

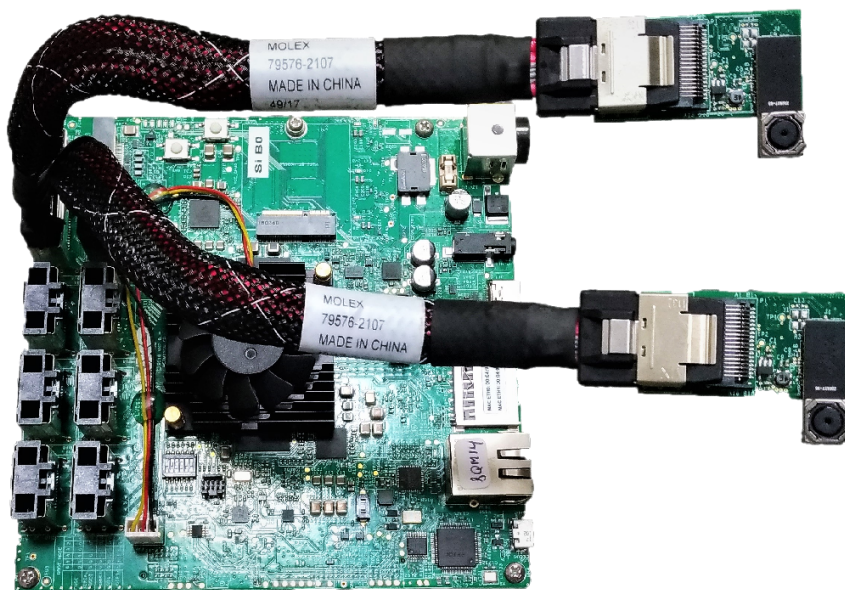


Figure 14. i.MX 8QM MEK Dual MIPI OV5640 example

The cameras will reach up to 2592x1944@15fps and you can test it by using the pipelines below:

```
$ gst-launch1.0 v4l2src device=/dev/video0 ! autovideosink  
$ gst-launch1.0 v4l2src device=/dev/video1 ! autovideosink
```

Please, check the [Advanced camera use cases topics](#) chapter at the end of this documentation for more dual camera use cases and details.

7.2.4. Eight OV10635

The last i.MX 8QM MEK covered topic includes the use of the dual MIPI-CSI interfaces and two MX8XMIPI4CAM2 in combination with eight MCIMXCAMERA1MP daughter cards connected to them. For this, change the DTB file to `fsl-imx8qm-mek-8cam.dtb` and boot the board.

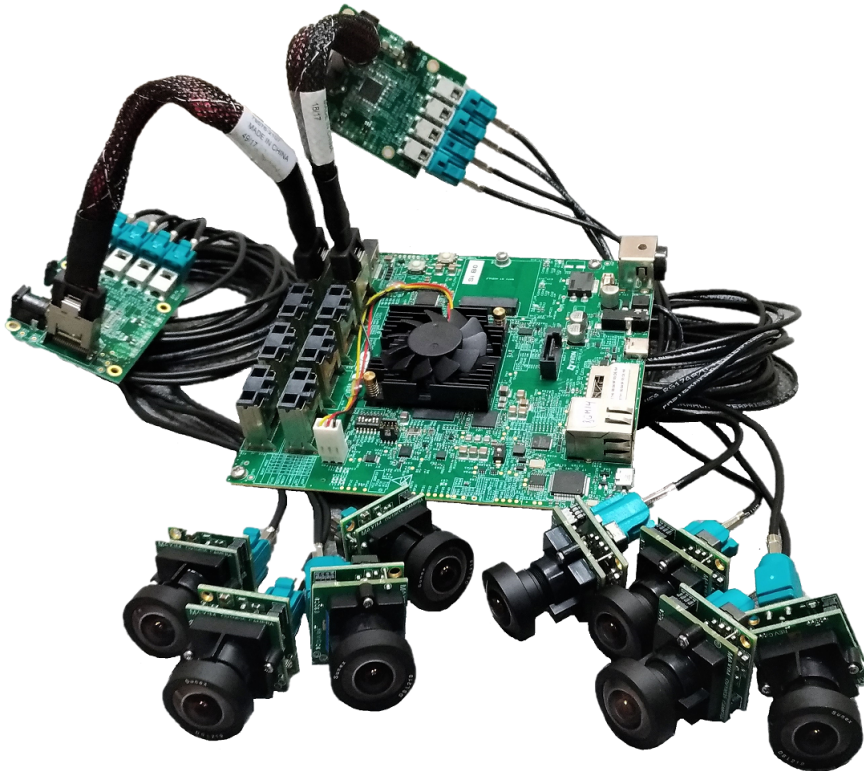


Figure 15. i.MX 8QM MEK 8 cameras example

Please, check the [Advanced camera use cases topics](#) chapter at the end of this document to check details about how to use the eight cameras at the same time.

8. Advanced camera use cases topics

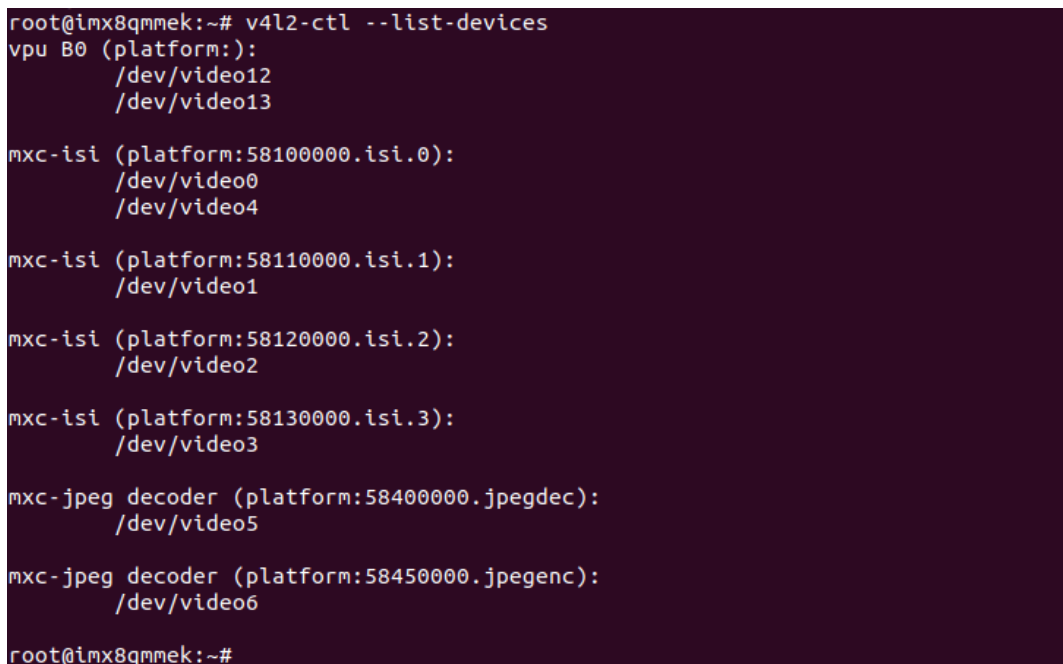
This section describes some advanced topics and tips to get high experiences in camera use cases in the i.MX 8 boards.

8.1. Multiple cameras output

In order to use more than one camera at the same time displayed in the same monitor output, boot the board and enter with the following command line to get the cameras ids:

```
$ v4l2-ctl --list-devices
```

It will result in something similar to the image below:

A terminal window with a dark background and light text. The prompt is 'root@imx8qmnek:~#'. The command 'v4l2-ctl --list-devices' has been executed, resulting in the following output:

```
root@imx8qmnek:~# v4l2-ctl --list-devices
vpu B0 (platform:):
  /dev/video12
  /dev/video13

mxc-isi (platform:58100000.isi.0):
  /dev/video0
  /dev/video4

mxc-isi (platform:58110000.isi.1):
  /dev/video1

mxc-isi (platform:58120000.isi.2):
  /dev/video2

mxc-isi (platform:58130000.isi.3):
  /dev/video3

mxc-jpeg decoder (platform:58400000.jpegdec):
  /dev/video5

mxc-jpeg decoder (platform:58450000.jpegenc):
  /dev/video6

root@imx8qmnek:~#
```

Figure 16. v4l2-ctl results

According to the image, **mxc-isi** found and set **four cameras** to **/dev/video0**, **video1**, **video2**, and **video3**. So we can build the GStreamer pipeline below changing it as required:

```
$ gst-launch-1.0 -v imxcompositor_g2d name=comp \
sink_0::xpos=0 sink_0::ypos=0 sink_0::width=640 sink_0::height=480 \
sink_1::xpos=0 sink_1::ypos=480 sink_1::width=640 sink_1::height=480 \
sink_2::xpos=640 sink_2::ypos=0 sink_2::width=640 sink_2::height=480 \
sink_3::xpos=640 sink_3::ypos=480 sink_3::width=640 sink_3::height=480 \
! video/x-raw,format=RGB16 ! waylandsink \
v4l2src device=/dev/video0 ! video/x-raw,width=640,height=480 ! comp.sink_0 \
v4l2src device=/dev/video1 ! video/x-raw,width=640,height=480 ! comp.sink_1 \
v4l2src device=/dev/video2 ! video/x-raw,width=640,height=480 ! comp.sink_2 \
v4l2src device=/dev/video3 ! video/x-raw,width=640,height=480 ! comp.sink_3
```

This pipeline enables the user to set up more than one camera to the same screen using the `imxcompositor_g2d` for it. This is the unique solution available to create an interface over Weston/Wayland interface, i.e., in i.MX 8 devices we need to use GPU to create interfaces GUI.

This pipeline results in the image below:



Figure 17. GStreamer 4 cameras output

It is just an example and you are encouraged to change it for other camera use cases, such as the **eight cameras** supported by the i.MX 8QM MEK.

8.2. Camera Zero-copy example

The i.MX 8QM MEK and the i.MX 8QXP MEK uses the Amphion VPU, which adopted the `V4L2 API` for the VPU solutions.

This API is already heavily used outside NXP products and has a great and vast community working day after day to provide new features for it. One of these features is the use of `DMABUF`.

The `DMABUF` has been used on i.MX processors since the i.MX 6 and in combination with the `V4L2 API` can provide the `Zero-copy` GStreamer pipeline feature. The `Zero-copy` uses this `DMABUF` to create a direct access to the video frame data, avoiding any kind of copy to cache or memory. This adoption decrease significantly the CPU usage, once there is no memory copy/paste manipulation.

Just as an example, compare the GStreamer pipelines below and its respective `top` commands:

8.2.1. GStreamer camera encode example without Zero-copy support

```
gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=300 ! \  
'video/x-raw,format=(string)NV12,width=1920,height=1080,framerate=(fraction)30/1' ! \  
queue ! v4l2h264enc ! \  
avimux ! filesink location=test.avi
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
3893	root	20	0	431588	35076	30352	S	102.2	1.3	0:06.94	gst-launch+
5	root	20	0	0	0	0	I	4.3	0.0	0:00.99	kworker/u8+
3894	root	20	0	3636	2256	1792	R	2.2	0.1	0:00.10	top
1	root	20	0	155844	6224	4636	S	0.0	0.2	0:04.37	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:+
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_+
7	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:00.02	rcu_preempt
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_sched
10	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
14	root	rt	0	0	0	0	S	0.0	0.0	0:00.01	migration/1
15	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/1
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1

Figure 18. Encode CPU usage without Zero-copy example

8.2.2. GStreamer camera encode example with Zero-copy support

```
gst-launch-1.0 v4l2src device=/dev/video0 num-buffers=300 io-mode=dmabuf ! \  
'video/x-raw,format=(string)NV12,width=1920,height=1080,framerate=(fraction)30/1' ! \  
queue ! v4l2h264enc output-io-mode=dmabuf-import ! \  
avimux ! filesink location=test.avi
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
5	root	20	0	0	0	0	I	6.3	0.0	0:01.40	kworker/u8+
3903	root	20	0	407236	35080	30352	S	5.6	1.3	0:00.63	gst-launch+
3904	root	20	0	3636	2192	1728	R	1.0	0.1	0:00.09	top
40	root	20	0	0	0	0	I	0.3	0.0	0:01.38	kworker/u8+
94	root	20	0	0	0	0	I	0.3	0.0	0:00.56	kworker/0:1
1	root	20	0	155844	6224	4636	S	0.0	0.2	0:04.37	systemd
2	root	20	0	0	0	0	S	0.0	0.0	0:00.01	kthreadd
3	root	20	0	0	0	0	I	0.0	0.0	0:00.00	kworker/0:0
4	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	kworker/0:+
6	root	0	-20	0	0	0	I	0.0	0.0	0:00.00	mm_percpu_+
7	root	20	0	0	0	0	S	0.0	0.0	0:00.01	ksoftirqd/0
8	root	20	0	0	0	0	I	0.0	0.0	0:00.02	rcu_preempt
9	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_sched
10	root	20	0	0	0	0	I	0.0	0.0	0:00.00	rcu_bh
11	root	rt	0	0	0	0	S	0.0	0.0	0:00.00	migration/0
12	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/0
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1
13	root	20	0	0	0	0	S	0.0	0.0	0:00.00	cpuhp/1

Figure 19. Encode CPU usage for Zero-copy support

Note that the Zero-copy plus the V4L2 API have a range of opportunities and a lot of new possibilities, which if were listed generates a new entire document, so for now this topic will be restricted just for this example.

8.3. V4L2 Encode and Decode Extra Properties

The i.MX 8QM MEK and the i.MX 8QXP MEK uses the Amphion VPU, which adopted the **V4L2 API** for the VPU solutions. This API provides the following encode and decode plugins:

- v4l2jpegdec: V4L2 JPEG Decoder
- v4l2mpeg4dec: V4L2 MPEG4 Decoder
- v4l2mpeg2dec: V4L2 MPEG2 Decoder
- v4l2avsdec: V4L2 AVS Decoder
- v4l2divxdec: V4L2 DIVX Decoder
- v4l2spkdec: V4L2 SPK Decoder
- v4l2h263dec: V4L2 H263 Decoder
- v4l2h264dec: V4L2 H264 Decoder
- v4l2h265dec: V4L2 H265 Decoder
- v4l2rvdec: V4L2 RV Decoder
- v4l2vp6dec: V4L2 VP6 Decoder
- v4l2vp8dec: V4L2 VP8 Decoder
- v4l2vc1dec: V4L2 VC1 Decoder
- v4l2h264enc: V4L2 H.264 Encoder
- v4l2video4jpegdec: V4L2 JPEG Decoder
- v4l2jpegenc: V4L2 JPEG Encoder

Those plugins do not provide a direct support for some video control properties, such as **BITRATE**, **GOP SIZE** or **INFRA FRAME**. For add this control, the user has to use the **extra-controls** properties. You can check an example below:

```
gst-launch-1.0 v4l2src ! video/x-raw,width=640,height=480 ! \  
v4l2h264enc extra-controls="controls,h264_entropy_mode=0,video_bitrate=245000;" ! \  
h264parse ! v4l2h264dec ! queue ! waylandsink sync=false
```