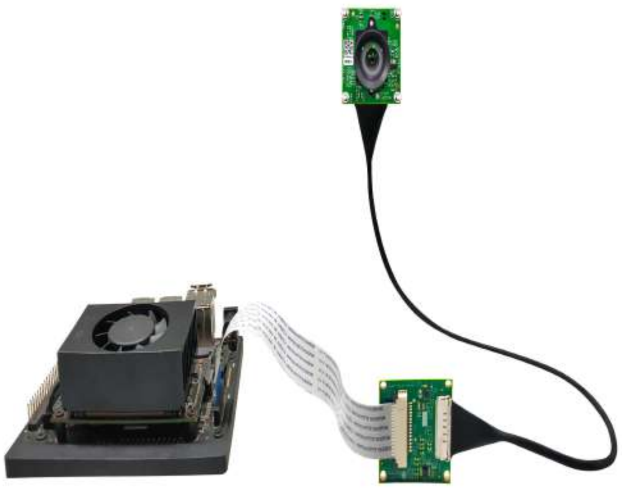
e-CAM23_CUNX

Gstreamer Usage Guide



Version 1.0

e-con Systems

15/02/2021

e-con Systems

Your Product Development Partner

Contents

Introduction to Gstreamer

Gstreamer is a powerful and flexible multimedia framework with a lot of capabilities. It supports various features such as capturing, displaying, encoding, and decoding of image data. While using Gstreamer, the possibilities of manipulating data are limitless.

The commands and output messages in this manual are represented by different colors as shown in below table.

Table 1: Notation of Colors

Color	Notation
Blue	Commands running in development board
Red	Output message in development PC
Orange	Commands running in client PC

This document explains how to install Gstreamer-1.0 on the NVIDIA®Jetson Nano/ Xavier NX™ development kit and use it with the available plugins.

Installation Procedure

This section describes how to install Gstreamer-1.0 binaries and libraries.

To know the details of available plugins, please refer to the *L4T32.1_Accelerated_Gstreamer _User_Guide.pdf* from Jetson Download Center.

The steps to install Gstreamer-1.0 binaries and libraries are as follows:

1. Run the following commands to install Gstreamer-1.0 binaries and libraries on the Jetson Nano/ Xavier NX™ platform.

```
$ sudo apt-add-repository universe
$ sudo apt-get update
$ sudo apt-get install gstreamer1.0-tools
gstreamer1.0-alsa gstreamer1.0-plugins-base
gstreamer1.0-plugins-good gstreamer1.0-plugins-bad
gstreamer1.0-plugins-ugly gstreamer1.0-libav
```

Note: If the installation process stops with **could not get lock /var/lib/dpkg/lock** message, run the following command to remove the file and proceed with the installation.

```
$ sudo rm /var/lib/dpkg/lock
```

2. Run the following command to check the Gstreamer-1.0 version.

```
$ gst-inspect-1.0 --version
```

Note: Make sure that e-CAM23_CUNX is connected and the required drivers are loaded.

During booting, the module drivers for e-CAM23_CUNX will be loaded automatically in the Jetson™ board.

3. Run the following command to confirm whether the camera connected is initialized.

```
$ dmesg | grep "Detected OV2311 sensor"
```

The output message appears as shown below.

```
Detected OV2311 sensor
```

The output message indicates that camera connected is initialized properly.

4. Run the following command to check the presence of video node.

```
$ ls /dev/video*
```

The output message for camera setup appears as shown below.

```
/dev/video0
```

Tested Gstreamer Examples

The section describes some of the tested Gstreamer commands which work on the Jetson Nano/ Xavier NX™ development kit.

Note: Please run the following commands to change the power mode to maximum for better performance and to get maximum frame rate.

```
$ sudo nvpmodel -m 0

$ sudo jetson_clocks
```

Example 1: Streaming 2MP from a single camera (HW accelerated)

Run the following command to stream 2MP video from a single camera.

```
$ gst-launch-1.0 v4l2src device=/dev/video<n> ! "video/x-
raw,width=1600,height=1300,format=GRAY8" ! nvvidconv !
"video/x-
raw(memory:NVMM),width=1600,height=1300,format=I420" !
fpsdisplaysink video-sink=nvoverlaysink text-
overlay=false sync=false -v


$ gst-launch-1.0 -v v4l2src device=/dev/video<n> !
"video/x-raw,width=1600,height=1300,format=GRAY16_BE" !
rawvideoparse use-sink-caps=false width=1600 height=1300
format="gray16-le" ! videoconvert ! "video/x-
raw,width=1600,height=1300,format=I420" ! fpsdisplaysink
video-sink=autovideosink text-overlay=false sync=false -v
```

Note: Replace **<n>** with the number of video device node, from which you need to stream.

Example 2: Record 2MP in H.264 format to a video file (HW accelerated)

Run the following command to record 2MP video in H.264 format to a video file.

```
$ gst-launch-1.0 v4l2src device=/dev/video<n> ! "video/x-
raw, format=(string)GRAY8, width=(int)1600,
height=(int)1300" ! nvvidconv ! "video/x-raw(memory:NVMM),
format=(string)I420" ! omxh264enc qp-range=20,20:20,20:-
1,-1 ! matroskamux ! queue ! filesink location=<file.mkv>


$ gst-launch-1.0 v4l2src device=/dev/video<n> ! "video/x-
raw, format=(string)GRAY16_BE, width=(int)1600,
height=(int)1300" ! rawvideoparse use-sink-caps=false
width=1600 height=1300 format="gray16-le" ! videoconvert !
"video/x-raw,width=1600,height=1300,format=I420" !
omxh264enc qp-range=20,20:20,20:-1,-1 ! matroskamux !
queue ! filesink location=<file.mkv>
```

Note: Replace **<n>** with the number of video device node, from which you need to stream. Change **<filename>** with a name (e.g.: Video0), to which the video will get stored.

Example 3: Playback of saved video file (HW accelerated)

Run the following command to playback the saved video.

```
$ gst-launch-1.0 filesrc location=<filename>.mkv !
matroskademux ! h264parse !  omxh264dec ! nvoverlaysink
```

Note: Change **<filename>** with the name under which the video is recorded

Example 4: Network streaming H.264 encoded FHD video using RTP over UDP (HW accelerated)

Run the following command to stream the video captured by camera connected with Jetson Nano/ Xavier NX™ development board.

```
$ gst-launch-1.0 v4l2src device=/dev/video<n> ! "video/x-
raw, format=(string)GRAY8,
width=(int)1600,height=(int)1300" ! nvvidconv ! "video/x-
raw(memory:NVMM), format=(string)I420" ! omxh264enc qp-
range=35,35:35,35:-1,-1 ! rtph264pay mtu=60000 ! udpsink
clients=<ip_address>:<port_no> sync=false


$ gst-launch-1.0 v4l2src device=/dev/video<n> ! "video/x-
raw, format=(string)GRAY16_BE,
width=(int)1600,height=(int)1300" ! rawvideoparse use-
sink-caps=false width=1600 height=1300 format="gray16-
le" ! videoconvert ! "video/x-
raw,width=1600,height=1300,format=I420"  ! omxh264enc qp-
range=35,35:35,35:-1,-1 ! rtph264pay mtu=60000 ! udpsink
clients=<ip_address>:<port_no> sync=false
```

Note: Do not close the application in Jetson Nano/ Xavier NX™ development board.

Run the following command in the Client device to view the video.

```
$ gst-launch-1.0 udpsrc port=<port_no>
caps="application/x-rtp, media=(string)video, clock-
rate=(int)90000,encoding-name=(string)H264, sprop-
parameter-
sets=(string)\"Z0JAKpWgHgCJ+VA\\=\\,aM48gA\\=\\=\",payloa
d=(int)96" ! rtph264depay ! decodebin ! autovideosink
sync=false
```

Note: Replace **<n>** with the number of video device node, from which you need to stream, **<ip_address>** with the IP address of the Client device (For example, 192.168.6.100) and **<port_no>** with the port number of the Client device (For example, 5000).

Example 6: Capturing 2MP still image

Run the following command to capture 2MP still image.

```
$ gst-launch-1.0 v4l2src num-buffers=1
device=/dev/video<n> ! 'video/x-raw, format=(string)GRAY8,
width=1600, height=1300' ! jpegenc ! filesink
location=<filename>.jpg


$ gst-launch-1.0 v4l2src num-buffers=1
device=/dev/video<n> ! 'video/x-raw,
format=(string)GRAY16_BE, width=1600, height=1300' !
rawvideoparse use-sink-caps=false width=1600 height=1300
format="gray16-le" ! videoconvert ! "video/x-
raw,width=1600,height=1300,format=I420" ! jpegenc !
filesink location=<filename>.jpg
```

Note: Replace **<n>** with the number of video device node, from which you need to stream. Change **<filename>** with a name (For example, Capture_1), to which the image will get stored.

Troubleshooting

In this section, you can view the commonly occurring issue and their troubleshooting step.

During network streaming, I face issues in video quality and frame rate. How to overcome this issue?

Network streaming quality and frame rate depends on the network bandwidth and receiver decode capability. In the receiver end, if you use software decoder, the frame rate and quality will get affected. Therefore, avoid using software decoder in the receiver end to resolve this issue.

FAQ

1. **I have run the Gstreamer command to stream camera as mentioned in this document, but I cannot get the maximum fps?**

   Gstreamer will use mmap method, due to this, the frame rate may get dropped. Also, there is limitation in the display plugin used in Gstreamer, which will affect the frame rate.

   You can use the ecam_tk1_guvcview camera application to achieve the maximum frame rate, by changing the exposure and setting the appropriate value.

2. **Why does the still capture sample gstreamer pipeline create improper white balance and exposure ?**

   The sample provided is just to demonstrate the still capture feature of e-CAM23_CUNX. To get the desired result, run the following command to set the properties of v4l2src plugin.

   ```
   $ gst-inspect-1.0 v4l2src
   ```

What’s Next?

After understanding the usage of Gstreamer application, you can refer to the following documents to understand more about e-CAM23_CUNX.

- *e-CAM23_CUNX Datasheet*
- *e-CAM23_CUMI2311_MOD Datasheet*
- *e-CAM23_CUNX_Lens Datasheet*
- *e-CAM23_CUNX_Release_Package_Manifest*

Glossary

2MP: 2 MegaPixel.

IP: Internet Protocol.

ISP: Image Signal Processor.

RTP: Real-time Transport Protocol.

UDP: User Datagram Protocol.

Support

Contact Us

If you need any support on e-CAM23_CUNX product, please contact us using the Live Chat option available on our website - https://www.e-consystems.com/

Creating a Ticket

If you need to create a ticket for any type of issue, please visit the ticketing page on our website - https://www.e-consystems.com/create-ticket.asp

RMA

To know about our Return Material Authorization (RMA) policy, please visit the RMA Policy page on our website - https://www.e-consystems.com/RMA-Policy.asp

General Product Warranty Terms

To know about our General Product Warranty Terms, please visit the General Warranty Terms page on our website - https://www.e-consystems.com/warranty.asp

Revision History

Rev	Date	Description	Author
1.0	15-Feb-2021	Initial Draft	Application Enginnering Team