JETSON TK1/TEGRA LINUX DRIVER PACKAGE MULTIMEDIA USER GUIDE

DA_07303-001_01 | July 8, 2014
Advance Information | Subject to Change

Release R19.3

DOCUMENT CHANGE HISTORY

DA_07303-001_01

Version	Date	Authors	Description of Change
v1.0	10 June 2014	mzensius	Initial release.
v1.1	25 June 2014	mzensius	Corrections to Video Format conversions.
v1.2	8 July 2014	mzensius	Converted to non-confidential document.

TABLE OF CONTENTS

JETSON TK1/TEGRA LINUX DRIVER PACKAGE MULTIMEDIA USER GUIDE

This document is a user guide for the Gstreamer v0.10-based accelerated solution included in NVIDIA® Tegra® Linux Driver Package for Ubuntu Linux 14.04 on the Jetson TK1 platform.

This document contains the following sections:

GSTREAMER INSTALLATION AND SETUP

This section describes how to install and configure Gstreamer v0.10.

To install Gstreamer

- Install Gstreamer on the Jetson TK1 platform with the following command:

```
$ sudo apt-get install gstreamer-tools gstreamer0.10-alsa
gstreamer0.10-plugins-base  gstreamer0.10-plugins-good
gstreamer0.10-plugins-bad gstreamer0.10-plugins-ugly gstreamer0.10-
ffmpeg
```

To check the Gstreamer version

- Check the Gstreamer version with the following command:

```
$ gst-inspect-0.10 –version
```

Note: Gstreamer version 0.10 plugins are included pre-installed in Linux for Tegra (L4T) R19.3 release package for Jetson TK1.

Gstreamer version 0.10 includes the following gst-openmax video decoders:

Video Decoder	Description
nv_omx_h264dec	OpenMAX IL H.264/AVC video decoder
nv_omx_mpeg4dec	OpenMAX IL MPEG-4 video decoder
nv_omx_vp8dec	OpenMAX IL VP8 video decoder
nv_omx_h263dec	OpenMAX IL H.263 video decoder

Gstreamer version 0.10 includes the following gst-openmax video encoders:

Video Encoders	Description
nv_omx_h264enc	OpenMAX IL H.264/AVC video encoder
nv_omx_vp8enc	OpenMAX IL VP8 video encoder

Gstreamer version 0.10 includes the following gst-openmax video sinks:

Video Sink	Description
nv_omx_videosink	OpenMAX IL videosink element
nv_omx_hdmi_videosink	OpenMAX IL HDMI videosink element

Gstreamer version 0.10 includes the following gst-openmax audio decoders:

Audio Decoder	Description
nv_omx_aacdec	OpenMAX IL AAC audio decoder
nv_omx_mp3dec	OpenMAX IL MP3 audio decoder

DECODE EXAMPLES USING GST-LAUNCH-0.10

The examples in this section show how you can perform audio and video decode with Gstreamer.

Audio Decode Examples

The following examples show how you can perform audio decode.

AAC Decode (NVIDIA-accelerated decode)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_00 ! queue ! nv_omx_aacdec ! alsasink -v -e
```

AAC Decode (OSS software decode)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_00 ! queue ! ffdec_aac ! alsasink -v -e
```

AMR-WB Decode (OSS software decode)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_00 ! queue ! ffdec_amrwb ! audioconvert ! alsasink -v -e
```

AMR-NB Decode (OSS software decode)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.audio_00 ! queue ! ffdec_amrnb ! audioconvert ! alsasink -v -e
```

MP3 Decode (NVIDIA-accelerated decode)

```
$ gst-launch-0.10 filesrc location=<filename.mp3> ! mpegaudioparse !
ffdec_mp3 ! audioconvert ! alsasink -v -e
```

Note: To route audio over HDMI, set the alsasink property `device` to `aux_plug`.

Video Decode Examples

The following examples show how you can perform video decode.

H.264 Decode (NVIDIA accelerated decode)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! nv_omx_hdmi_videosink –v -e
```

VP8 Decode (NVIDIA accelerated decode)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_vp8dec ! nv_omx_hdmi_videosink –v -e
```

MPEG-4 Decode (NVIDIA accelerated decode)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_mpeg4dec ! nv_omx_hdmi_videosink –v -e
```

Image Decode

```
$ gst-launch-0.10 filesrc location=<filename.jpg> ! nvjpegdec ! freeze
! xvimagesink –v -e
```

ENCODE EXAMPLES USING GST-LAUNCH-0.10

The examples in this section show how you can perform audio and video encode with Gstreamer.

Audio Encode Examples

The following examples show how you can perform audio encode.

AAC Encode (OSS software encode)

```
$ gst-launch-0.10 audiotestsrc ! 'audio/x-raw-int, rate=(int)44100,
channels=(int)2' ! ffenc_aac ! qtmux ! filesink location=test.mp4 -v -e
```

AMR-WB Encode (OSS software encode)

```
$ gst-launch-0.10 audiotestsrc ! 'audio/x-raw-int, rate=(int)16000,
channels=(int)1' ! voamrwbenc ! qtmux ! filesink location=test.mp4 -v -
e
```

Video Encode Examples

The following examples show how you can perform video encode.

H.264 Encode (NVIDIA accelerated encode)

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280,
height=(int)720, format=(fourcc)I420' ! nv_omx_h264enc ! qtmux !
filesink location=test.mp4 -v -e
```

VP8 Encode (NVIDIA accelerated encode)

```
gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280,
height=(int)720, format=(fourcc)I420' ! nv_omx_vp8enc ! qtmux !
filesink location=test.mp4 -v -e
```

MPEG-4 Encode (OSS software encode)

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280,
height=(int)720, format=(fourcc)I420' ! ffenc_mpeg4 ! qtmux ! filesink
location=test.mp4 -v -e
```

Image Encode

```
$ gst-launch-0.10 videotestsrc num-buffers=1 ! 'video/x-raw-yuv,
width=(int)1280, height=(int)720, format=(fourcc)I420' ! nvjpegenc !
filesink location=test.jpg -v -e
```

CAMERA CAPTURE

The default image capture application in the R19.3 release is `nvgstcapture-0.10`. For usage information enter the following command:

```
$ nvgstcapture-0.10 --help
```

The `nvgstcapture-0.10` application uses the `v4l2src` plugin to capture still images and video.

The following table shows USB camera support.

USB Camera Support	Feature
YUV	Preview display
	Image capture (VGA, 640 x 480)
	Video capture (480p, 720p, H.264/VP8 encode)
MJPEG	Preview display
	Image capture VGA, 640 x 480 720p, 1280 x 720
	Video capture (480p, 720p, 1080p, MJPEG encode)

raw-yuv Capture (I420 format) and preview display with xvimagesink

```
$ gst-launch-0.10 v4l2src device="/dev/video0" ! "video/x-raw-yuv,
width=640, height=480, format=(fourcc)I420" ! xvimagesink -v -e
```

VIDEO PLAYBACK

The default playback application in the R19.3 release is nvgstplayer-0.10. For usage information enter the following command:

```
$ nvgstplayer-0.10 --help
```

Video can be output to HD displays using the HDMI connector on the Jetson TK1 platform. The gstreamer-0.10 application supports currently the following video sinks:

HDMI Overlay Sink (Video playback on overlay in full-screen mode)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! nv_omx_hdmi_videosink –v -e
```

Xvimagesink (Windowed video playback)

```
$ gst-launch-0.10 filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv
! xvimagesink –v -e
```

VIDEO FORMAT CONVERSION

The NVIDIA proprietary `nvvidconv` gstreamer-0.10 plug-in allows you to convert between OSS (raw) video formats and NVIDIA video formats. The `nvvidconv` plug-in currently supports the format conversions described in this section.

raw-yuv Input Formats

Currently `nvvidconv` supports the following raw-yuv input formats: I420, YV12, YUY2, UYVY, YVYU, Y444, and NV12.

Converting raw-yuv to nv-yuv

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280,
height=(int)720, format=(fourcc)YUY2' ! nvvidconv ! 'video/x-nv-yuv' !
nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -v -e
```

Converting raw-yuv to nvrm-yuv

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280,
height=(int)720, format=(fourcc)YUY2' ! nvvidconv ! 'video/x-nv-yuv' !
nv_omx_h264enc ! qtmux ! filesink location=test.mp4 -v -e
```

raw-gray Input Formats

Currently `nvvidconv` supports the GRAY8 raw-gray input format.

Converting raw-gray to nv-yuv

```
$ gst-launch-0.10 videotestsrc num-buffers=300 ! 'video/x-raw-gray,
bpp=(int)8, depth=(int)8, width=(int)640, height=(int)480,
framerate=(fraction)30/1' ! nvvidconv ! 'video/x-nv-yuv,
format=(fourcc)I420' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -v -e
```

Converting raw-gray to nvrm-yuv

```
$ gst-launch-0.10 videotestsrc num-buffers=300 ! 'video/x-raw-gray,
bpp=(int)8, depth=(int)8, width=(int)640, height=(int)480,
framerate=(fraction)30/1' ! nvvidconv ! 'video/x-nvrm-yuv,
format=(fourcc)I420' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -v -e
```

raw-yuv Output Formats

Currently `nvvidconv` supports the following raw-yuv output formats: I420, YUY2, UYVY, and YVYU.

Converting nv-yuv to raw-yuv

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4 ! qtdemux name=demux
! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! xvimagesink –v -e
```

Converting nvrm-yuv to raw-yuv

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4 ! qtdemux name=demux
! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! 'video/x-raw-yuv,
format=(fourcc)UYVY' ! xvimagesink –v -e
```

raw-gray Output Formats

Currently `nvvidconv` supports the GRAY8 raw-gray output format.

Converting nv-yuv to raw-gray

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4 ! qtdemux name=demux
! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-raw-gray' !
ffmpegcolorspace ! xvimagesink -v -e
```

Converting nvrm-yuv to raw-gray

```
gst-launch-0.10 filesrc location=640x480_30p.mp4 ! qtdemux name=demux !
nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! 'video/x-raw-gray' !
ffmpegcolorspace ! xvimagesink -v -e
```

RGB Output Formats

Currently `nvvidconv` supports the following RGB output formats: BGRA, RGBA, BGRx, and RGBx.

Converting nv-yuv to raw-rgb

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4! qtdemux name=mux !
nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! ximagesink -v -e
```

Converting nvrm-yuv to raw-rgb

```
$ gst-launch-0.10 filesrc location=640x480_30p.mp4! qtdemux name=mux !
nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! ximagesink -v -e
```

VIDEO SCALING

The NVIDIA proprietary `nvvidconv` gstreamer-0.10 plug-in also allows you to perform video scaling. The `nvvidconv` plug-in currently supports scaling with the format conversions described in this section.

raw-yuv Input Formats

Currently `nvvidconv` supports the following raw-yuv input formats for scaling: I420, YUY2, UYVY, YVYU, Y444, and NV12.

Converting raw-yuv to nv-yuv with scaling

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280,
height=(int)720, format=(fourcc)I420' ! nvvidconv ! 'video/x-nv-yuv,
width=(int)640, height=(int)480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -v -e
```

Converting raw-yuv to nvrm-yuv with scaling

```
$ gst-launch-0.10 videotestsrc ! 'video/x-raw-yuv, width=(int)1280,
height=(int)720, format=(fourcc)NV12' ! nvvidconv ! 'video/x-nvrm-yuv,
width=(int)640, height=(int)480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -v -e
```

raw-gray Input Formats

Currently `nvvidconv` supports the GRAY8 raw-gray input format for scaling.

Converting raw-gray to nv-yuv with scaling

```
$ gst-launch-0.10 videotestsrc num-buffers=300 ! 'video/x-raw-gray,
bpp=(int)8, depth=(int)8, width=(int)1280, height=(int)720,
framerate=(fraction)30/1' ! nvvidconv ! 'video/x-nv-yuv,
width=(int)640, height=(int)480, format=(fourcc)I420' ! nv_omx_h264enc
! qtmux ! filesink location=test.mp4 -v -e
```

Converting raw-gray to nvrm-yuv with scaling

```
$ gst-launch-0.10 videotestsrc num-buffers=300 ! 'video/x-raw-gray,
bpp=(int)8, depth=(int)8, width=(int)1920, height=(int)1080,
framerate=(fraction)30/1' ! nvvidconv ! 'video/x-nvrm-yuv,
width=(int)640, height=(int)480, format=(fourcc)I420' ! nv_omx_h264enc
! qtmux ! filesink location=test.mp4 -v -e
```

raw-yuv Output Formats

Currently `nvvidconv` supports the following raw-yuv output formats for scaling: I420, YUY2, UYVY, and YVYU.

Converting nv-yuv to raw-yuv with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux
name=demux ! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-
raw-yuv, width=(int)640, height=(int)480, format=(fourcc)YUY2' !
xvimagesink –v -e
```

Converting nvrm-yuv to raw-yuv with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux
name=demux ! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv !
'video/x-raw-yuv, width=(int)640, height=(int)480, format=(fourcc)UYVY'
! xvimagesink –v -e
```

raw-gray Output Formats

Currently `nvvidconv` supports the GRAY8 raw-gray output format for scaling.

Converting nv-yuv to raw-gray with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux
name=demux ! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-
raw-gray, bpp=(int)8, depth=(int)8, width=(int)320, height=(int)240' !
ffmpegcolorspace ! xvimagesink -v -e
```

Converting nvrm-yuv to raw-gray

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux
name=demux ! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv !
'video/x-raw-gray, bpp=(int)8, depth=(int)8, width=(int)640,
height=(int)480' ! ffmpegcolorspace ! xvimagesink -v -e
```

RGB Output Formats

Currently `nvvidconv` supports the following RGB output formats for scaling: BGRA, RGBA, BGRx, and RGBx.

Converting nv-yuv to raw-rgb with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4! qtdemux name=mux !
nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-raw-rgb,
width=(int)640, height=(int)480' ! ximagesink -v -e
```

Converting nvrm-yuv to raw-rgb

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4! qtdemux name=mux !
nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! 'video/x-raw-rgb,
width=(int)640, height=(int)480' ! ximagesink -v -e
```

NVIDIA Input and Output Formats

Currently `nvvidconv` supports the NVIDIA input and output formats for scaling described in the following table:

Format	Description
NV12	NVIDIA gst-openmax decoder output format.
I420	NVIDIA gst-openmax encoder input format.

Scaling nv-yuv

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux
! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-nv-yuv,
width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -v -e
```

Converting nv-yuv to nvrm-yuv with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux
! nv_omx_h264dec ! 'video/x-nv-yuv' ! nvvidconv ! 'video/x-nvrm-yuv,
width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -v -e
```

Scaling nvrm-yuv

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux
! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! 'video/x-nvrm-yuv,
width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -v -e
```

Converting nvrm-yuv to nv-yuv with scaling

```
$ gst-launch-0.10 filesrc location=1280x720_30p.mp4 ! qtdemux name=mux
! nv_omx_h264dec ! 'video/x-nvrm-yuv' ! nvvidconv ! 'video/x-nv-yuv,
width=640, height=480' ! nv_omx_h264enc ! qtmux ! filesink
location=test.mp4 -v -e
```

VIDEO TRANSCODE

You can perform video transcoding between the following video formats.

H.264 Decode to VP8 Encode (NVIDIA-accelerated decode to NVIDIA-accelerated encode)

```
$ gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! nv_omx_vp8enc ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -v -e
```

VP8 Decode to H.264 Encode (NVIDIA-accelerated decode to NVIDIA-accelerated encode)

```
$ gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_vp8dec ! nv_omx_h264enc ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -v -e
```

MPEG-4 Decode to VP8 Encode (NVIDIA-accelerated decode to NVIDIA-accelerated encode)

```
$ gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_mpeg4dec ! nv_omx_vp8enc ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -v -e
```

MPEG-4 Decode to H.264 Encode (NVIDIA-accelerated decode to NVIDIA-accelerated encode)

```
$ gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_mpeg4dec ! nv_omx_h264enc ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -v -e
```

H.264 Decode to MPEG-4 Encode (NVIDIA-accelerated decode to OSS software encode)

```
$ gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! ffenc_mpeg4 ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -v -e
```

VP8 Decode to MPEG-4 Encode (NVIDIA-accelerated decode to OSS software encode)

```
gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_vp8dec ! ffenc_mpeg4 ! qtmux
name=mux ! filesink location=<Transcoded_filename.mp4> demux.audio_00 !
queue ! aacparse ! mux.audio_00 -v -e
```

H.264 Decode to Theora Encode (NVIDIA-accelerated decode to OSS software encode)

```
$ gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_h264dec ! theoraenc ! oggmux
name=mux ! filesink location=<Transcoded_filename.ogg> demux.audio_00 !
queue ! faad ! audioconvert ! vorbisenc ! mux. -v -e
```

VP8 Decode to Theora Encode (NVIDIA-accelerated decode to OSS software encode)

```
$ gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_vp8dec ! theoraenc ! oggmux
name=mux ! filesink location=<Transcoded_filename.ogg> demux.audio_00 !
queue ! faad ! audioconvert ! vorbisenc ! mux. -v -e
```

MPEG-4 Decode to Theora Encode (NVIDIA-accelerated decode to OSS software encode)

```
$ gst-launch filesrc location=<filename.mp4> ! qtdemux name=demux
demux.video_00 ! queue ! nv_omx_mpeg4dec ! theoraenc ! oggmux
name=mux ! filesink location=<Transcoded_filename.ogg> demux.audio_00 !
queue ! faad ! audioconvert ! vorbisenc ! mux. -v -e
```

Notice

Trademarks

Copyright