Rockchip Gstreamer User Guide

ID: RK-YH-YF-921

Release Version: V1.0.1

Release Date: 2022-02-24

Security Level: □Top-Secret □Secret □Internal ■Public

Rockchip Electronics Co., Ltd.

No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This document is going to introduce the ways to build and test Gstreamer and related plugins.

Product Version

Chipset	Version
RK356X	1.14.x
RK3588	1.18.x

Intended Audience

This document (this guide) is mainly intended for:

Technical support engineers

Software development engineers

Revision History

Date	Version	Author	修改说明
2022-01-06	V1.0.0	Jair Wu	Initial version
2022-02-24	V1.0.1	Jair Wu	Fix a wrong command option

Contents

1. Source Code and Build

1.1 Path of the Source Code

The source code of Gstreamer and related plug-ins can be downloaded from the network, and then generated by adding the patches provided by RK. For details, please refer to `<SDK>/buildroot/package/gstreamer1/`.

The source code of the rkmpp plugin is in the directory: `<SDK>/external/gstreamer-rockchip`.

1.2 Build

Buildroot:

Enable related macros (which are enabled by default) and build them in the SDK root directory directly. It supported to select building versions, such as `BR2_PACKAGE_GSTREAMER1_14` and `BR2_PACKAGE_GSTREAMER1_18`.

```
BR2_PACKAGE_MPP=y
BR2_PACKAGE_MPP_ALLOCATOR_DRM=y
BR2_PACKAGE_GSTREAMER1_ROCKCHIP=y
BR2_PACKAGE_LINUX_RGA=y
BR2_PACKAGE_CA_CERTIFICATES=y
BR2_PACKAGE_LIBSOUP_SSL=y
BR2_PACKAGE_GSTREAMER1=y
BR2_PACKAGE_GST1_PLUGINS_BASE=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_ALSA=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_VIDEOCONVERT=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_VIDEOTESTSRC=y
BR2_PACKAGE_GST1_PLUGINS_GOOD=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_AUDIOPARSERS=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_AUTODETECT=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_DEINTERLACE=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_FLV=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_GDKPIXBUF=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_MATROSKA=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_MPG123=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_SOUPHTTPSRC=y
BR2_PACKAGE_GST1_PLUGINS_BAD=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_DVBSUBOVERLAY=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_DVDSPU=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_JPEGFORMAT=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_KMS=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_MPEGDEMUX=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_MPEG2ENC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_VIDEOPARSERS=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_ADPCMDEC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_ADPCMENC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_FAAD=y
BR2_PACKAGE_GST1_PLUGINS_UGLY=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_ASFDEMUX=y
```

```
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_DVDLPCMDEC=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_DVDSUB=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_MPEG2DEC=y
...
```

The complete list of plugins can be found in menuconfig->Target packages->Audio and video applications->gstreamer 1.x.

Debian:

The source code should be placed on the board, and make sure that the `debian` directory exists in the root directory of the source code. Enter the source root directory and execute:

```
# 1 Update software sources
apt update
# 2 Install dependent libraries
apt build-dep .
# 3 Optional: start building the deb installation package
dpkg-buildpackage -b -d -uc -us
# After the building is completed, the deb installation package will be generated
in the upper directory, which can be installed by using dpkg -i xxx.deb.
# 3 Optional: build and install
meson build && ninja -C build install
```

It is generally recommended to use the first way to build the deb installation package, which can ensure that the options such as compilation and installation are unified.

Note: Some compilation options depend on the macro definitions in header files such as `video-format.h`, so you need to install the `libgstreamer-plugins-base1.0-dev` package first to ensure the headers such as `video-format.h` to the latest and ensure that certain features are turned on.

2. Commonly Used Commands

- gst-launch-1.0

  Gstreamer launcher for quickly building pipelines, examples are as follows:

```
  # Generate a video by videotestsrc, and display it through xvimagesink
  gst-launch-1.0 videotestsrc ! xvimagesink
```

- gst-play-1.0

  Gstreamer player, used to play various streaming media, examples are as follows:

```
  # Play test.mp4 and display it through xvimagesink
  gst-play-1.0 test.mp4 --videosink=xvimagesink
  # Commonly used command options
--flags          # bit0: video, bit1: audio, bit2: subtitle, such as --flags=1
means only video is played
  --videosink    # specify videosink
  --audiosink    # specify audiosink
  --use-playbin3 # use playbin3, otherwise use playbin2
```

- gst-inspect-1.0

  A finder to list all plugins or detailed information of a plugin, for example:

```
  # Without any parameters, list all plugins
  gst-inspect-1.0
  # List all information about the xvimagesink plugin
  gst-inspect-1.0 xvimagesink
```

- Enable log function

```
  #Set environment variables
  export GST_DEBUG=2
  #Or specified before the command, and invalid after the end of the command
  GST_DEBUG=2 gst-play-1.0 ...

  #Specify different log levels for different modules, support wildcards,
fpsdisplaysink is specified as DEBUG (5), xvimage* is specified as FIXME (3),
others are specified as WARNING (2)
  GST_DEBUG=2,fpsdisplaysink:5,xvimage*:3
```

The log levels are divided into ERROR(1), WARNING(2), FIXME(3), INFO(4), DEBUG(5), LOG(6), TRACE(7) and so on.

3. Commonly Used Plugins

3.1 Source

Refers to plugins that can generate data but cannot receive data.

- filesrc

  Read data from a file, an example is as follows:

```
  gst-launch-1.0 filesrc location=/tmp/test ! filesink location=/tmp/test2
```

- videotestsrc

  Generate video data, an example is as follows:

```
  # Output video through default format
  gst-launch-1.0 videotestsrc ! xvimagesink
  # Output the video through the specified format
  gst-launch-1.0 videotestsrc ! "video/x-raw,width=1920,height=1080,format=
(string)NV12" !xvimagesink
```

3.2 Sink

Refers to plugins that accept data but do not send data.

- filesink

  Save the received data as a file, an example is as follows:

```
gst-launch-1.0 filesrc location=/tmp/test ! filesink location=/tmp/test2
```

- fakesink

  Discard all the received data, an example is as follows:

```
gst-launch-1.0 filesrc location=/tmp/test ! fakesink
```

- xvimagesink

  Video Sink, receive video and display, which is implemented by the X11 interface, an example is as follows:

```
gst-launch-1.0 videotestsrc ! xvimagesink
```

- kmssink

  Video Sink, receives video and displays it. It is implemented through the kms interface and requires exclusive hardware decoding layer. The example is as follows:

```
gst-launch-1.0 videotestsrc ! kmssink
# Common commands
connector-id        #specifies the screen
plane-id            #pecifies the hardware layer
render-rectangle    #specifies the rendering range
```

- waylandsink

  Video Sink, receives video and displays, it is implemented through the wayland interface, the example is as follows:

```
gst-launch-1.0 videotestsrc ! waylandsink
```

- rkximagesink

  Video Sink, receives video and displays it, zero-copy and other functions are implemented through the drm interface, and with better performance, but requires exclusive hard decoding layer. An example is as follows:

```
gst-launch-1.0 videotestsrc ! rkximagesink
```

- fpsdisplaysink

  Video Sink, receives the video and counts the frame rate, and at the same time transfers the video to the next level Sink for display, an example is as follows:

```
gst-launch-1.0 videotestsrc ! fpsdisplaysink video-sink=xvimagesink
```

4. Command Examples

```
GST_DEBUG=fpsdisplaysink:6 gst-play-1.0 --flags=3 --videosink="fpsdisplaysink
video-sink=xvimagesink signal-fps-measurements=true text-overlay=false
sync=false" --audiosink="alsasink device= hw:0,0" test.mp4

GST_DEBUG=fpsdisplaysink:6          #Set the log level of Gstreamer, specify 6
for fpsdisplaysink, and turn off log of other modules
--flags=3                           #Turn off subtitles
--videosink="fpsdisplaysink ..."    #Specify fpsdisplaysink for frame rate
statistics
  video-sink=xvimagesink            #Specifies xvimagesink as the final display
Sink
  signal-fps-measurements=true      #Enable FPS statistics
  text-overlay=false                #Turns off the frame rate display, "true"
will overlie the frame rate information on the screen
  sync=false                        #Turns off clock synchronization
--audiosink="alsasink ..."          #Specifies alsasink as audio sink
  device=hw:0,0                     #Specifies the sound card hw:0,0
```

5. AFBC

AFBC stands for ARM Frame Buffer Compression, which is a compression format used to save bandwidth. Currently, the encoding formats of AFBC supported by the mppvideodec plugin are: H264, H265, VP9, and the supported color formats are NV12, NV12 10bit. The way to open is as follows:

```
export GST_MPP_VIDEODEC_DEFAULT_ARM_AFBC=1

# AFBC should use the Cluster layer to play
# Or use waylandsink, which can be composited to Esmart/Smart layers using GPU
# GST_DEBUG=*mpp*:4, turn on the DEBUG switch of the mpp plugin. You can check
whether AFBC is successfully turned on through the log printed by rkmpp. If AFBC
is not printed, it may be unsuccessfully turned on or the format does not support
compression
GST_DEBUG=*mpp*:4,fpsdisplaysink:6 gst-play-1.0 --flags=3 --
videosink="fpsdisplaysink video-sink=waylandsink text-overlay=false signal-fps-
measurements=true" test.mp4
GST_DEBUG=*mpp*:4,fpsdisplaysink:6 gst-play-1.0 --flags=3 --
videosink="fpsdisplaysink video-sink=\"kmssink plane-id=101\" text-overlay=false
signal-fps- measurements=true" test.mp4
```

6. Subtitle

When subtitles are turned on, there will be lags. Usually, subtitle synthesis requires intercept some images from the video and convert them to RGB, and then synthesize subtitles and then convert them back to the source format before sending them for display. That is, the time-consuming of decoding also needs to consider the time-consuming of subtitle synthesis. , causing the overall frame rate to drop. Use the gst-play-1.0 command to test and subtitles can be turned off with `--flags=3`. Subtitles should be implemented independently of the video layer using frameworks such as QT.

7. Layers Assignment

When using rkximagesink or kmssink, it is required to have a exclusive hardware layer, and the plug-in will automatically find the layer to play, but the automatically found layer may not meet the requirements, so you have to manually specify the layer, the way is as follows:

```
gst-play-1.0 --flags=3 test.mp4 --videosink="kmssink plane-id=117"
```

The 117 is the ID of the target layer, which can be confirmed through the `/sys/kernel/debug/dri/0/state` node. You can use the following command to list all layers:

```
root@linaro-alip:/# cat /sys/kernel/debug/dri/0/state | grep "plane\["
plane[57]: Smart1-win0
plane[71]: Cluster1-win0
plane[87]: Smart0-win0
plane[101]: Cluster0-win0
plane[117]: Esmart1-win0
plane[131]: Esmart0-win0
# You can also use cat /sys/kernel/debug/dri/0/state directly to list complete
information
```

The plane[xx] is the plane-id. Usually, different layers support different formats. For example, Cluster supports AFBC, but Esmart does not support AFBC. Please refer to the datasheet or TRM for details.

8. FAQ

1. There is no lagging when playing 4K 30FPS, but there is lagging when playing 4K 60FPS.

   Due to system load, DDR bandwidth and other issues, 4K 60FPS may not be achieved. You can try to enable AFBC, refer to the AFBC chapter. In addition, the synchronization function of subtitles and sink can be turned off, such as `gst-play-1.0 test.mp4 --flags=3 --videosink="waylandsink sync=false"`, when the frame rate cannot reach 60FPS, turning on sync will cause the video frame timestamps do not align with clocks resulting in obvious frame drop.

2. There is relatively lagging when playing some sources, and the CPU usage is very high.

   Currently hard decode supports H264, H265, VP8, VP9, MPEG. You can turn on DEBUG through `echo 0x100 > /sys/module/rk_vcodec/parameters/mpp_dev_debug` to see if the serial port or dmesg has decoded printing. If not, it may be a format not supported by the hard decode.

3. Some sources cannot be played, LOG is lagging and the progress is not printed or the progress is always 0

   You can try to use playbin3, like `gst-play-1.0 --flags=3 --use-playbin3 test.mp4`.

4. Flickering when playing 4K video after AFBC is turned on

First make sure to turn on performance mode, `echo performance | tee $(find /sys/ -name *governor)`. Then, confirm whether there is obvious scaling in the vertical direction, such as using the vertical screen to play the horizontal video, in this case, the AFBC performance is not as good as the non-AFBC performance.

5. Play with pictures but no sound

   You can manually specify the audiosink, such as `gst-play-1.0 --flags=3 test.mp4 --audiosink="alsasink device=hw:0,0"`. It is recommended to make sure it can work using basic testing tools such as aplay and then use gstreamer to test.