密级状态：绝密(　　)　　秘密(　　)　　内部(　　)　　公开(√)

# RK_Linux_Camera_Gstreamer 应用开发

（技术部，第二系统产品部）

| | | |
|---|---|---|
| 文件状态：<br>[　] 正在修改<br>[√] 正式发布 | 当前版本： | V1.0 |
| | 作　　者： | 陈潇、陈城 |
| | 完成日期： | 2017-12-12 |
| | 审　　核： | |
| | 完成日期： | 2017-12-12 |

福州瑞芯微电子股份有限公司

Fuzhou Rockchips Electronics Co . , Ltd

# 版 本 历 史

| 版本号 | 作者 | 修改日期 | 修改说明 | 备注 |
|---|---|---|---|---|
| V1.0 | 陈潇、陈城 | 2017-12-12 | 发布初版 | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

# 目 录

# 1.1 概述

GStreamer 是用来构建流媒体应用的开源多媒体框架(framework)，其目标是要简化音/视频应用程序的开发，目前已经能够被用来处理像 MP3、Ogg、MPEG1、MPEG2、AVI、Quicktime 等。其是一个 libraries 和 plugins 的集合，用于帮助实现各种类型的多媒体应用程序，比如播放器，转码工具，多媒体服务器等。

利用 Gstreamer 编写多媒体应用程序，就是利用 elements 构建一个 pipeline。element 是一个对多媒体流进行处理的 object，比如如下的处理：

（1）读取文件

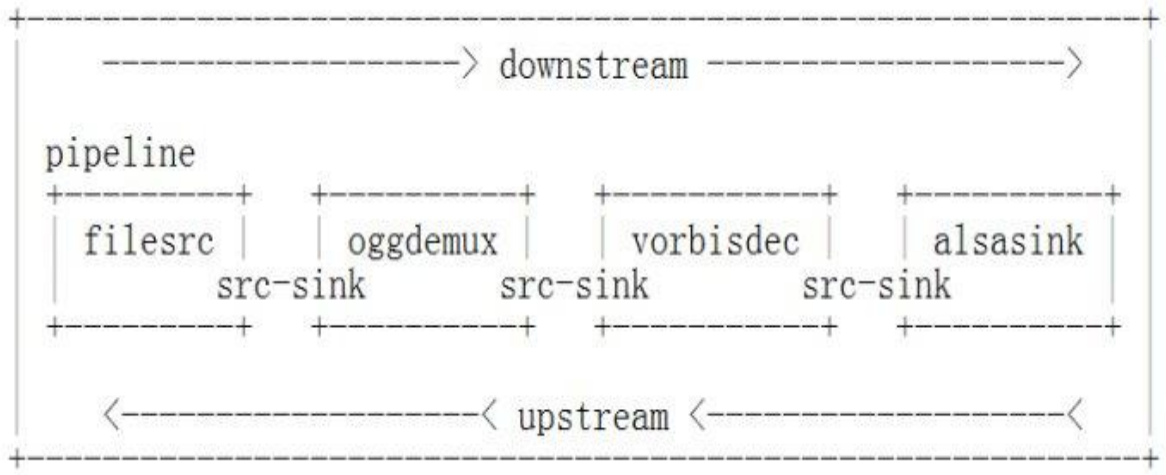
（2）不同格式的编解码

（3）从硬件采集设备上的数据

（4）在硬件设备上播放多媒体

（5）多个流的复用

elements 的输入叫做 sink pads，输出叫做 source pads。应用程序通过 pad 把 element 连接起来构成 pipeline，如下图所示，其中顺着流的方向为 downstream，相反方向是 upstream。



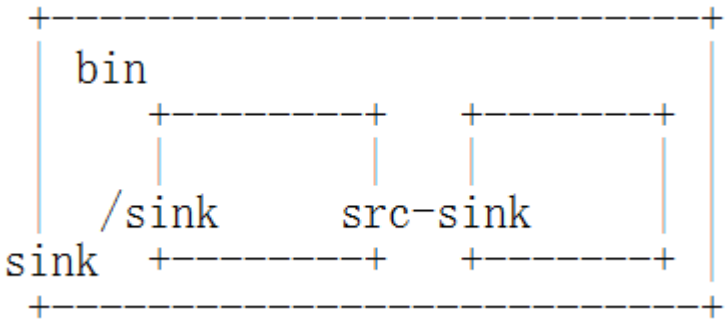

**Elements**

element 是 pipeline 的最小组成部分。element 提供了多个 pads，或者为 sink，或者为 source。一个 element 有四种可能的状态，分别是 NULL，READY，PAUSED，PLAYING。NULL 和 READY 状态下，element 不对数据做任何处理，PLAYING 状态对数据进行处理，PAUSE 状

态介于两者之间，对数据进行 preroll。应用程序通过函数调用控制 pipeline 在不同状态之间进行转换。

**Bin**

bin 是由多个 element 构成的特殊的 element，用图来说明：




**Pipeline**

pipeline 是具备如下特性的特殊的 bin：

（1）选择并管理一个全局的时钟。

（2）基于选定的时钟管理 running_time。running_time 用于同步，指的是 pipeline 在 PLAYING 状态下花费的时间。

（3）管理 pipeline 的延迟。

（4）通过 GstBus 提供 element 与应用程序间的通讯方式。

（5）管理 elements 的全局状态，比如 EOS，Error 等。

**Caps**

Caps，也就是媒体类型，采用 key/value 对的列表来描述。key 是一个字符串类型，value 的类型可能是 int/float/string 类型的 single/list/range。Data flow and events 除了数据流，还有 events 流。与数据流不同，events 的传送方向既有 downstream 的，也有 upstream 的。events 用于传递 EOS，flushing，seeking 等消息。有的 events 必须和 data flow 一起进行 serialized。serialized 的 events 比如 TAG，非 serialized 的 events 比如 FLUSH。

**Pipeline construction**

gst_pipeline_create()函数用于创建一个 pipeline，gst_bin_add()函数用于向 pipeline 中添加 element，gst_bin_remove()函数用于从 pipeline 中移除 element。

gst_element_get_pad（）函数用于检索 pipeline 中的 element。gst_pad_link（）函数用于把 pads 连接在一起。有的 element 会在数据流开始传送的时候创建新的 pads，通过调用函数 g_signal_connect（）函数，能在新的 pads 被创建的时候接收到消息。由于处理的数据互相不兼容，有的 elements 是不能被连接到一起的。gst_pad_get_caps（）函数查询 element 能够处理的数据类型。

## 1.2 Camera 插件

gst-inspect-1.0：显示所有支持的 gstreamer 插件

```
[root@rockchip:/]# gst-inspect-1.0
avi:  avidemux: Avi demuxer
avi:  avimux: Avi muxer
avi:  avisubtitle: Avi subtitle parser
autodetect:  autovideosink: Auto video sink
autodetect:  autovideosrc: Auto video source
autodetect:  autoaudiosink: Auto audio sink
autodetect:  autoaudiosrc: Auto audio source
coreelements:  capsfilter: CapsFilter
coreelements:  concat: Concat
coreelements:  dataurisrc: data: URI source element
coreelements:  downloadbuffer: DownloadBuffer
coreelements:  fakesrc: Fake Source
coreelements:  fakesink: Fake Sink
coreelements:  fdsrc: Filedescriptor Source
coreelements:  fdsink: Filedescriptor Sink
coreelements:  filesrc: File Source
coreelements:  funnel: Funnel pipe fitting
coreelements:  identity: Identity
coreelements:  input-selector: Input selector
coreelements:  output-selector: Output selector
coreelements:  queue: Queue
coreelements:  queue2: Queue 2
coreelements:  filesink: File Sink
coreelements:  tee: Tee pipe fitting
coreelements:  typefind: TypeFind
coreelements:  multiqueue: MultiQueue
coreelements:  valve: Valve element
coreelements:  streamiddemux: Streamid Demux
pbtypes:  GstVideoMultiviewFlagsSet (GstDynamicTypeFactory)
id3demux:  id3demux: ID3 tag demuxer
videobox:  videobox: Video box filter
playback:  playbin: Player Bin 2
playback:  playbin3: Player Bin 3
playback:  playsink: Player Sink
playback:  subtitleoverlay: Subtitle Overlay
playback:  streamsynchronizer: Stream Synchronizer
playback:  decodebin: Decoder Bin
playback:  decodebin3: Decoder Bin 3
```

gst-inspect-1.0 插件名：显示某个插件的信息

```
[root@rockchip:/]#
[root@rockchip:/]# gst-inspect-1.0 v4l2src
Factory Details:
  Rank                     primary (256)
  Long-name                Video (video4linux2) Source
  Klass                    Source/Video
  Description              Reads frames from a Video4Linux2 device
  Author                   Edgard Lima <edgard.lima@gmail.com>, Stefan Kost <ensonic@users.sf.net>

Plugin Details:
  Name                     video4linux2
  Description              elements for Video 4 Linux
  Filename                 /usr/lib/gstreamer-1.0/libgstvideo4linux2.so
  Version                  1.12.2
  License                  LGPL
  Source module            gst-plugins-good
  Source release date      2017-07-14
  Binary package           GStreamer Good Plug-ins source release
  Origin URL               Unknown package origin

GObject
 +----GInitiallyUnowned
       +----GstObject
             +----GstElement
                   +----GstBaseSrc
                         +----GstPushSrc
                               +----GstV4l2Src
```

Camera 数据采集处理插件：v4l2src、camerabin

V4l2src：功能比较简单 elements 插件，基于 v4l2 协议进行数据采集，因此 camera 设备需支持 v4l2 协议，目前 rk linux 平台 usb camera、isp raw camera、hdmi in 等驱动都已支持 v4l2 协议。

Camerabin：由多个 element 构成的特殊的 element，功能比较强大，有拍照、录像、缩放、对焦等功能。

**V4l2src 插件预览 pipeline：**

gst-launch-1.0 v4l2src --gst-debug-level=3 device=/dev/video0 ! videoconvert ! video/x-raw,format=NV12,width=640,height=480 ! queue ! kmssink

gst-debug-level=3：打印等级，数值越大等级越高打印信息越多

typedef enum {

GST_LEVEL_NONE = 0, GST_LEVEL_ERROR = 1, GST_LEVEL_WARNING = 2, GST_LEVEL_FIXME = 3,

GST_LEVEL_INFO = 4, GST_LEVEL_DEBUG = 5, GST_LEVEL_LOG = 6, GST_LEVEL_TRACE = 7,

```
  /* add more */
  GST_LEVEL_MEMDUMP = 9,
  /* add more */
  GST_LEVEL_COUNT
} GstDebugLevel;
```

device=/dev/video0：指定打开的 camera 设备节点，默认 video0

videoconvert：将 src 数据格式转换为 sink 可以显示的数据格式

video/x-raw,format=NV12,width=640,height=480：sink 显示数据格式

queue：队列用来缓存数据防止 pipeline 堵塞

kmssink：显示插件

**Camerabin 插件预览 pipeline：**

gst-launch-1.0 -v -m camerabin

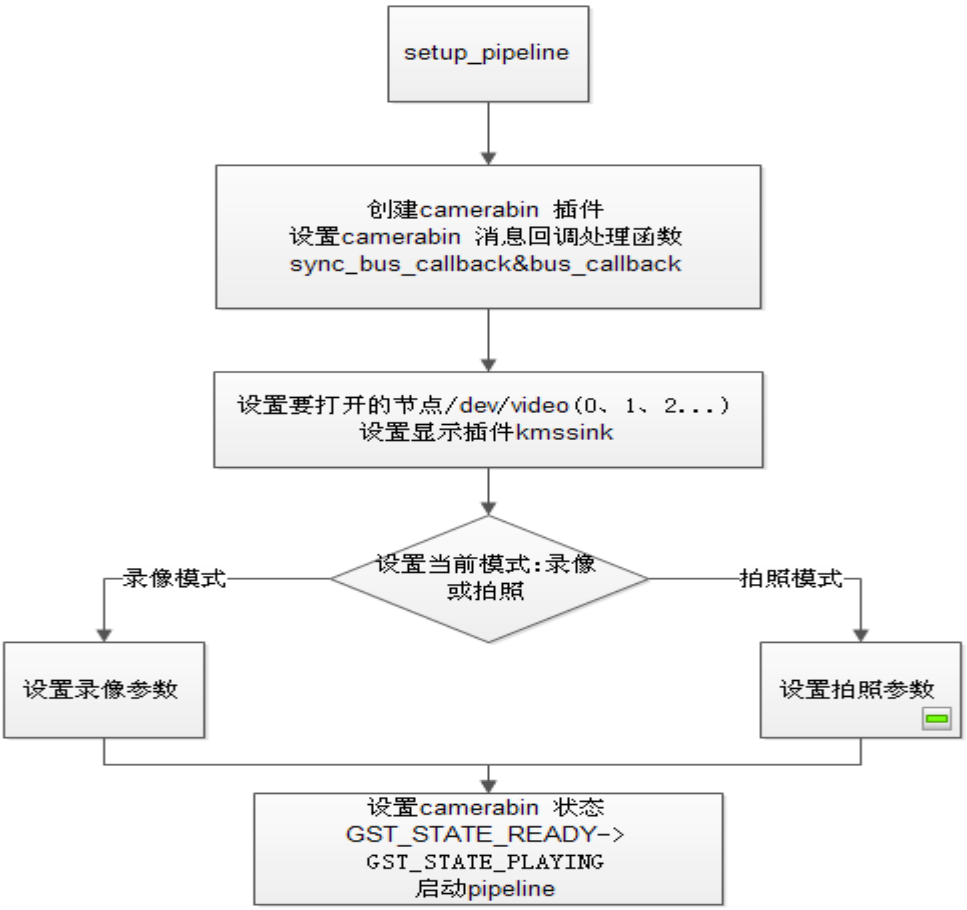
viewfinder-sink=kmssink

viewfinder-caps=video/x-raw,format=NV12,width=640,height=480

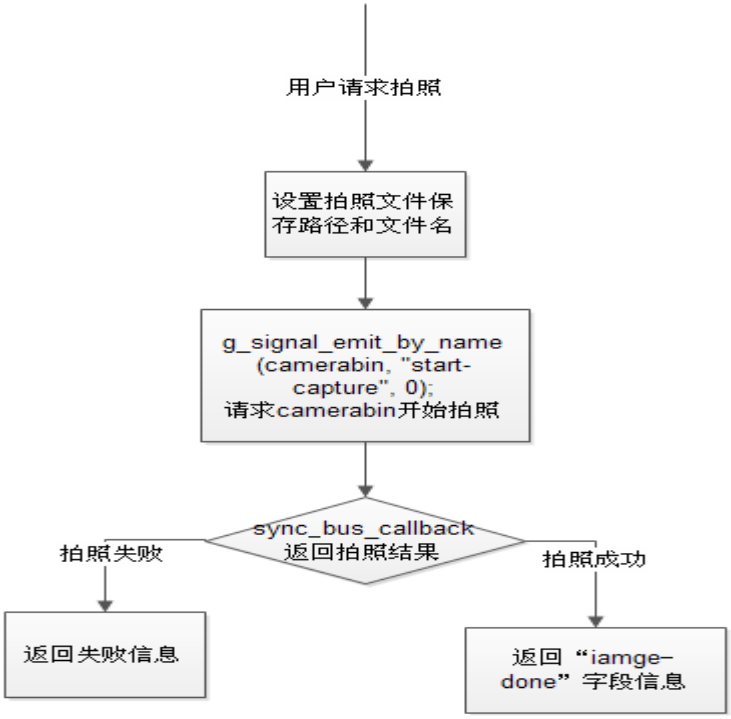
camerabin 由 v4l2src、videoconvert 等构成，也是基于 v4l2 协议。

**Camerabin 实现预览、拍照、录像流程：**

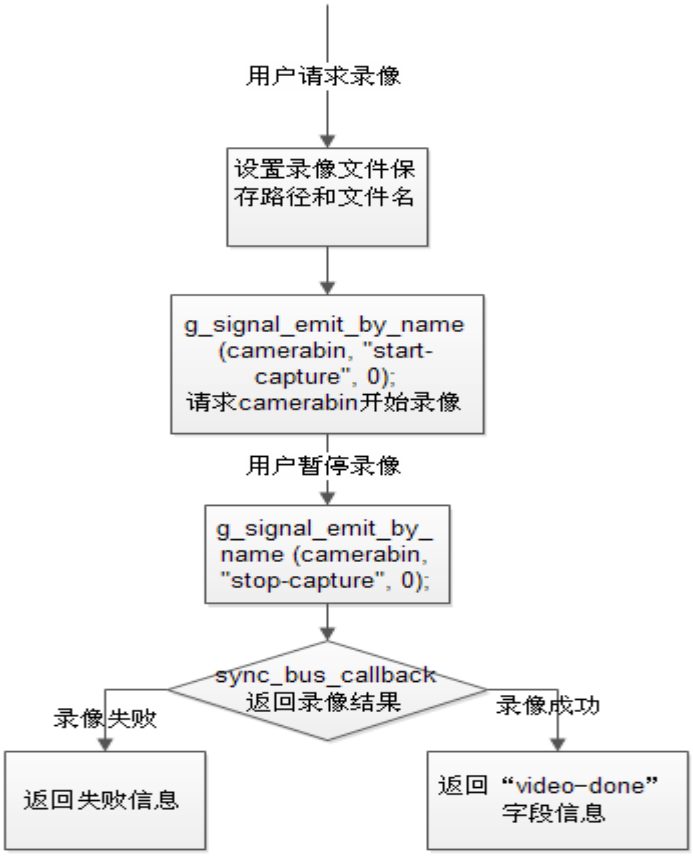
具体实现代码可参考 app/camera/目录下的源码



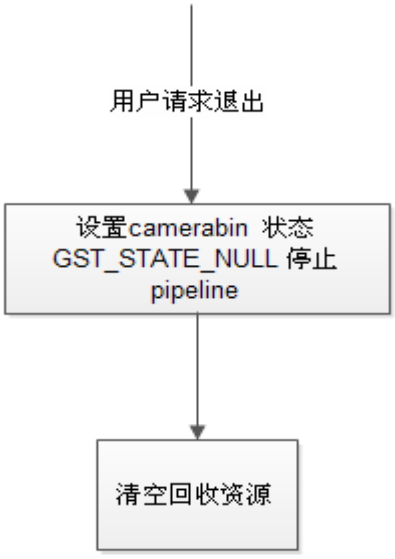

建立 pipeline 打开 camera 进行预览流程




拍照流程




录像流程




退出 pipeline 关闭 camera 流程

## 1.3 ISP 3A 移植与开发

**环境搭建与移植（rk buildroot sdk 已集成 gstreamer 可略过）：**

1. 首先需要搭建 Gstreamer 环境，如开发板上已经安装好 Gstreamer 环境，可忽略这一步骤。

   以下操作在 firefly 开发板 ubuntu 操作系统完成，安装所需的程序包：

   ```
   $apt-get install -y bison flex
   libffi-devlibmount-devlibpcre3 libpcre3-devzlib1g-devlibssl-dev gtk-doc-tools
   ```

2. 安装 ORC 支持库， 编译 gst-plugins-base 将会依赖这个库。

   下载安装包：

   https://gstreamer.freedesktop.org/src/orc/orc-0.4.27.tar.xz

   请运行以下命令：

   ```
   $./autogen.sh--prefix=/usr/lib
   $make
   $make install
   ```

3. 安装 GLIB 支持库

   下载安装包：

   http://ftp.acc.umu.se/pub/GNOME/sources/glib/2.52/glib-2.52.3.tar.xz

   请运行以下命令：

   ```
   $xz -d glib-2.52.3.tar.xz; tar xvf glib-2.52.3.tar
   $cd glib-2.52.3
   $./autogen.sh
   $make
   $make install
   ```

4. 安装 Gstreamer 程序包，需要下载以下几个软件源码包：

   下载以下安装包:https://gstreamer.freedesktop.org/src/

   ```
   gstreamer-1.12.2
   gst-plugins-base-1.12.2
   ```

gst-plugins-good-1.12.2

gst-plugins-bad-1.12.2

gst-plugins-ugly-1.12.2

请运行以下命令：

```
$cd gstreamer-1.12.2
$./autogen.sh
$make
$make install

$cd gst-plugins-base-1.12.2
$./autogen.sh
$make
$make install

$cd gst-plugins-good-1.12.2
$./autogen.sh
$make
$make install

$cd gst-plugins-bad-1.12.2
$./autogen.sh
$make
$make install

$cd gst-plugins-ugly-1.12.2
$./autogen.sh
```

```
$make
$make install
```

**安装使用 Gstreamer rkisp element：**

首先解压缩基于 Gstreamer 的 rkisp 插件，并根据不同环境进行编译安装。

Firefly 开发板上，请运行以下命令：

```
$ ./autogen.sh --prefix=/usr/local --enable-gst --enable-rkiq
$make
$ make install
```

在没有编译环境的开发板上，请先在 PC 端运行以下命令进行交叉编译后，再将编译生成的库安装到开发板上，这里以 excavator 开发板为例：

```
$ export PATH=/path/to/cross-compiler:$PATH
$ CC=aarch64-linux-gcc ./autogen.sh --prefix=./out --host=aarch64-linux
--enable-gst --enable-rkiq
$ make
$ make install
```

查看 rksip element 详细信息，请运行以下命令：

```
$ gst-inspect-1.0 rkisp
```

Gstreamer rkisp 插件作为一个 source element，通过 v4l2 框架从 ISP 获取数据流。以下示例命令将数据流保存到文件：

$ gst-launch-1.0 rkisp io-mode=1 num-buffers=10 ! video/x-raw, format=NV12, width=640, height=480, framerate=30/1 ! videoscale ! filesink location=/tmp/output.dat

创建预览窗口，实时显示数据流：

$ gst-launch-1.0 rkisp io-mode=4 ! video/x-raw, format=NV12, width=640, height=480, framerate=30/1 ! videoconvert ! autovideosink

**ISP 3A 开发：**

在不使用 Gstreamer 插件，自行编写 v4l2 应用的情况下，可以通过以下方式启动 3A 效果。

1. 包含头文件：

```
#include <rkisp_interface.h>
```

2. 提供了两个接口 rkisp_start/rkisp_stop。rkisp_start 需要在 *VIDIOC_STREAMON 命令之前调用*, rkisp_stop 建议在 VIDIOC_STREAMOFF 之前调用。

```
int rkisp_start(void* &engine, int vidFd, const char* ispNode, const char* tuningFile);
int rkisp_stop(void* &engine);
```

3. 编译链接提供的动态库 librkisp.so