# Camera 应用开发

# 参考手册 V1.0

正点原子

正点原子公司名称 ：广州市星翼电子科技有限公司

原子哥在线教学平台：www.yuanzige.com

开源电子网 / 论坛 ：http://www.openedv.com/forum.php

正点原子淘宝店铺 ：https://openedv.taobao.com

正点原子官方网站 ：www.alientek.com

正点原子 B 站视频 ：

https://space.bilibili.com/394620890

电话：020-38271790 传真：020-36773971

请关注正点原子公众号，资料发布更新我们会通知。

请下载原子哥 APP，数千讲视频免费学习，更快更流畅。

扫码关注正点原子公众号

扫码下载“原子哥”APP

# 文档更新说明

| 版本 | 版本更新说明 | 负责人 | 校审 | 发布日期 |
|---|---|---|---|---|
| V1.0 | 初稿： | 正点原子 linux 团队 | 正点原子 linux 团队 | 2024.03.26 |

# 目录

# 前言

鉴于正点原子 RK 系列用户对于摄像头开发应用不熟悉，好几个用户使用 RK3568 问过如何玩摄像头。笔者认为玩 RK 系列的用户都有一定的 Linux 基础了，RK 系列不适合新手入门，所以没写摄像头开发文档，响应用户的需要。正点原子编写了 Camera 应用开发手册，方便用户开发摄像头应用！

本文档基于 RK3588/RK3568 等开发板使用摄像头，支持 USB 摄像头及 MIPI 摄像头。网络摄像头本次不在讨论范围。

# 免责声明

本文档所提及的产品规格和使用说明仅供参考，如有内容更新，恕不另行通知；除非有特殊约定，本文档仅作为产品指导，所作陈述均不构成任何形式的担保。

为了得到最新版本的产品信息，请用户定时访问正点原子资料下载中心或者与淘宝正点原子旗舰店客服联系索取。感谢您的包容与支持。

# 第一章 摄像头概述

本章带大家查看摄像头节点，同时看看支持哪些摄像头。

### 1.1 查找 USB/MIPI 摄像头节点

MIPI 摄像头：正点原子 RK3568 支持 IMX415（800W），IMX335（500W），OV13850（1300W）。正点原子 RK3588 支持 IMX415（800W），暂时不支持其他 MIPI 摄像头（可询问客服技术，有没有支持其他摄像头）。

USB 摄像头：正点原子 RK3588/3568 都支持 UVC 免驱摄像头。

MIPI 摄像头可以使用 ISP 调优，默认已经调优，支持输出可用分辨率范围内任意分辨率，最小是 ISP 输出的最小分辨率。如 IMX415 支持 3840x2160 输出，那么可以设置 1200*1200 等分辨率，支持自定义。但是 USB 摄像头，只能支持可用分辨率，如 640*480，1280*720 等，是固定的，具体查看 USB 摄像头可用的分辨率。

以 ATK-DLRK3568 为例：

查看 MIPI 摄像头（插上 MIPI 摄像头 IMX415 为例）和 USB 摄像头节点：

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


如上图，MIPI 摄像头由 rkisp-vir0 节点输出。一般第一个就是 main_path 节点，第二个就是 self_path 节点，关于摄像头链路讲解可以参考我们正点原子 RK/RV 系列的教程 https://www.bilibili.com/video/BV1dW4y1f7Qu?t=2.5&p=8。

如上面的 USB Camera，可以看到笔者使用的是 RGB Camera，节点是第一个就是 video9。

### 1.2 查看 USB/MIPI 摄像头支持分辨率

以 ATK-DLRK3568 为例：

由 1.1 小节可以知道，MIPI 摄像头的节点为 video0，另一个是 video1，我们以 video0 为例。

使用下面的指令查看 MIP 摄像头支持的分辨率是 3840*2160。注意下面是摄像头支持最大分辨率输出，并且 MIPI 摄像头是可以自定义输出分辨率的，并且支持多种格式输出。默认是 30fps。

```
v4l2-ctl -d /dev/video0 --list-formats-ext
```

```
root@ATK-DLRK356X:/#  v4l2-ctl -d /dev/video0 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture Multiplanar

        [0]: 'UYVY' (UYVY 4:2:2)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [1]: '422P' (Planar YUV 4:2:2)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [2]: 'NV16' (Y/CbCr 4:2:2)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [3]: 'NV61' (Y/CrCb 4:2:2)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [4]: 'YM16' (Planar YUV 4:2:2 (N-C))
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [5]: 'NV21' (Y/CrCb 4:2:0)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [6]: 'NV12' (Y/CbCr 4:2:0)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [7]: 'NM21' (Y/CrCb 4:2:0 (N-C))
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [8]: 'NM12' (Y/CbCr 4:2:0 (N-C))
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [9]: 'YU12' (Planar YUV 4:2:0)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [10]: 'YM24' (Planar YUV 4:4:4 (N-C))
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [11]: 'RGGB' (8-bit Bayer RGRG/GBGB)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [12]: 'GRBG' (8-bit Bayer GRGR/BGBG)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [13]: 'GBRG' (8-bit Bayer GBGB/RGRG)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [14]: 'BA81' (8-bit Bayer BGBG/GRGR)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [15]: 'RG10' (10-bit Bayer RGRG/GBGB)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [16]: 'BA10' (10-bit Bayer GRGR/BGBG)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [17]: 'GB10' (10-bit Bayer GBGB/RGRG)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [18]: 'BG10' (10-bit Bayer BGBG/GRGR)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [19]: 'RG12' (12-bit Bayer RGRG/GBGB)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [20]: 'BA12' (12-bit Bayer GRGR/BGBG)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [21]: 'GB12' (12-bit Bayer GBGB/RGRG)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
        [22]: 'BG12' (12-bit Bayer BGBG/GRGR)
                Size: Stepwise 32x16 - 3840x2160 with step 8/8
root@ATK-DLRK356X:/#
```

查看 USB 摄像头的分辨率。可以看到下面的 USB 只有两种输出格式。可用性范围就小了点。

由 1.1 小节可知，USB 摄像头的节点是 video9。

执行下面的指令，查看 USB 摄像头支持的格式、分辨率及帧率，USB 摄像头必需要严格按支持的分辨率设置。

```
v4l2-ctl -d /dev/video9 --list-formats-ext
```

```
root@ATK-DLRK356X:/# v4l2-ctl -d /dev/video9 --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture

        [0]: 'MJPG' (Motion-JPEG, compressed)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.040s (25.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.040s (25.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                Size: Discrete 1600x896
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x1024
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.040s (25.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 1024x576
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 960x720
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 800x600
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.040s (25.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.040s (25.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.040s (25.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
        [1]: 'YUYV' (YUYV 4:2:2)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.200s (5.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 1600x896
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 1280x1024
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 1024x576
                        Interval: Discrete 0.067s (15.000 fps)
                Size: Discrete 960x720
                        Interval: Discrete 0.067s (15.000 fps)
                Size: Discrete 800x600
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.033s (30.000 fps)
                        Interval: Discrete 0.050s (20.000 fps)
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.100s (10.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.200s (5.000 fps)
                        Interval: Discrete 0.200s (5.000 fps)
root@ATK-DLRK356X:/#
```

# 第二章 基于 Qt 使用 Camera

本章将带领大家用 Qt 来使用 USB 摄像头及 MIPI 摄像头。

## 2.1 使用 QCamera 类

Qt 的 QCamera 类支持 USB 摄像头及 MIPI 摄像头。我们可以直接使用这个类来开发摄像头应用。下面贴上重要部分代码。详细请看源码。开发板网盘资料-A 盘路径下的 01、程序源码\0*、Camera 开发例程\Qt，*代表 1~9。由于 RK3568 与 RK3588 或者后面的开发板路径不一样，所以用*替代。详细源码看 01_qcamera。

要使用 QCamera 类，要加在项目文件中加上 QT += core gui multimedia multimediawidgets。其中 multimediawidgets 为 QCamera 视频输出显示的模块。所以要加上#include <QVideoWidget>头文件。multimedia 为 QCamera 的媒体模块，同时包含#include <QCamera>头文件。

新建一个 QWidget 模板项目，在 Widget 的构造函数里添加如下代码。

```
this->resize(640, 480);
// 请根据各自的摄像头节点填写
m_qcamera = new QCamera("/dev/video0", this);

if (!m_qcamera) {
    qDebug() << "摄像头初始化失败！";
}

QCameraViewfinderSettings settings;
// 设置分辨率
settings.setResolution(640, 480);
m_qcamera->setViewfinderSettings(settings);

m_videoWidget = new QVideoWidget(this);
m_videoWidget->resize(this->size());

// 设置视频输出
m_qcamera->setViewfinder(m_videoWidget);
m_qcamera->start();
// 注 m_videoWidget 有可能刷新延时，可能是 Qt 底层刷新问题
// 这里用 QVideoWidget 只是用于测试，请用定时器用 update()延时更新整个界面
m_videoWidget->show();
```

笔者在 RK3568 上已经验证上面的代码，测试了 video0 与 video9。MIPI 摄像头与 USB 摄像头都可以使用，确保代码可行性！注意 RK3588 或者后面原子出的板子不一定是 video0 与 video9 请大家根据 1.1 小节按实际情况测试。

# 第三章 基于 OpenCV 使用 Camera

本章与大家使用 OpenCV 来开发摄像头应用。

本章内容是用 python/c++显示摄像头的内容。

### 3.1 摄像头 python 版本

源码路径为开发板网盘资料-A 盘 01、程序源码\08、Camera 开发例程\python\camera.py。直接看源码，没什么可解释，简单。

```
import cv2

# 打开摄像头，例 video9，写 9
cap = cv2.VideoCapture(0)

# 检查摄像头是否成功打开
if not cap.isOpened():
    print("无法打开摄像头")
    exit()

# 设置摄像头的分辨率
# 假设我们想要设置为 640x480
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

# 循环读取摄像头的帧
while True:
    # 读取一帧
    ret, frame = cap.read()

    # 检查是否成功读取帧
    if not ret:
        print("无法接收帧（流可能已结束？）")
        break

    # 显示帧
    cv2.imshow('frame', frame)

    # 如果按下'q'键，则退出循环
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# 释放摄像头并关闭所有窗口
cap.release()
cv2.destroyAllWindows()
```

拷贝到开发板上直接执行 python3 camera.py 即可。

### 3.2 摄像头 c++版本

源码路径为开发板网盘资料-A 盘 01、程序源码\08、Camera 开发例程\c++\opencv.cpp。

```cpp
#include <opencv2/opencv.hpp>
#include <iostream>

int main(int argc, char** argv)
{
    // 创建一个 VideoCapture 对象，video0,写 0
    cv::VideoCapture cap(0);

    // 检查摄像头是否成功打开
    if (!cap.isOpened()) {
        std::cerr << "Error opening video capture" << std::endl;
        return -1;
    }

    // 创建一个窗口来显示视频
    cv::namedWindow("Camera Feed", cv::WINDOW_AUTOSIZE);

    // 循环读取摄像头的帧
    while (true) {
        // 读取一帧
        cv::Mat frame;
        if (!cap.read(frame)) {
            std::cerr << "Failed to grab frame" << std::endl;
            break;
        }

        // 显示帧
        cv::imshow("Camera Feed", frame);

        // 等待 30 毫秒，如果用户在这段时间内按下了'q'键，则退出循环
        if (cv::waitKey(30) == 'q') {
            break;
        }
    }

    // 释放 VideoCapture 对象和销毁所有窗口
    cap.release();
    cv::destroyAllWindows();

    return 0;
}
```

CMakeLists.txt 如下：

```
#@author       Deng Zhimao
```

```
2   #@email        dengzhimao@alientek.com
3   #http://www.openedv.com/forum.php
4
5   cmake_minimum_required(VERSION 3.8)
6   message(STATUS "cmake version ${CMAKE_VERSION}")
7
8   #请根据各自己的开发板设置相应的路径，以 ATK-DLRK3568 为例
9   set(TOOLCHAIN_DIR /home/alientek/ATK-
DLRK3568/buildroot/output/rockchip_rk3568/host)
10  set(CMAKE_CXX_COMPILER ${TOOLCHAIN_DIR}/bin/aarch64-buildroot-linux-
gnu-g++)
11  set(CMAKE_C_COMPILER ${TOOLCHAIN_DIR}/bin/aarch64-buildroot-linux-
gnu-gcc)
12  set(SYSROOT ${TOOLCHAIN_DIR}/aarch64-buildroot-linux-
gnu/sysroot/usr/include)
13  set(CMAKE_SYSROOT ${TOOLCHAIN_DIR}/aarch64-buildroot-linux-
gnu/sysroot)
14
15  set(CMAKE_CXX_STANDARD 11)
16  add_definitions(-g -O0 -ggdb -gdwarf -funwind-tables -rdynamic)
17  add_definitions(-Wno-write-strings -Wno-return-type)
18
19  set(OPENCV_LIBS opencv_core opencv_flann opencv_videoio opencv_video
opencv_highgui opencv_imgcodecs opencv_imgproc)
20
21  include_directories(${SYSROOT})
22  include_directories(${SYSROOT}/opencv4)
23
24  project(opencv)
25  add_executable(opencv opencv.cpp)
26  target_link_libraries(opencv ${OPENCV_LIBS})
```

请先安装交叉编译器看【正点原子】基于 Buildroot 系统_交叉编译器安装与使用参考手册。然后执行 cmake。

```
cmake .  #在 CMakeLists.txt 当前目录下 cmake
make
```

编译成功后拷贝对应的可执行文件 opencv 到开发板上执行即可。

# 第四章 基于 Qt 与 OpenCV 使用 Camera

前面两章分别使用了 Qt 与 opencv 来开发摄像头应用，很多时候我们想将这两者结合起来开发。本章我们来学习如何将 OpenCV 与 Qt 结合起来开发摄像头。

本章的内容是，使用 OpenCV 读取摄像头用 Qt 显示到屏幕上。

## 4.1 Qt 项目使用 OpenCV

请参考【正点原子】基于 Buildroot 系统_OpenCV4 使用参考手册第 1.3.4 小节。

## 4.2 Qt 使用 OpenCV 读取摄像头

开发板网盘资料-A 盘路径下的 01、程序源码\0*、Camera 开发例程\Qt，*代表 1~9。由于 RK3568 与 RK3588 或者后面的开发板路径不一样，所以用*替代。详细源码看 02_qt_opencv。

由于项目文件较多，这里只关键代码部分。其他详细请阅读源码。

```
1  void CameraFrameThread::run()
2  {
3      cv::VideoCapture cap(0); // RK3568MIPI 摄像头是 0 usb 摄像头是 9。其他板子请参考
4      cap.set(cv::CAP_PROP_FRAME_WIDTH, 640);
5      cap.set(cv::CAP_PROP_FRAME_HEIGHT, 480);
6
7      if (!cap.isOpened()) {
8          return;
9      }
10
11     while (true) {
12         cv::Mat frame;
13         cap.read(frame);
14         QImage tmpImage(frame.data, frame.cols, frame.rows, QImage::Format_BGR888);
15         if(!tmpImage.isNull())
16             emit imageIsReady(tmpImage);
17     }
18     cap.release();
19 }
```

笔者这里开启了一个线程，使用 OpenCV 读取摄像头的帧。然后构建 QImage，最后在 QLabel 上显示，非常简单。

至此，摄像头应用能基本满足用户的需要了，感谢大家的支持。

# 第五章 基于 Qt 与 V4l2 使用 Camera

本章仅支持 MIPI 摄像头，因为本章获取的是 RGB 数据，许多市面上的 USB 摄像头是无法直接获取 RGB 数据的。一般 USB 摄像头都是 YUV 数据格式的，与 Qt 一起使用需要代码转 RGB 数据。本章不涉及 YUV 转 RGB，也就不写 USB 摄像头使用 V4l2 了。

本章的内容是使用 V4l2 打开摄像头取数据然后用 Qt 显示到屏幕上。

## 5.1 MIPI 摄像头使用 v4l2

以 ATK-DLRK3588 为例，执行下面的指令获取摄像头支持的格式。这里是获取第 1.1 小节中 self_path 节点支持的摄像头格式。看到下图第【8】项，支持 RGBP 格式也就是 RGB565。我们就可以直接使用 v4l2 直接取这个格式的视频流。注意，如果你对 MIPI 摄像头支持的格式或链路不了解，请你移步到 B 站观看视频详细，视频以 RV1126 为例

（摄像头格式及链路视频链接：https://www.bilibili.com/video/BV1dW4y1f7Qu?t=1.5&p=8）。

```
v4l2-ctl --list-formats-ext --device /dev/video72
```

```
root@ATK-DLRK3588:/# v4l2-ctl --list-formats-ext --device /dev/video72
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture Multiplanar

        [0]: 'UYVY' (UYVY 4:2:2)
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
        [1]: 'NV16' (Y/CbCr 4:2:2)
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
        [2]: 'NV61' (Y/CrCb 4:2:2)
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
        [3]: 'NV21' (Y/CrCb 4:2:0)
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
        [4]: 'NV12' (Y/CbCr 4:2:0)
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
        [5]: 'NM21' (Y/CrCb 4:2:0 (N-C))
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
        [6]: 'NM12' (Y/CbCr 4:2:0 (N-C))
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
        [7]: 'GREY' (8-bit Greyscale)
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
        [8]: 'RGBP' (16-bit RGB 5-6-5)
                Size: Stepwise 32x32 - 1920x2160 with step 8/8
root@ATK-DLRK3588:/#
```

源码路径：在我们的开发板网盘资料-A 盘路径下的 01、程序源码\0*、Camera 开发例程\Qt，*代表 1~9。由于 RK3568 与 RK3588 或者后面的开发板路径不一样，所以用*替代。详细源码看 03_ v4l2_camera，这里笔者不做详细代码讲解，这个是我们 IMX6ULL 开发板的 C 应用基础知识内容，还是唠叨一下 RK 系列不适合初学者入门，RK 开发板适合有点基础的用户。

源码为参考代码，不代表实际应用开发，仅供参考。

代码过长，仅贴部分代码。

cameraframethread.cpp 源码部分代码如下：

```
1   // v4l2-ctl --list-formats-ext --device /dev/video72
2   // 根据各自的开发板确认摄像头节点,必须确认节点是否支持 RGBP
3   #define VIDEO_DEV           "/dev/video72"
4   struct v4l2_format fmt = {0};
5   fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE_MPLANE;
6   ......省略许多代码
7   if (ioctl(fd, VIDIOC_G_FMT, &fmt) == -1) {
8       qDebug("ERROR: failed to VIDIOC_G_FMT");
9       close(fd);
10      return;
11  }
12
13  fmt.fmt.pix_mp.width = 640;
14  fmt.fmt.pix_mp.height = 480;
15  fmt.fmt.pix_mp.pixelformat =  V4L2_PIX_FMT_RGB565;
16  fmt.fmt.pix_mp.field = V4L2_FIELD_NONE;
```

```
17
18
19  if (ioctl(fd, VIDIOC_S_FMT, &fmt) == -1) {
20      qDebug("ERROR: failed to VIDIOC_S_FMT");
21      close(fd);
22      return;
23  }
```