页面 讨论

导航
WIKI首页
官方店铺
资料下载中心
所有页面

所有开发板
Stm32mp157
100ask_imx6ull
100ask imx6ull mini
roc-rk3399-pc
firefly-rk3288
100ask_am335x
jz2440
Tiny4412
百问网imx6ull-qemu
其它开发板

Wiki工具
特殊页面
引用本页

更多
链入页面
相关更改
可打印版本
固定链接
页面信息
页面日志

分类
有语法高亮错误的页面
含有受损文件链接的页面
Linux Operating System
Visual
V4L2

V4L2摄像头概述

框架目的

本文所要讲述的 V4L2 Linux 内核框架[1]允许控制两个相机传感器、可捕获以不同的像素格式或诸如JPEG编码流数据原始帧摄像头接口。

在其他的 Linux 多媒体框架和应用程序的支持下，通常可以使用它进行快照、进行预览、进行视频录制，甚至可以从进行远程传输图像。

系统概述

800px

组件说明

应用程序（用户空间）

任何依赖 V4L2 Linux 内核接口或 libv4l 抽象层的应用程序，GStreamer框架提供了对应的应用程序。

V4L2 实用程序（用户空间）

一组用于测试，配置和使用整个相机子系统的工具，包括外部相机传感器和相机接口。V4l2-ctl 是最有用的实用程序之一。

V4L2库（libv4l）（用户空间）

V4L2 Linux 内核接口之上的一组库，这些库对内核接口进行了抽象化，以简化操作、保持兼容性或在基于 V4L2 的应用程序和 V4L2 内核接口之间添加一些挂钩。

yavta（用户空间）

Yavta是一种测试工具，它依赖于 V4L2 Linux 的内核接口。

V4L2内核（内核空间）

该层表示标准的 Linux 内核的 V4L2 框架。

stm32-dcmi（内核空间）

V4L2 DCMI Linux 设备驱动程序处理 DCMI 硬件模块。

相机传感器X 驱动程序（内核空间）

V4L2 Linux 设备驱动程序处理摄像头传感器X 的外围设备，处理一些 GPIO 以及可能的电源，以打开/关闭摄像头传感器的电源。通过 i2c 总线与摄像机传感器进行通讯。

DCMI（硬件）

数码相机存储器接口的硬件模块。

相机传感器X（硬件）

相机传感器的外围设备。

API描述

在 Linux Media 子系统文档[2]中描述了 V4L2 用户态 API

Linux内核文档[3]的 “V4L2内核支持” 部分介绍了 V4L2 内核框架的内部API。

配置

内核配置

目录 展开 收起

STMicroelectronics 默认启用STM32相机接口和OV5640相机传感器。

但是，使用上游的内核版本时并非如此。在这种情况下，可以使用 Linux内核menuconfig工具 启用 DCMI V4L2 驱动程序：

```
[*] Device Drivers --->
    [*] Multimedia support --->
        [*] V4L platform devices  --->
            [*] STM32 Digital Camera Memory Interface (DCMI) support
```

还必须启用连接到摄像头接口的外部摄像头传感器，这是基于 OV5640 Omnivision 摄像头传感器的示例，该传感器位于 MB1379 摄像子板[4]上，该子板[4]连接到 STM32MP15 评估板[6]的 CN7 摄像头连接器[5]。：

```
[*] Device Drivers --->
    [*] Multimedia support --->
        I2C Encoders, decoders, sensors and other helper chips  --->
            [*] OmniVision OV5640 sensor support
```

设备树配置

[1] 借助 Linux 内核的设备树机制，请参阅 DCMI设备树配置 文章以获取 DCMI 和传感器配置的完整资料。

如何使用框架

可使用 V4l2-ctl, gst-launch 或 gst-play 一个实用的命令行程序启用此处描述的用例。

列出视频设备及其功能

使用 --list-devices 选项列出所有可用的视频设备：

```
Board $> v4l2-ctl --list-devices
```

```
STM32 Camera Memory Interface (platform:dcmi):
        /dev/video0
```

如果有多个设备可用，请在任何 V4l2-ctl 命令之后使用 -d 选项来定位特定设备。如果未指定 -d 选项，则默认选择 /dev/video0 。

为了获得有关特定设备的信息，请使用 -D 选项：

```
Board $> v4l2-ctl -d /dev/video0 -D
```

```
Driver Info (not using libv4l2):
        Driver name   : stm32-dcmi
        Card type     : STM32 Camera Memory Interface
        Bus info      : platform:dcmi
        Driver version: X.Y.Z
        Capabilities  : 0x85200001
                Video Capture
                Read/Write
                Streaming
                Extended Pix Format
                Device Capabilities
        Device Caps   : 0x05200001
                Video Capture
                Read/Write
                Streaming
```

```
                Extended Pix Format
```

控制相机

将 V4l2-ctl 与 -L 选项一起使用，以获取受支持的控件的列表：

```
Board $> v4l2-ctl -L
```

```
User Controls

                     contrast (int)    : min=0 max=255 step=1 default=0 value=0 flags=slider
                   saturation (int)    : min=0 max=255 step=1 default=64 value=64 flags=slider
                          hue (int)    : min=0 max=359 step=1 default=0 value=0 flags=slider
       white_balance_automatic (bool)   : default=1 value=1 flags=update
                  red_balance (int)    : min=0 max=4095 step=1 default=0 value=128 flags=inactive, slider
                 blue_balance (int)    : min=0 max=4095 step=1 default=0 value=128 flags=inactive, slider
                     exposure (int)    : min=0 max=65535 step=1 default=0 value=885 flags=inactive, volatile
               gain_automatic (bool)   : default=1 value=1 flags=update
                         gain (int)    : min=0 max=1023 step=1 default=0 value=32 flags=inactive, volatile
              horizontal_flip (bool)   : default=0 value=0
                vertical_flip (bool)   : default=0 value=0

Camera Controls

                auto_exposure (menu)   : min=0 max=1 default=0 value=0 flags=update
                                0: Auto Mode
                                1: Manual Mode

Image Processing Controls

                 test_pattern (menu)   : min=0 max=1 default=0 value=0
                                0: Disabled
                                1: Color bars
```

注意："value=" 字段返回控件的当前值

可以通过 --set-ctrl 选项更改控制值，例如：

```
Board $> v4l2-ctl --set-ctrl test_pattern=1
```

控制值可以动态更改。在以下示例中，在运行预览时启用/禁用颜色栏：

在后台开始预览

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=1280, Height=720, framerate=(fraction)15/1" ! queue ! autovideosink -e &
```

然后交替激活颜色栏

```
Board $> v4l2-ctl --set-ctrl test_pattern=1;sleep 1;v4l2-ctl --set-ctrl test_pattern=0;sleep 1;v4l2-ctl --set-ctrl test_pattern=1;sleep
1;v4l2-ctl --set-ctrl test_pattern=0;killall gst-launch-1.0
```

预览运行时还可以更改水平/垂直翻转：

```
Board $> v4l2-ctl --set-ctrl horizontal_flip=1;sleep 2;v4l2-ctl --set-ctrl horizontal_flip=0;sleep 2;v4l2-ctl --set-ctrl
vertical_flip=1;sleep 2;v4l2-ctl --set-ctrl vertical_flip=0;killall gst-launch-1.0
```

设置像素格式，分辨率和帧率

使用 --list-formats-ext 选项可获取受支持的像素格式、分辨率和帧速率：

```
Board $> v4l2-ctl --list-formats-ext
```

```
ioctl: VIDIOC_ENUM_FMT
        Index       : 0
        Type        : Video Capture
        Pixel Format: 'JPEG' (compressed)
        Name        : JFIF JPEG
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 2592x1944
                        Interval: Discrete 0.067s (15.000 fps)

        Index       : 1
        Type        : Video Capture
        Pixel Format: 'UYVY'
        Name        : UYVY 4:2:2
                Size: Discrete 176x144
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 640x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x480
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 720x576
                        Interval: Discrete 0.067s (15.000 fps)
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1024x768
                        Interval: Discrete 0.067s (15.000 fps)
```

```
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.067s (15.000 fps)

 Index       : 2
 Type        : Video Capture
 Pixel Format: 'YUYV'
 Name        : YUYV 4:2:2
        Size: Discrete 176x144
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 320x240
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 640x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x576
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1024x768
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.067s (15.000 fps)

 Index       : 3
 Type        : Video Capture
 Pixel Format: 'RGBP'
 Name        : 16-bit RGB 5-6-5
        Size: Discrete 176x144
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 320x240
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 640x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x480
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 720x576
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1024x768
```

```
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1280x720
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 1920x1080
                Interval: Discrete 0.067s (15.000 fps)
                Interval: Discrete 0.033s (30.000 fps)
        Size: Discrete 2592x1944
                Interval: Discrete 0.067s (15.000 fps)
```

为了更改摄像机的配置，请首先使用 --set-parm 选项选择帧速率：

```
Board $> v4l2-ctl --set-parm=30
```

然后使用 --set-fmt-video 选项选择所需的分辨率和像素格式：

```
Board $> v4l2-ctl --set-fmt-video=width=320,height=240,pixelformat=RGBP
```

设置帧率

对于 V4l2-ctl，请使用 --set-parm 选项仅给出的帧速率分子，分母固定为1（仅允许整数帧速率值）：

```
Board $> v4l2-ctl --set-parm=<framerate numerator>
```

以 30fps 的速度拍摄 100 张 VGA 图片：

```
Board $> v4l2-ctl --set-parm=30;v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=JPEG --stream-mmap --stream-count=100 --stream-to=pics@30fps.jpeg
```

使用 gst-play 以 30fps 的速度重播：

```
Board $> gst-play-1.0 pics@30fps.jpeg --videosink="videorate ! video/x-raw, framerate=(fraction)30/1 ! autovideosink"
```

以 15fps 的速度拍摄 100 张 VGA 图片：

```
Board $> v4l2-ctl --set-parm=15;v4l2-ctl --set-fmt-video=width=640,height=480,pixelformat=JPEG --stream-mmap --stream-count=100 --stream-to=pics@15fps.jpeg
```

使用 gst-play 以 15fps 的速度重播：

```
Board $> gst-play-1.0 pics@15fps.jpeg --videosink="videorate ! video/x-raw, framerate=(fraction)15/1 ! autovideosink"
```

对于 GStreamer，使用帧速率的上限：

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, ... framerate=(fraction)<numerator>/<denominator>" ! ...
```

预览 VGA@30fps

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=640, Height=480, framerate=(fraction)30/1" ! queue ! autovideosink -e
```

预览 VGA@15fps

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=640, Height=480, framerate=(fraction)15/1" ! queue ! autovideosink -e
```

抓取原始帧

在磁盘上捕获 QVGA RGB565 原始帧：

```
Board $> v4l2-ctl --set-fmt-video=width=320,height=240,pixelformat=RGBP --stream-mmap --stream-count=1 --stream-to=grab-320x240-rgb565.raw
```

必须首先将此原始帧转换为 JPEG 然后才能将其显示出来，：

```
Board $> gst-launch-1.0 filesrc location= grab-320x240-rgb565.raw blocksize=153600 ! "video/x-raw, format=(string)RGB16, width=(int)320, height=(int)240, framerate=(fraction)30/1" ! videoconvert ! jpegenc ! filesink location=grab-320x240-rgb565.jpeg
```

之后，可以使用 weston-image 命令显示 JPEG 文件：

```
Board $> weston-image grab-320x240-rgb565.jpeg
```

全屏预览

```
Board $> gst-launch-1.0 v4l2src ! "video/x-raw, width=1280, Height=720, framerate=(fraction)15/1" ! queue ! autovideosink -e
```

> 注意：请注意，GStreamer会覆盖以前可能在视频设备上设置的所有参数（例如，通过 V4l2-ctl 命令设置的参数，例如分辨率，像素格式，帧速率等）

拍照

捕获 5Mp JPEG：

```
Board $> v4l2-ctl --set-parm=15; v4l2-ctl --set-fmt-video=width=2592,height=1944,pixelformat=JPEG --stream-mmap --stream-count=1 --stream-skip=3 --stream-to=pic-5Mp.jpeg; v4l2-ctl --set-parm=30
```

然后显示它：

```
Board $> weston-image pic-5Mp.jpeg
```

现在可以使用 gst-typefind 命令检查图片分辨率：

```
Board $> gst-typefind-1.0 pic-5Mp.jpeg
```

```
pic-5Mp.jpeg - image/jpeg, width=(int)2592, height=(int)1944, sof-marker=(int)0
```

0

通过网络传输图片

请参阅 如何通过网络传输图片文件 文章，以获取有关如何通过网络流式传输摄像头内容的一些示例参考。

如何跟踪和调试

如何监视

检查 devicetree 配置

下面的命令用于检查是否已启用 DCMI，检查使用了哪个摄像头传感器并记录有关其 devicetree 设置的其他详细信息：

```
rm devicetree.txt
echo "[devicetree]" >> devicetree.txt
echo "|-[dcmi]" >> devicetree.txt
find /proc/device-tree/soc | grep dcmi | sed 's/\/proc\/device-tree\/soc\//| |-/' >> devicetree.txt
echo "|" >> devicetree.txt
echo "|-[camera:" | tr -d "\n" >> devicetree.txt
cat /proc/device-tree/soc/i2c*/camera*/compatible >> devicetree.txt
echo "]" >> devicetree.txt
find /proc/device-tree/soc | grep camera | sed 's/\/proc\/device-tree\/soc\//  |-/' >> devicetree.txt
echo "" >> devicetree.txt
cat devicetree.txt
```

```
[devicetree]
|-[dcmi]
| |-dcmi@4c006000
| |-dcmi@4c006000/compatible
| |-dcmi@4c006000/clocks
| |-dcmi@4c006000/resets
| |-dcmi@4c006000/pinctrl-1
| |-dcmi@4c006000/port
| |-dcmi@4c006000/port/endpoint
| |-dcmi@4c006000/port/endpoint/hsync-active
| |-dcmi@4c006000/port/endpoint/vsync-active
| |-dcmi@4c006000/port/endpoint/remote-endpoint
| |-dcmi@4c006000/port/endpoint/bus-width
| |-dcmi@4c006000/port/endpoint/pclk-sample
| |-dcmi@4c006000/port/endpoint/phandle
| |-dcmi@4c006000/port/endpoint/linux,phandle
| |-dcmi@4c006000/port/endpoint/name
| |-dcmi@4c006000/port/name
| |-dcmi@4c006000/clock-names
| |-dcmi@4c006000/status
| |-dcmi@4c006000/interrupts
| |-dcmi@4c006000/dma-names
| |-dcmi@4c006000/phandle
| |-dcmi@4c006000/reg
| |-dcmi@4c006000/pinctrl-0
| |-dcmi@4c006000/dmas
| |-dcmi@4c006000/linux,phandle
| |-dcmi@4c006000/name
| |-dcmi@4c006000/pinctrl-names
| |-pin-controller/dcmi-sleep@0
| |-pin-controller/dcmi-sleep@0/pins
| |-pin-controller/dcmi-sleep@0/pins/pinmux
| |-pin-controller/dcmi-sleep@0/pins/name
| |-pin-controller/dcmi-sleep@0/phandle
| |-pin-controller/dcmi-sleep@0/linux,phandle
| |-pin-controller/dcmi-sleep@0/name
| |-pin-controller/dcmi@0
| |-pin-controller/dcmi@0/pins
| |-pin-controller/dcmi@0/pins/pinmux
| |-pin-controller/dcmi@0/pins/bias-disable
| |-pin-controller/dcmi@0/pins/name
| |-pin-controller/dcmi@0/phandle
| |-pin-controller/dcmi@0/linux,phandle
| |-pin-controller/dcmi@0/name
```

```
   |
|-[camera:ovti,ov5640]
  |-i2c@40013000/camera@3c
  |-i2c@40013000/camera@3c/compatible
  |-i2c@40013000/camera@3c/powerdown-gpios
  |-i2c@40013000/camera@3c/DOVDD-supply
  |-i2c@40013000/camera@3c/clocks
  |-i2c@40013000/camera@3c/rotation
  |-i2c@40013000/camera@3c/port
  |-i2c@40013000/camera@3c/port/endpoint
  |-i2c@40013000/camera@3c/port/endpoint/hsync-active
  |-i2c@40013000/camera@3c/port/endpoint/vsync-active
  |-i2c@40013000/camera@3c/port/endpoint/remote-endpoint
  |-i2c@40013000/camera@3c/port/endpoint/bus-width
  |-i2c@40013000/camera@3c/port/endpoint/pclk-sample
  |-i2c@40013000/camera@3c/port/endpoint/phandle
  |-i2c@40013000/camera@3c/port/endpoint/data-shift
  |-i2c@40013000/camera@3c/port/endpoint/linux,phandle
  |-i2c@40013000/camera@3c/port/endpoint/name
  |-i2c@40013000/camera@3c/port/name
  |-i2c@40013000/camera@3c/clock-names
  |-i2c@40013000/camera@3c/status
  |-i2c@40013000/camera@3c/reset-gpios
  |-i2c@40013000/camera@3c/phandle
  |-i2c@40013000/camera@3c/reg
  |-i2c@40013000/camera@3c/linux,phandle
  |-i2c@40013000/camera@3c/name
```

如何追踪

V4L2 用户界面 API 的工作过程

可以使用下面的命令启用对 V4L2 用户态 API [2]的跟踪:

```
Board $> echo 0x3 >  /sys/devices/platform/soc/*.dcmi/video4linux/video*/dev_debug
```

跟踪在内核日志缓冲区中的输出:

```
Board $> dmesg
```

```
[10130.641469] video0: VIDIOC_TRY_FMT: type=vid-cap, width=640, height=480, pixelformat=YUYV, field=none, bytesperline=
1280, sizeimage=614400, colorspace=8, flags=0x0, ycbcr_enc=1, quantization=1, xfer_func=2
[10130.641550] video0: VIDIOC_S_FMT: type=vid-cap, width=640, height=480, pixelformat=YUYV, field=none, bytesperline=12
80, sizeimage=614400, colorspace=8, flags=0x0, ycbcr_enc=1, quantization=1, xfer_func=2
[10130.641597] video0: VIDIOC_G_PARM: type=vid-cap, capability=0x1000, capturemode=0x0, timeperframe=1/30, extendedmode
=0, readbuffers=2
[10130.641638] video0: VIDIOC_G_PARM: type=vid-cap, capability=0x1000, capturemode=0x0, timeperframe=1/30, extendedmode
=0, readbuffers=2
[10130.641681] video0: VIDIOC_S_PARM: type=vid-cap, capability=0x1000, capturemode=0x0, timeperframe=1/30, extendedmode
=0, readbuffers=2
[10130.641740] video0: VIDIOC_G_CTRL: error -22: id=0x980927, value=0
[10130.642770] video0: VIDIOC_REQBUFS: count=0, type=vid-cap, memory=mmap
[10130.642819] video0: VIDIOC_CREATE_BUFS: index=0, count=0, memory=mmap, type=vid-cap, width=640, height=480, pixelfor
mat=YUYV, field=none, bytesperline=1280, sizeimage=614400, colorspace=8, flags=0x0, ycbcr_enc=1, quantization=1, xfer_f
u n c = 2
[10130.658541] video0: VIDIOC_G_CTRL: error -22: id=0x980927, value=0
[10130.662770] video0: VIDIOC_REQBUFS: count=3, type=vid-cap, memory=mmap
[10130.662852] video0: VIDIOC_QUERYBUF: 00:00:00.00000000 index=0, type=vid-cap, flags=0x00002000, field=any, sequence=
```

```
0, memory=mmap, bytesused=0, offset/userptr=0x0, length=614400
[10130.662892] timecode=00:00:00 type=0, flags=0x00000000, frames=0, userbits=0x00000000
[10130.662917] video0: VIDIOC_QUERYBUF: 00:00:00.00000000 index=1, type=vid-cap, flags=0x00002000, field=any, sequence=
0, memory=mmap, bytesused=0, offset/userptr=0x96000, length=614400
[10130.662952] timecode=00:00:00 type=0, flags=0x00000000, frames=0, userbits=0x00000000
[10130.662967] video0: VIDIOC_QUERYBUF: 00:00:00.00000000 index=2, type=vid-cap, flags=0x00002000, field=any, sequence=
0, memory=mmap, bytesused=0, offset/userptr=0x12c000, length=614400
[10130.663002] timecode=00:00:00 type=0, flags=0x00000000, frames=0, userbits=0x00000000
[10130.666880] video0: VIDIOC_STREAMON: type=vid-cap
[10130.857484] video0: VIDIOC_CREATE_BUFS: index=3, count=1, memory=mmap, type=vid-cap, width=640, height=480, pixelfor
mat=YUYV, field=none, bytesperline=1280, sizeimage=614400, colorspace=8, flags=0x0, ycbcr_enc=1, quantization=1, xfer_f
 u n c = 2
[10130.857585] video0: VIDIOC_QUERYBUF: 00:00:00.00000000 index=3, type=vid-cap, flags=0x00002000, field=any, sequence=
0, memory=mmap, bytesused=0, offset/userptr=0x1c2000, length=614400
[10130.857627] timecode=00:00:00 type=0, flags=0x00000000, frames=0, userbits=0x00000000
[10131.022069] video0: VIDIOC_STREAMOFF: type=vid-cap
```

V4L2核心框架跟踪

可以使用以下命令来启用 V4L2 核心框架[3]的跟踪：

```
Board $> echo 0x3 > /sys/module/videobuf2_core/parameters/debug
Board $> echo 0x3 > /sys/module/videobuf2_v4l2/parameters/debug
```

跟踪在内核日志缓冲区中的输出：

```
Board $> dmesg
```

```
[11875.487933] vb2-core: __setup_offsets: buffer 0, plane 0 offset 0x00000000
[11875.501731] vb2-core: __setup_offsets: buffer 1, plane 0 offset 0x001fb000
[11875.514901] vb2-core: __setup_offsets: buffer 2, plane 0 offset 0x003f6000
[11875.532298] vb2-core: __setup_offsets: buffer 3, plane 0 offset 0x005f1000
[11875.540019] vb2-core: __vb2_queue_alloc: allocated 4 buffers, 1 plane(s) each
[11875.563689] vb2_dc_mmap: mapped dma addr 0xf1900000 at 0xb4f05000, size 2076672
[11875.571174] vb2-core: vb2_mmap: buffer 0, plane 0 successfully mapped
[11875.589656] vb2-core: vb2_core_qbuf: qbuf of buffer 0 succeeded
[11875.595684] vb2_dc_mmap: mapped dma addr 0xf1b00000 at 0xb4d0a000, size 2076672
[11875.603062] vb2-core: vb2_mmap: buffer 1, plane 0 successfully mapped
[11875.609668] vb2-core: vb2_core_qbuf: qbuf of buffer 1 succeeded
[11875.615642] vb2_dc_mmap: mapped dma addr 0xf1d00000 at 0xb4b0f000, size 2076672
[11875.623016] vb2-core: vb2_mmap: buffer 2, plane 0 successfully mapped
[11875.629628] vb2-core: vb2_core_qbuf: qbuf of buffer 2 succeeded
[11875.635617] vb2_dc_mmap: mapped dma addr 0xf1f00000 at 0xb4914000, size 2076672
[11875.642952] vb2-core: vb2_mmap: buffer 3, plane 0 successfully mapped
[11875.649715] vb2-core: vb2_core_qbuf: qbuf of buffer 3 succeeded
[11875.734058] vb2-core: vb2_core_streamon: successful
[11875.961291] vb2-core: vb2_buffer_done: done processing on buffer 0, state: 6
[11875.967036] vb2-core: vb2_core_dqbuf: returning done buffer
[11875.972437] vb2-core: vb2_core_dqbuf: dqbuf of buffer 0, with state 0
[11876.094639] vb2-core: vb2_buffer_done: done processing on buffer 1, state: 6
[11876.100367] vb2-core: vb2_core_dqbuf: returning done buffer
[11876.105788] vb2-core: vb2_core_dqbuf: dqbuf of buffer 1, with state 0
```

DCMI V4L2 内核驱动程序跟踪

可以使用下面的命令启用 DCMI 的动态调试跟踪[7]：

```
Board $> echo "module stm32_dcmi +p" > /sys/kernel/debug/dynamic_debug/control
```

这是一个 5Mp jpeg 捕获的示例：

```
Board $> gst-launch-1.0 v4l2src ! image/jpeg, width=2592, height=1944 ! fakesink
Board $> dmesg
```

```
[12706.715949] stm32-dcmi 4c006000.dcmi: Sensor format set to 0x4001 2592x1944
[12706.721548] stm32-dcmi 4c006000.dcmi: Buffer format set to JPEG 2592x1944
[12707.365947] stm32-dcmi 4c006000.dcmi: Sensor format set to 0x4001 2592x1944
[12707.371551] stm32-dcmi 4c006000.dcmi: Buffer format set to JPEG 2592x1944
[12707.437537] stm32-dcmi 4c006000.dcmi: Sensor format set to 0x4001 2592x1944
[12707.443042] stm32-dcmi 4c006000.dcmi: Buffer format set to JPEG 2592x1944
[12707.459767] stm32-dcmi 4c006000.dcmi: Setup queue, count=4, size=5038848
[12707.518914] stm32-dcmi 4c006000.dcmi: buffer[0] phy=0xf1900000 size=5038848
[12707.526068] stm32-dcmi 4c006000.dcmi: buffer[1] phy=0xf1e00000 size=5038848
[12707.533299] stm32-dcmi 4c006000.dcmi: buffer[2] phy=0xf2300000 size=5038848
[12707.541456] stm32-dcmi 4c006000.dcmi: buffer[3] phy=0xf2800000 size=5038848
[12707.551443] stm32-dcmi 4c006000.dcmi: Start streaming, starting capture
[12707.820885] stm32-dcmi 4c006000.dcmi: buffer[0] done seq=0, bytesused=499936
[12708.087436] stm32-dcmi 4c006000.dcmi: buffer[1] done seq=1, bytesused=447472
[12708.306415] stm32-dcmi 4c006000.dcmi: Stop streaming, errors=0 (overrun=0), buffers=2
[12708.319095] stm32-dcmi 4c006000.dcmi: Start streaming, starting capture
[12708.333571] stm32-dcmi 4c006000.dcmi: Stop streaming, errors=0 (overrun=0), buffers=0
```

如何调试

错误

在内核日志中无条件地跟踪错误：

```
 Board $> dmesg
[   87.233672] stm32-dcmi 4c006000.dcmi: Some errors found while streaming: errors=1 (overrun=1), buffers=24
```

内存跟踪

帧需要大块连续内存，它们由 V4L2 框架通过 DMA 分配。可以使用以下方法跟踪这些分配：

```
Board $> echo "module dma_contiguous +p" > /sys/kernel/debug/dynamic_debug/control
Board $> echo "module videobuf2_dma_contig +p" > /sys/kernel/debug/dynamic_debug/control
```

VGA 预览后的轨迹

```
[11311.617688] vb2_dc_mmap: mapped dma addr 0xf1900000 at 0xb3b6a000, size 614400
[11311.617986] vb2_dc_mmap: mapped dma addr 0xf1a00000 at 0xb3ad4000, size 614400
[11311.618071] vb2_dc_mmap: mapped dma addr 0xf1b00000 at 0xb3a3e000, size 614400
[11311.764146] vb2_dc_mmap: mapped dma addr 0xf1c00000 at 0xb307c000, size 614400
```

4 帧VGA YUV422帧：640x480x2=614400字节

源代码位置

用户空间

- V4L2 实用程序源代码

- V4L2 库源代码
- Yavta 源代码

内核空间

- V4L2 核心源代码
- stm32-dcmi V4L2 驱动程序源代码
- i2c 相机传感器 V4L2 驱动程序源代码

参考

1. 维基百科上有关 V4L2 Linux 内核框架的信息
2. Linux Media基础结构用户空间 API >> 第一部分-Linux API视频
3. 媒体子系统内核内部 API >> 1. Video4Linux设备
4. MB1379相机子板
5. STM32MP157C-EV1评估板 CN7 摄像机传感器连接器
6. STM32MP15 评估板
7. 如何使用内核动态调试