以下是基于rk3188 sdk所编写的。

1. **源码目录**

Kernel:

drivers/media/video:

|

|\_\_ rk30\_camera\_oneframe.c VIP/CIF 驱动，单帧模式，与camera host平台设备匹配

|

|\_\_ rk30\_camera\_pingpong.c rk文档说暂不支持

|

|\_\_ generic\_sensor.c generic\_sensorc.h rk抽象sensor的代码共用部分，以宏的方式实现

|

|\_\_ ov2659.c ov5640.c ov5642.c ov公司camera驱动

|

|\_\_ gc0307.c gc0308.c gc公司的camera驱动

|

|\_\_ rk30\_camera.c camera host及sensor平台设备注册相关代码，include

| plat-rk/rk-camera.c

|

|\_\_ soc\_camera.c 注册sensor平台设备驱动和一个ic\_drv驱动

|

arch/arm/mach-rk3188/

|

|\_\_ board-rk3188-sdk.c include mach-rk30/board-rk3168-tb-camera.c

arch/arm/mach-rk30

|

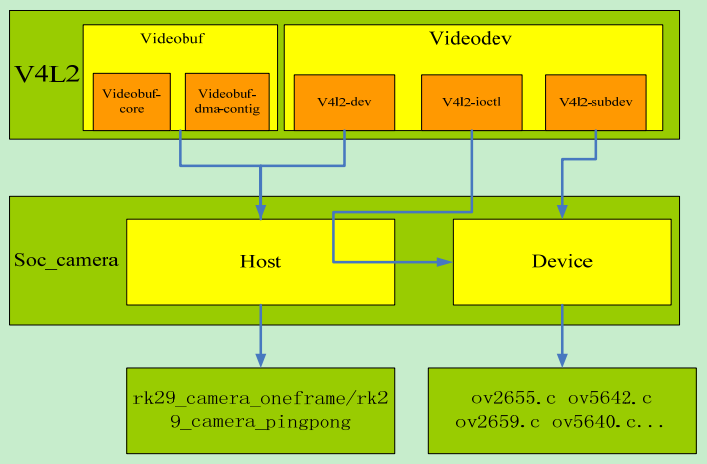
|\_\_ board-rk3168-tb-camera.c 定义各种sensor宏，include rk30\_camera.c

arch/arm/plat-rk/

|

|\_\_ rk-camera.c 主要是定义了rk\_camera\_platform

1. **驱动框架图**

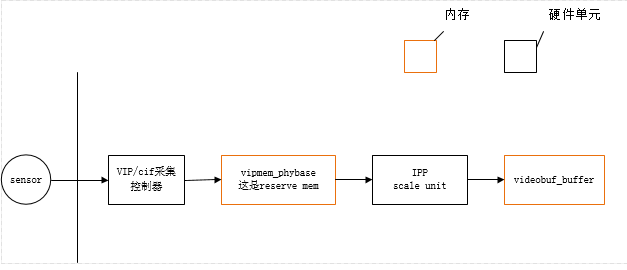


Host：struct soc\_camera\_host；

Device: struct soc\_camera\_device；

1. **图像采集流程**

流程图如下



1. 采集控制器把采集到的一帧图像放到vipmem\_phybase的一段内存空间;
2. IPP图像缩放单元从vipmem\_phybase拿一帧图像，进行缩放处理；
3. IPP处理图像完后，把图像放到videobuf\_buffer内存；
4. 代码处理流程

* IOCTL VIDIOC\_REQBUFS

为videobuf\_buffer申请空间。

1. q->ops->buf\_setup(q, &count, &size);

q为videobuf\_queue，buf\_setup为rk\_videobuf\_setup函数，rk\_videobuf\_setup申请n（n为申请videobuf\_buffer的数量，由用户决定）个rk\_camera\_work，然后把rk\_camea\_work添加到pcdev->camera\_work\_queue链表中，下面会用到该链表。

1. \_\_videobuf\_mmap\_setup (q, count, size, req->memory);

设置videobuf\_buffer的state为VIDEOBUF\_NEEDS\_INIT。

* VIDIOC\_QBUF流程

QBUF流程把videobuf\_buffer挂到采集链表，之后，IPP处理完的时候，把图像放到该buffer。

1. q->ops->buf\_prepare(q, buf, field);

设置videobuf\_buffer的state为VIDEOBUF\_PREPARED状态。

1. list\_add\_tail(&buf->stream, &q->stream);

buf为videobuf\_buffer，q为videobuf\_queue，把buf放到q->stream链表尾。

* IOCTL VIDIOC\_STREAMON

表示开始采集。调用rk\_videobuf\_queue。



如果videobuf\_buffer的state为VIDEOBUF\_PREPARED，则调用buf\_queue回调函数，上一步已经把state设置为VIDEOBUF\_PREPARED。

1. q->ops->buf\_queue(q, buf);

vb->state = VIDEOBUF\_QUEUED;

设置videobuf\_buffer的状态为VIDEOBUF\_QUEUED，表示已经放到链表中。

list\_add\_tail(&vb->queue, &pcdev->capture)；

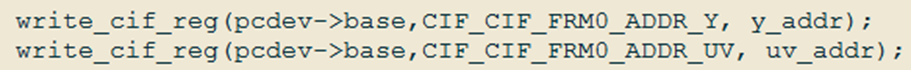
vb为上面的buf，pcdev为rk\_camera\_dev（包含soc\_camera\_host的一个结构体），把buf放到pcdev->capture链表后面，表示准备采集。

rk\_videobuf\_capture(vb, pcdev)；

这里主要负责初始化采集控制器IPP采集图像的目的地址，目的地址的计算方法如下图代码所示：



vipmem\_bsize为一帧图像的大小，vb->i表示第几个buffer，这样就可以算出存放Y的地址y\_addr和存放uv的地址uv\_addr。接着把地址写到寄存器，如下：



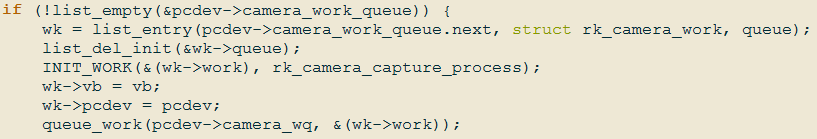
1. ici->ops->s\_stream(icd, 1);



往采集控制器控制寄存器写值，启动采集。

* 采集控制器中断处理函数

采集控制器的中断处理函数为rk\_camera\_dmairq。



前面VIDIOC\_REQBUFS的时候，已经把rk\_camera\_work添加到camear\_work\_queue链表，所以list\_empty返回0，表示不为空。

INIT\_WORK初始化wk->work的处理函数为rk\_camera\_capture\_process，然后，queue\_work把wk->work添加到pcdev->camera\_wq工作队列，工作队列开始处理工作队列中的work。

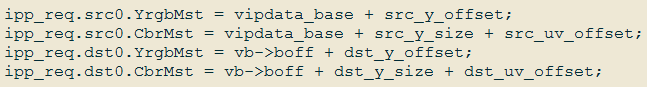
rk\_camera\_capture\_process



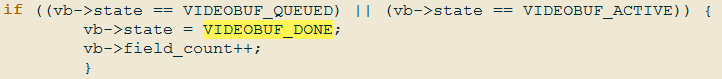
因为这里采用的是IPP单元进行缩放，所以，scale\_crop\_cb为rk\_camera\_scale\_crop\_ipp函数。

rk\_camera\_scale\_crop\_ipp

该函数的主要工作为初始化struct rk29\_ipp\_req ipp\_req，ipp\_req表示ipp unit的寄存器信息，包括图像源地址和目的地址等，初始化图像buffer的地址如下所示：



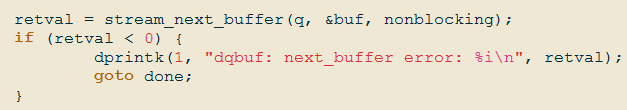
初始化完后，调用ipp\_blit\_sync(&ipp\_req)，开始ipp unit的工作，等到ipp处理完，该函数才返回。



继续回到rk\_camera\_capture\_process函数，rk\_camera\_scale\_crop\_ipp返回后，判断vb->state是否为VIDEOBUF\_QUEUED状态，前面已经把videobuf\_buffer的state设置为VIDEOBUF\_QUEUED，所以，在这，设置videobuf\_buffer的state为VIDEOBUF\_DONE状态。

* IOCTL VIDIOC\_DQBUF流程

DQBUF流程表示把采集好的buffer从链表中取出来。



stream\_next\_buffer查询videobuf\_buffer的状态是否为VIDEOBUF\_DONE的状态，如果不是，返回-EAGAIN或者错误，返回-EAGAIN表示图像还没有准备好，待会才查询一次；如果是，返回0，表示图像已经采集完成，下一步，用户空间就可以获取图像，进行下一步处理。



把videobu\_buffer从strem链表中删除，然后，设置state为VIDEOBUF\_IDLE状态，以便下一次继续使用该buf。