android camera(三)：camera V4L2 FIMC - CLK - 博客频道

前面两篇说的有点多了，不过多了解点东西也挺好的，遇到问题时可以有更多的思路，真正驱动是从这一块开始。一般BSP的camera都是完好的，我们只用关心驱动这些就可以了。

1. V4L21）简介

在Linux中，摄像头方面的标准化程度比较高，这个标准就是V4L2驱动程序，这也是业界比较公认的方式。

V4L全称是Video for Linux，是Linux内核中标准的关于视频驱动程序，目前使用比较多的版本是Video for Linux 2，简称V4L2。它为Linux下的视频驱动提供了统一的接口，使得应用程序可以使用统一的API操作不同的视频设备。从内核空间到用户空间，主要的数据流和控制类均由V4L2驱动程序的框架来定义。

V4L2驱动程序一般只提供Video数据的获得，而如何实现视频预览，如何向上层发送数据，如何把纯视频流和取景器、视频录制等实际业务组织起来，都是camera的硬件抽象层需要负责的工作。

V4L2驱动核心实现为如下文件：drivers/media/video/v4l2-dev.c。

V4l2-dev.h中定义的video_device是V4L2驱动程序的核心数据结构，它为具体的摄像头sensor驱动提供了接口调用。

V4l2的采集过程（应用程序）：

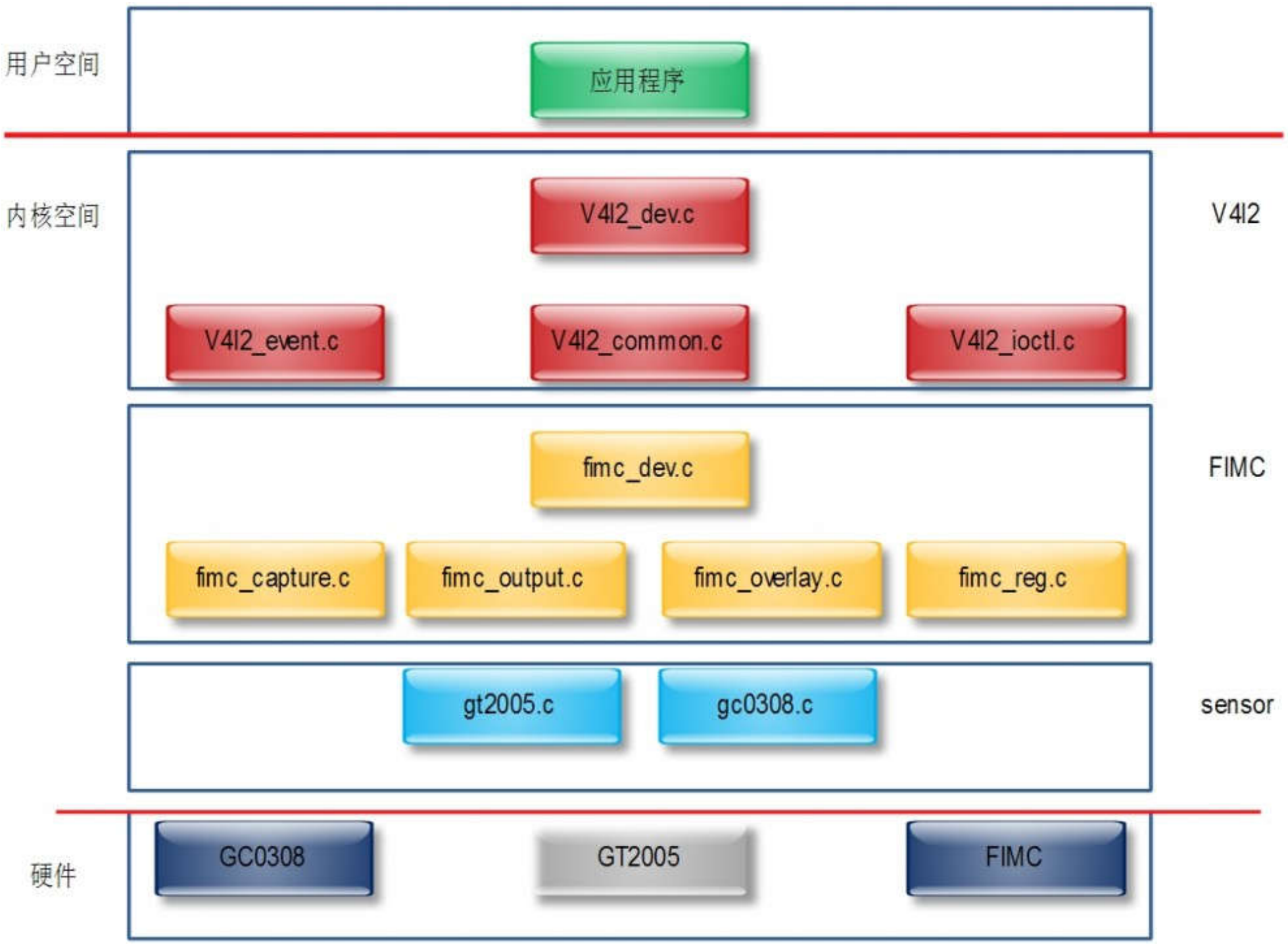
1) 打开设备，获得文件描述符；

2) 设置图片格式；

3) 分配缓冲区；

4) 启动采集过程，读取数据；

5) 停止采集，关闭设备。



2）数据结构

V4L2的主要数据结构是video_device，定义在v4l2_dev.h中：

```
1. struct  video_device
2. {
3.         /*  device  ops  */
4.         const  struct  v4l2_file_operations  *fops;    /*接口函数指针*/
5.         /*  sysfs  */
6.         struct  device  dev;           /*  v4l  设备结构  */
7.         struct  cdev  *cdev;           /*  字符设备结构*/
8.         /*  Set  either  parent  or  v4l2_dev  if  your  driver  uses  v4l2_device  */
9.         struct  device  *parent;           /*  设备父指针  */
10.        struct  v4l2_device  *v4l2_dev;      /*  v4l2设备指针*/
11.        /*  device  info  */
12.        char  name[32];    /*设备名称*/
13.        int  vfl_type;
14.        /*  'minor'  is  set  to  -1  if  the  registration  failed  */
15.        int  minor;        /*次设备号*/
16.        u16  num;
17.        /*  use  bitops  to  set/clear/test  flags  */
18.        unsigned  long  flags;
19.        /*  attribute  to  differentiate  multiple  indices  on  one  physical  device  */
20.        int  index;
21.        /*  V4L2  file  handles  */
22.        spinlock_t           fh_lock;  /*  Lock  for  all  v4l2_fhs  */
23.        struct  list_head        fh_list;  /*  List  of  struct  v4l2_fh  */
24.        int  debug;                  /*  debug  级别*/
25.        /*  Video  标准变量  */
26.        v4l2_std_id  tvnorms;               /*  Supported  tv  norms  */
27.        v4l2_std_id  current_norm;      /*  Current  tvnorm  */
28.        /*  回调函数  */
29.        void  (*release)(struct  video_device  *vdev);
30.        /*  ioctl  回调函数  */
31.        const  struct  v4l2_ioctl_ops  *ioctl_ops;
32. };
```

主要接口函数有：

intvideo_register_device(struct video_device *vdev, int type, int nr);

static intv4l2_ioctl(struct inode *inode, struct file *filp,   unsigned int cmd, unsigned long arg);

2. FIMC 1）简介

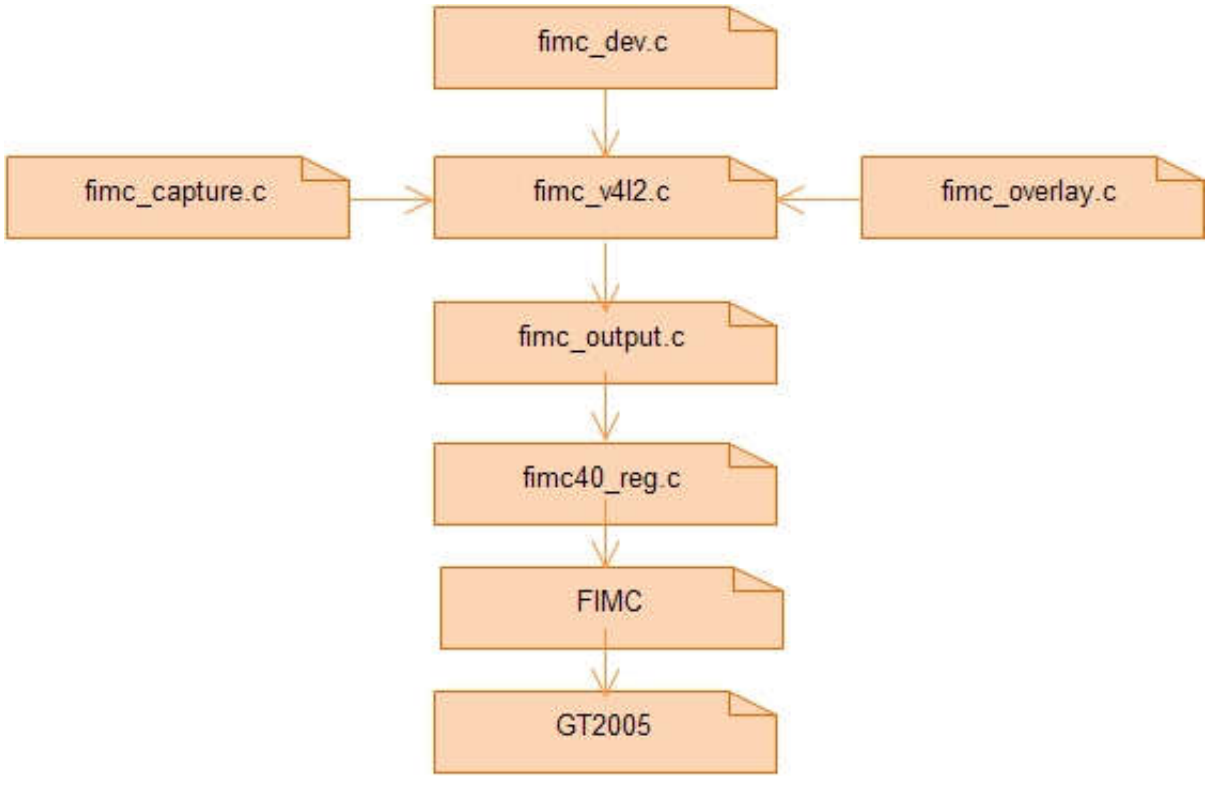
FIMC这个模块不仅仅是一个摄像头的控制接口，它还承担着V4L2的output功能和overlay的功能。

FIMC的驱动在内核中的位置：drivers/media/video/samsung/fimc

它包含下边的文件：

fimc_regs.c
fimc_capture.c
fimc_dev.c
fimc_output.c
fimc_overlay.c
fimc_v4l2.c

它们的组织关系如下：



可以看到，FIMC的驱动实现了v4l2所有的接口，可以分为v4l2-input设备接口，v4l2-output设备接口以及v4l2-overlay设备接口。这里我们主要关注v4l2-input设备接口，因为摄像头属于视频输入设备。

fimc_v4l2.c里面注册了很多的回调函数，都是用于实现v4l2的标准接口的，但是这些回调函数基本上都不是在fimc_v4l2.c里面实现的，而是有相应的.c分别去实现。比如：

v4l2-input设备的操作实现：fimc_capture.c
v4l2-output设备的操作实现： fimc_output.c
v4l2-overlay设备的操作实现： fimc_overlay.c

这些代码其实都是和具体硬件操作无关的，这个驱动把所有操作硬件寄存器的代码都写到一个文件里面了，就是fimc40_regs.c。这样把硬件相关的代码和硬件无关的代码分开来实现是非常好的方式，可以最大限度的实现代码复用。

2） 数据结构

FIMC的主要数据结构fimc_control，定义在fimc.h中：