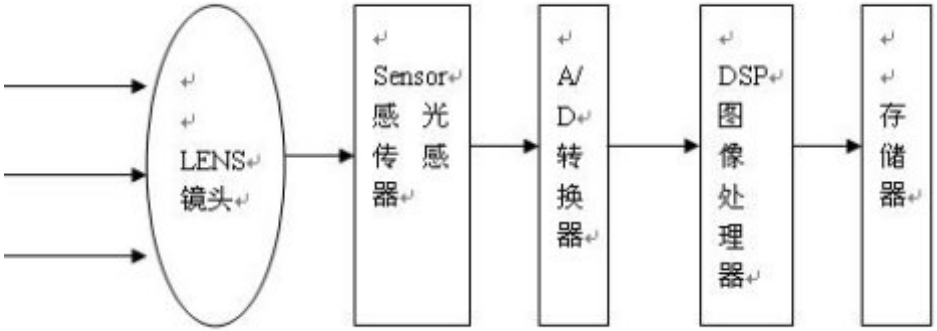
摄像头基础介绍

时间：2015-03-09 17:40:54　阅读：3385　评论：0　收藏：0　[点我收藏+]

标签：

一、摄像头结构和工作原理.



拍摄景物通过镜头，将生成的光学图像投射到传感器上，然后光学图像被转换成电信号，电信号再经过模数转换变为数字信号，数字信号经过 DSP 加工处理，再被送到电脑中进行处理，最终转换成手机屏幕上能够看到的图像。

数字信号处理芯片 DSP(DIGITAL SIGNAL PROCESSING)功能：主要是通过一系列复杂的数学算法运算，对数字图像信号参数进行优化处理，并把处理后的信号通过 USB 等接口传到 PC 等设备。DSP 结构框架:

1. ISP(image signal processor)(镜像信号处理器)

2. JPEG encoder(JPEG 图像解码器)

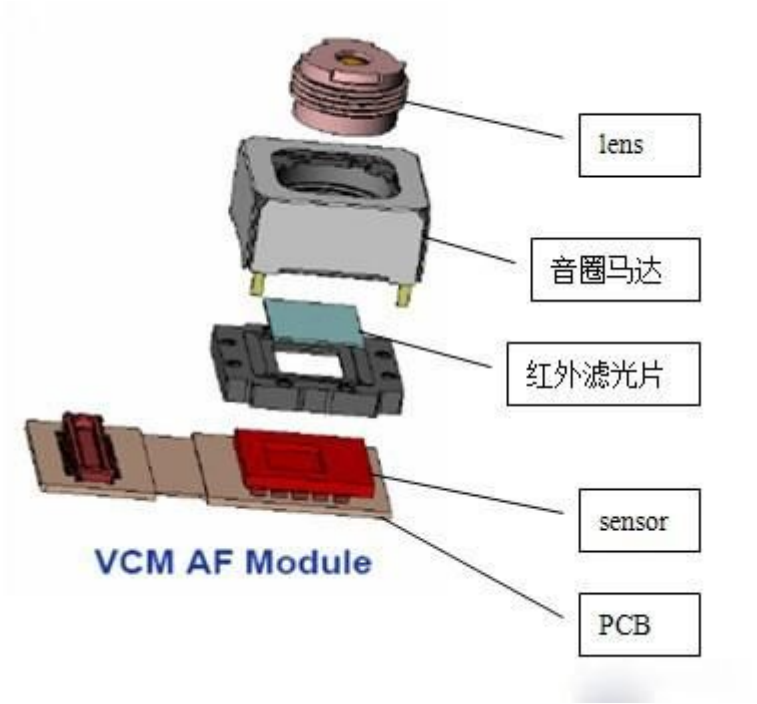
3. USB device controller(USB 设备控制器)

常见的摄像头传感器类型主要有两种，

一种是 CCD 传感器（Chagre Couled Device），即电荷耦合器。

一种是 CMOS 传感器（Complementary Metal-Oxide Semiconductor）即互补性金属氧化物半导体。

CCD 的优势在于成像质量好，但是制造工艺复杂，成本高昂，且耗电高。在相同分辨率下，CMOS 价格比 CCD 便宜，但图像质量相比 CCD 来说要低一些。CMOS 影像传感器相对 CCD 具有耗电低的优势，加上随着工艺技术的进步，CMOS 的画质水平也不断地在提高，所以目前市面上的手机摄像头都采用 CMOS 传感器。



手机摄像头的简单结构

滤光片有两大功用:

1.滤除红外线。滤除对可见光有干扰的红外光，使成像效果更清晰。

2.修整进来的光线。感光芯片由感光体(CELL)构成,最好的光线是直射进来,但为了怕干扰到邻近感光体,就需要对光线加以修整,因此那片滤光片不是玻璃,而是石英片,利用石英的物理偏光特性,把进来的光线,保留直射部份,反射掉斜射部份,避免去影响旁边的感光点.

二、相关参数和名词

1、常见图像格式

1.1 RGB 格式：

传统的红绿蓝格式，比如 RGB565，RGB888，其 16-bit 数据格式为 5-bit R + 6-bit G + 5-bit B。G 多一位，原因是人眼对绿色比较敏感。

1.2 YUV 格式：

luma (Y) + chroma (UV) 格式。YUV 是指亮度参量和色度参量分开表示的像素格式，而这样分开的好处就是不但可以避免相互干扰，还可以降低色度的采样率而不会对图像质量影响太大。YUV 是一个比较笼统地说法，针对它的具体排列方式，可以分为很多种具体的格式。

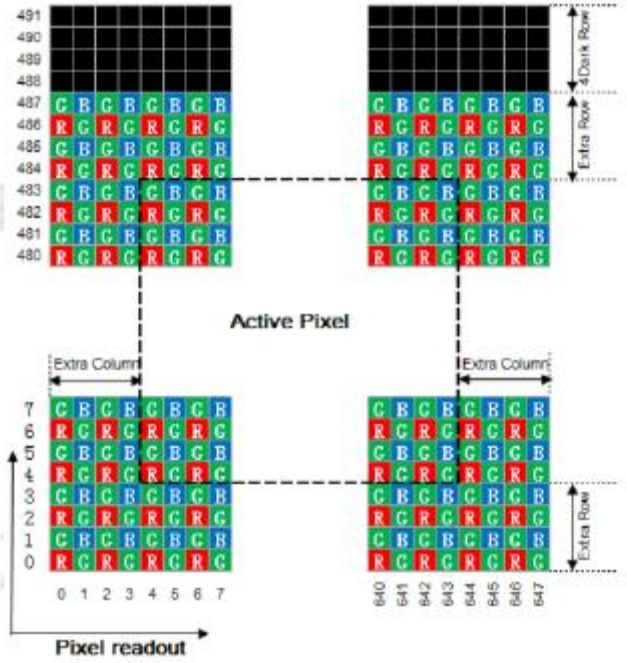
色度(UV)定义了颜色的两个方面–色调与饱和度，分别用 CB 和 CR 表示。其中，Cr 反映了 RGB 输入信号红色部分与 RGB 信号亮度值之间的差异。而 Cb 反映的是 RGB 输入信号蓝色部分与 RGB 信号亮度值之间的差异。

主要的采样格式有 YCbCr 4:2:0、YCbCr 4:2:2、YCbCr 4:1:1 和 YCbCr 4:4:4。

1.3 RAW data 格式：

RAW 图像就是 CMOS 或者 CCD 图像感应器将捕捉到的光源信号转化为数字信号的原始数据。RAW 文件是一种记录了数码相机传感器的原始信息，同时记录了由相机拍摄所产生的一些元数据（Metadata，如 ISO 的设置、快门速度、光圈值、白平衡等）的文件。RAW 是未经处理、也未经压缩的格式，可以把 RAW 概念化为“原始图像编码数据”或更形象的称为“数字底片”。sensor 的每一像素对应一个彩色滤光片，滤光片按 Bayer pattern 分布。将每一个像素的数据直接输出，即 RAW RGB data

Raw data（Raw RGB）经过彩色插值就变成 RGB.



RAW 格式图像示例

2. 相关技术指标

2.1 图像解析度/分辨率(Resolution)：

SXGA(1280 x1024)又称 130 万像素

XGA(1024 x768)又称 80 万像素

SVGA(800 x600)又称 50 万像素

VGA(640x480)又称 30 万像素(35 万是指 648X488)

CIF(352x288) 又称 10 万像素

SIF/QVGA(320x240)

QCIF(176x144)

QSIF/QQVGA(160x120)

2.2 彩色深度(色彩位数)：

256 色灰阶，有 256 种灰色（包括黑白）。

15 或 16 位彩色（高彩色）：65,536 种颜色。

24 位彩色（真彩色）：每种原色都有 256 个层次，它们的组合便有 256*256*256 种颜色。

32 位彩色：除了 24 位彩色的颜色外，额外的 8 位是储存重叠图层的图形资料(alpha 频道)。

2.3 光学变焦和数码变焦：

光学变焦： 通过镜头的调整，拉近拉远所要拍摄的对象，保持像素不变和画质基本不变，却可以拍到自己 理想的物像。

数码变焦：其实没有什么变焦，只是从原图片中截取出来放大，你从液晶屏幕上看到变大了，实际上画质并没有本质提高，而像素比你相机能拍摄的最大像素降低了。 画质上说基本是鸡肋把，但是可以提供一些方便。

2.4 图像压缩方式：

JPEG/M-JPEG

H.261/H.263

MPEG

H.264

2.5 图像噪音：

指的是图像中的杂点干挠。表现为图像中有固定的彩色杂点。

2.6 自动白平衡处理技术(auto White Balance)：

简单来说就是：摄像机对白色物体的还原。相关概念：色温。

2.7 视角：

与人的眼睛成像是相成原理，简单说就是成像范围。

2.8 自动对焦：

自动对焦可以分成两大类：一类是基于镜头与被拍摄目标之间距离测量的测距自动对焦，另一类是基于对焦屏上成像清晰的聚焦检测自动对焦(清晰度算法)。

注：变焦就是把远处的物体拉近。对焦是让图像清晰。

2.9 自动曝光和 Gamma：

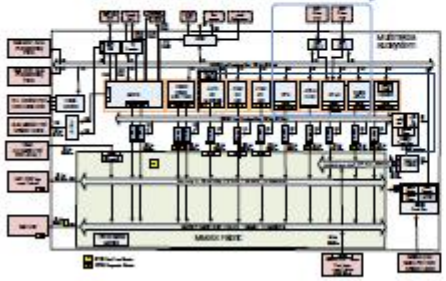
就是光圈和快门的组合。光圈，快门速度，ISO。Gamma 即人眼对亮度的响应曲线。

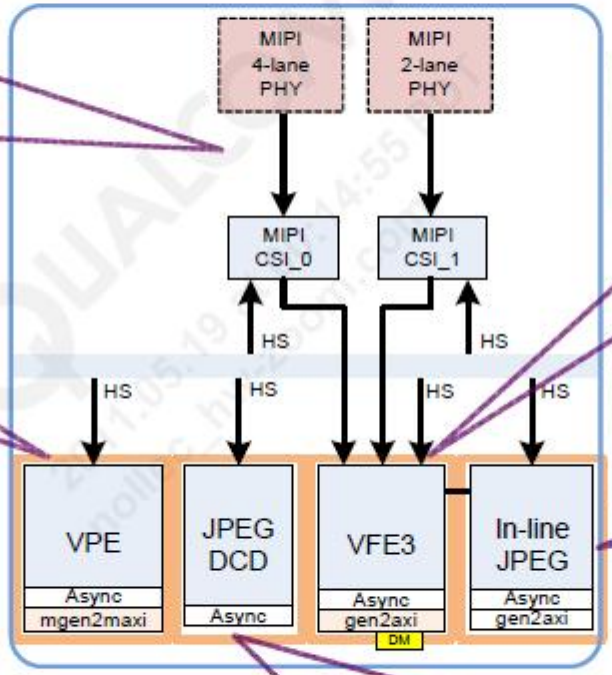
三、高通的 CAMERA 部分硬件架构

Camera Hardware Architecture

MIPI interface
- Offers 2-lane MIPI with 2048 Mbps limit and 4-lane MIPI interface with 2128 Mbps limit.
- Camera data from either is processed by internal MIPI MUX and sent to VFE 3.1 core for processing.

Video preprocessing (VPE)
- Used to zoom, rotate, and stabilize the camera frames before they get encoded by video encoder





VFE 3.1
- Offers advanced image processing blocks to process color and spatial data in real time

Inline JPEG
- Offers hardware accelerated JPEG encoding

JPEG DCD
- Will offer hardware-accelerated JPEG decoding

CAMERA 部分硬件架构

VFE：VIDEO front-end 视频前端

VPE：Video preprocessing 视频预处理

摄像头模组中自带了 ISP（图像信号处理器），所以，VFE 和 VPE 有关图像效果处理的功能都是关闭的。

1.VFE 的功能：

1.1 通过算法提高图像的质量。

1.2 提供高分辨率的图像的 AWB(自动白平衡)/AE(自动曝光)/AF(自动对焦)算法处理。

1.3 图像衰减校正。

1.4 低光下的噪声滤波。

1.5 图像色彩效果优化。

1.6 皮肤颜色效果优化。

1.7 图像抖动计算。

1.8 亮度适应算法。

2.VPE 的功能：

2.1 图像稳定性。

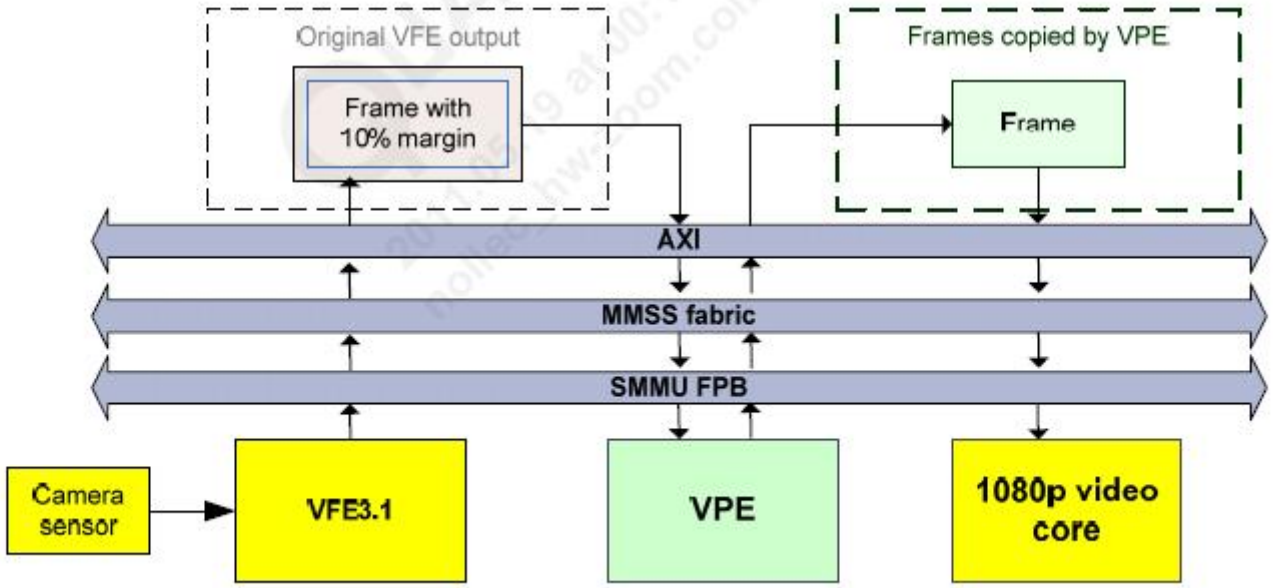
2.2 数字对焦。

2.3 图像旋转。

2.4 Overlay。

VPE Features

- VPE supports features such as:
  - Digital image stabilization
  - Digital zoom
  - Rotation
  - Overlay



三、android 系统 camera 基本架构

1.应用层

Camera 的应用层在 Android 上表现为直接调用 SDK API 开发的一个 Camera 应用 APK 包。代码在/android/packages/apps/Camera 下。主要对 android.hardware.Camera（在 Framework 中） 类的调用，并且实现 Camera 应用的业务逻辑和 UI 显示。一个 Android 应用中若要使用这个 android.hardware.Camera 类，需要在 Manifest 文件声明 Camera 的权限，另外还 需要添加一些< uses-feature> 元素来声明应用中的 Camera 特性，如自动对焦等。 具体做法可如下：

<uses-permission android:name = "android.permission.CAMERA" />

<uses-feature android:name = "android.hardware.camera" />

<uses-feature android:name = "android.hardware.camera.autofocus" />

2.Framework 层

2.1 android.hardware.Camera：代码位置/android/frameworks/base/core/java/android/hardware/Camera.java

这部分目标是 framework.jar。这是是 Android 提供给 app 层调用的 java 接口。这个类用来连接或断开一个 Camera 服务，设置拍摄参数，开始、停止预览，拍照等。

2.2 android.hardware.Camera 这个类是和 JNI 中定义的类是一个，有些方法通过 JNI 的方式调用本地代码得到，有些方法自己实现。

Camera 的 JAVA native 调用部分（JNI）：/android/frameworks/base/core/jni/android_hardware_Camera.cpp。Camera.java 承接 JAVA 代码到 C++ 代码的桥梁。编译生成 libandroid_runtime.so 。libandroid_runtime.so 库是公用的, 其中除了 Camera 还有其他方面的功能。

2.3 Camera 框架的 client 部分：

代码位置：/android/frameworks/base/libs/camera/下 5 个文件。

Camera.cpp

CameraParameters.cpp

ICamera.cpp

ICameraClient.cpp

ICameraService.cpp

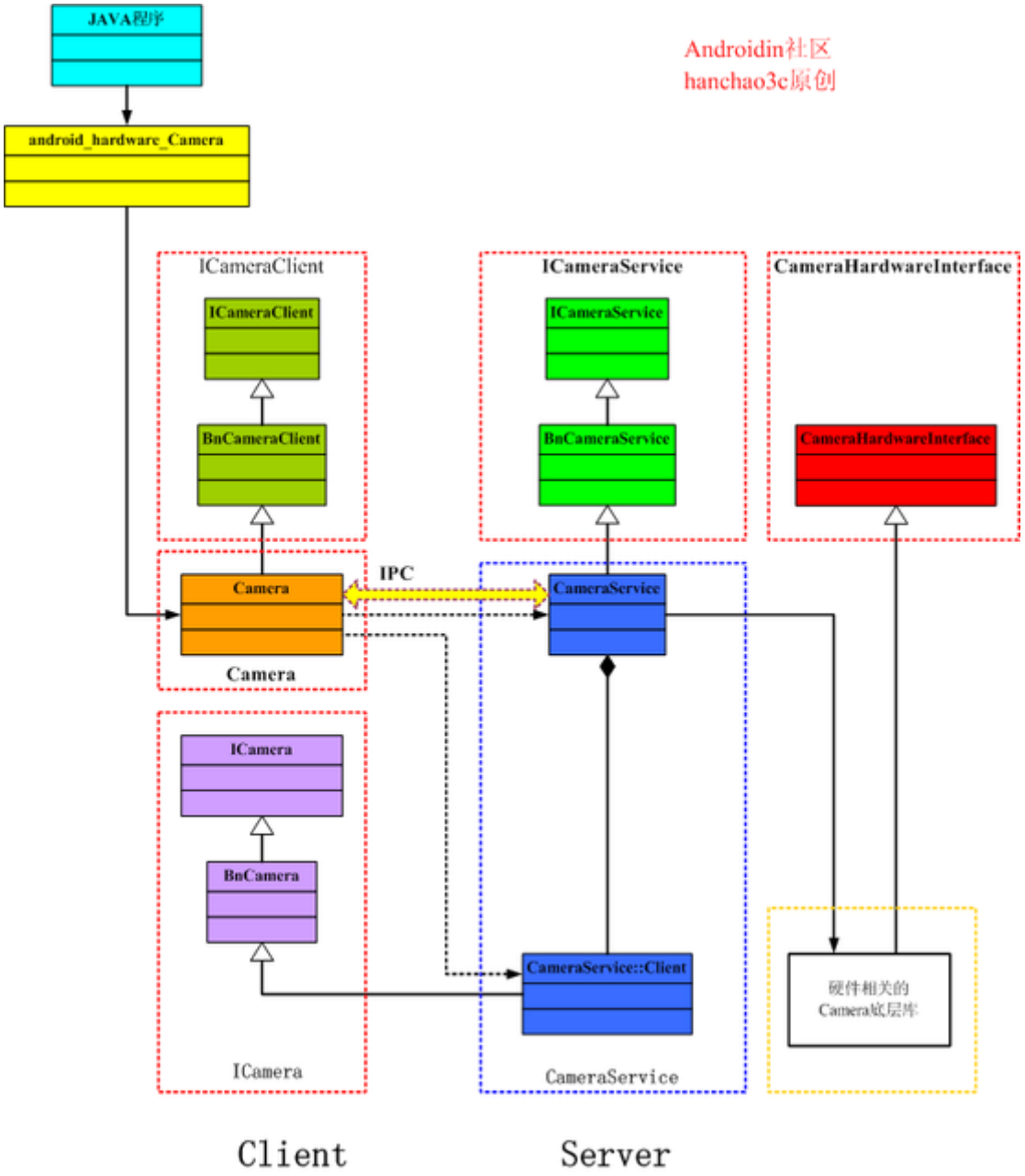
它们的头文件在/android/frameworks/base/include/camera 目录下。

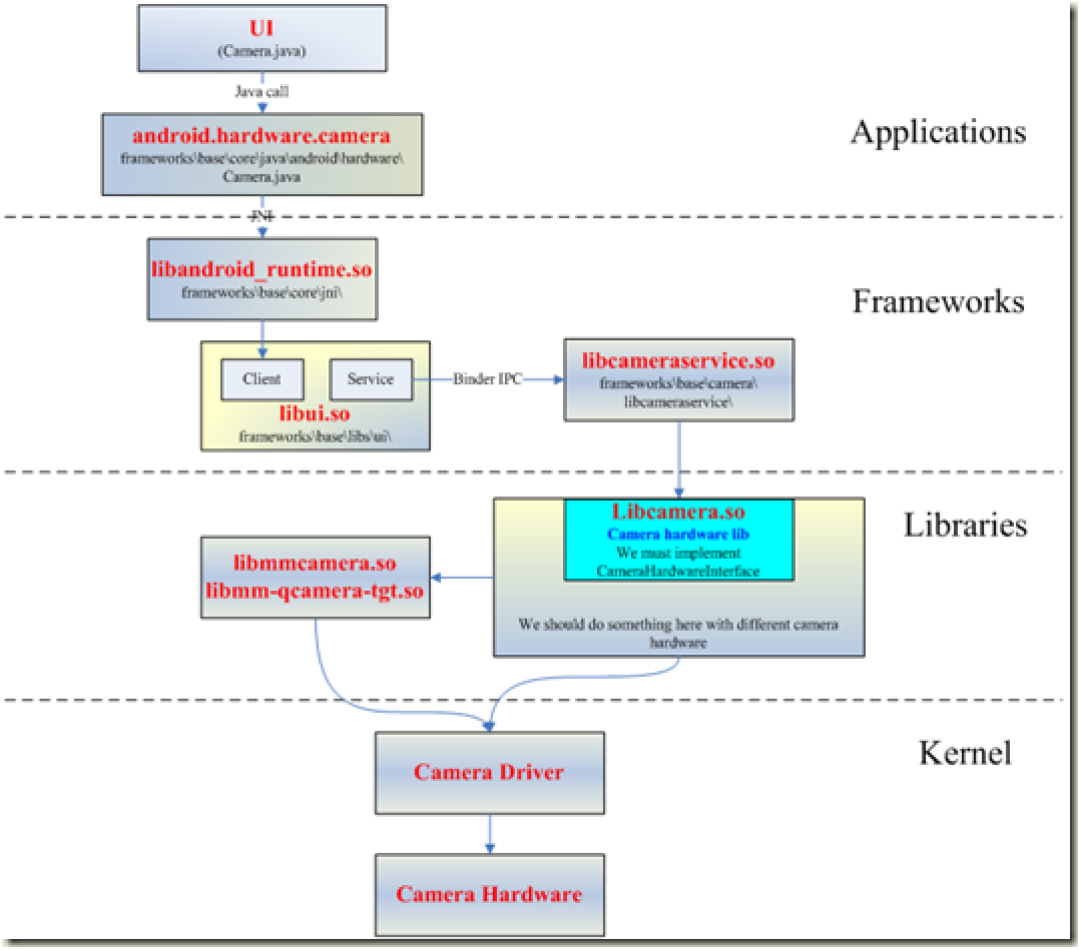
这部分的内容编译生成 libcamera_client.so 。在 Camera 模块的各个库中，libcamera_client.so 位于核心的位置，作为 Camera 框架的 Client 客户端部分，与另外一部分内容服务端 libcameraservice.so 通过进程间通讯（即 Binder 机制）的方式进行通讯。

2.4 Camera 框架的 service 部分：

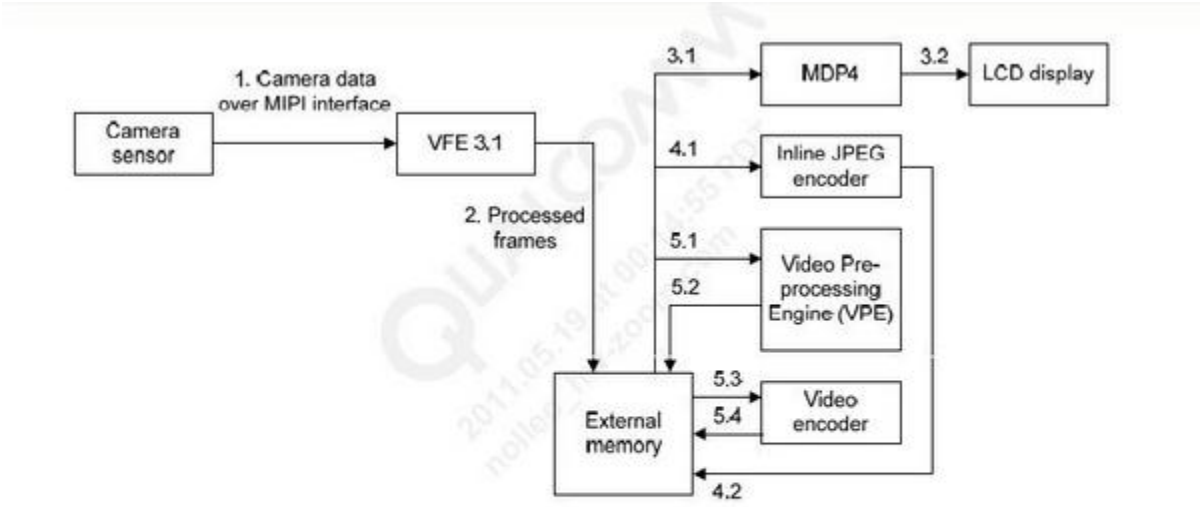
代码位置：/android/frameworks/base/services/camera/libcameraservice。

这部分内容被编译成库 libcameraservice.so 。CameraService 是 Camera 服务，Camera 框架的中间层，用于链接 CameraHardwareInterface 和 Client 部分 ，它通过调用实际的 Camera 硬件接口来实现功能，即下层 HAL 层。

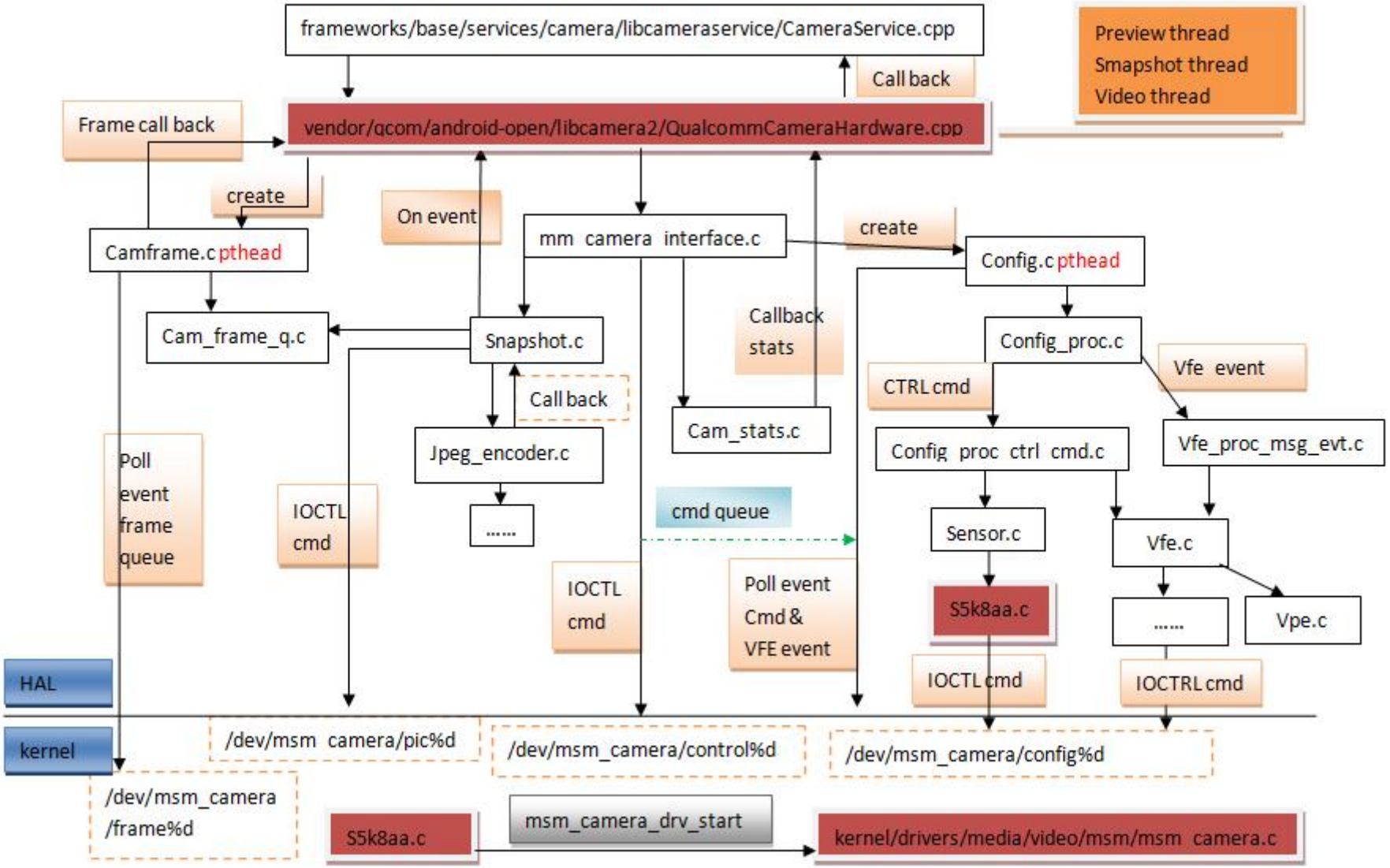




四. 摄像头预览、拍照、录像基本数据流向和处理流程以及驱动调试



HAl 层相关代码：（frameworks/base/services/camera/libcameraservice/CameraService.cpp）
vendor/qcom/android-open/libcamera2/QualcommCameraHardware.cppvendor/qcom/proprietary/mm-camera/apps/appslib/mm_camera_interface.cvendor/qcom/proprietary/mm-camera/apps/appslib/camframe.cvendor/qcom/proprietary/mm-camera/apps/appslib/snapshot.cvendor/qcom/proprietary/mm-camera/apps/appslib/jpeg_encoder.cvendor/qcom/proprietary/mm-camera/apps/appslib/cam_frame_q.cvendor/qcom/proprietary/mm-camera/apps/appslib/cam_display.cvendor/qcom/proprietary/mm-camera/targets/vfe31/8x60/vendor/qcom/proprietary/mm-camera/targets/vfe31/common/vpe1/QualcommCameraHardware.cpp 主要分为三个部分，preview，snapshot，video。它们分别用一个 pthread 进行处理。另外还有 auto focus 功能也是用 pthread 的方式处理。预览或拍照、视频线程处理得到的数据帧都以 datacallback 的方式回调到上层 CameraService.cpp 中，进行存储或预览等操作。以下是 HAL 层部分的代码大概的调用结构流程。


frameworks/base/services/camera/libcameraservice/CameraService.cpp
Call back
Preview thread
Smapshot thread
Video thread
Frame call back
vendor/qcom/android-open/libcamera2/QualcommCameraHardware.cpp
create
On event
mm_camera_interface.c
create
Config.c pthead
Camframe.c pthead
Cam_frame_q.c
Snapshot.c
Callback
stats
Config_proc.c
Vfe event
CTRL cmd
Call back
Cam_stats.c
Jpeg_encoder.c
Config_proc_ctrl_cmd.c
Vfe_proc_msg_evt.c
Poll
event
frame
queue
IOCTL
cmd
.....
cmd queue
Sensor.c
Vfe.c
IOCTL
cmd
Poll event
Cmd &
VFE event
S5k8aa.c
.....
Vpe.c
IOCTL cmd
IOCTRL cmd
HAL
kernel
/dev/msm_camera/pic%d
/dev/msm_camera/control%d
/dev/msm_camera/config%d
/dev/msm_camera
/frame%d
S5k8aa.c
msm_camera_drv_start
kernel/drivers/media/video/msm/msm_camera.c

1. 整个模块主要巡行三个主线程：control、config 及 frame。
2. control 用来执行总的控制，是上层控制接口。

   config 主要进行一些配置，这个线程里面主要进行 3A 的工作，另外还有一些跟效果有关的设置；

   frame 线程主要用来做帧 queue 的循环获取处理。所有事件或状态的反馈，用回调函数的方式传回 QualcommCameraHardware.cpp。

2. 驱动部分从设备驱动 s5k8aa.c 开始。新建平台设备后，执行入口函数 probe 时，调用创建摄像头设备功能函数

```
int msm_camera_drv_start(struct platform_device *dev,
                int (*sensor_probe)(const struct msm_camera_sensor_info *,
                      struct msm_sensor_ctrl *))
```

并将设备信息结构体和摄像头设备调用入口 sensor_probe 传入。msm_camera_drv_start（xxx）函数在 msm_camera.c 中实现。他创建了提供上层调用的四个终于设备结点：

/dev/msm_camera/frame%d

/dev/msm_camera/control%d

/dev/msm_camera/config%d

/dev/msm_camera/pic%d

实现了上层库对 VFE 模块，VPE 模块，jpeg_encoder 模块和摄像头 sensor 模块驱动的控制调用接口。在 file_operations 中的相应函

数中分别实现的是这些设备的新建初始化和 IOCTL 功能调用接口。

然后这个函数还创建了四个工作队列：

struct msm_device_queue event_q;

struct msm_device_queue frame_q;

struct msm_device_queue pict_q;

struct msm_device_queue vpe_q;

event_q 包括/dev/msm_camera/control%d 传入的控制信号队列，用于将上层传下来的控制命令(command)传到 config thread 中去。

frame_q 用于对图像帧的操作管理，预览或录像时帧将传递给 DSP 进行处理。

pict_q 包含拍照帧，用于给 jpeg_encoder 进行图像编码处理。

vpe_q 是 VPE 控制命令队列。

s5k8aa.c 是相应摄像头设备的驱动部分。它的功能很简单，主要实现 sensor 模块的创建、初始化和控制。主要实现以下三个函数：

s->s_init = ov2685_sensor_init;

s->s_release = ov2685_sensor_release;

s->s_config = ov2685_sensor_config;

ov2685_sensor_init 函数：

主要实现摄像头的上电、时钟控制（MCLK）、设备初始化功能。　　上电分为 DOVDD、DVDD、AVDD、reset、PWDN 几个部分。需要按照设备要求顺序操作，一般时钟控制顺序也包含在内。　　设备初始化过程是将 sensor 设备的所有寄存器全部初始化一遍，采用 IIC 方式将初始化寄存器地址和值全部发送到 sensor 端。完成后此时摄像头模组才能正常工作，并将图像通过 MIPI 线路传送到 CPU 端。

ov2685_sensor_config 函数：

主要实现对 sensor 的各种配置接口，相应的有帧率配置，白平衡效果设置，曝光度设置，特效设置等等。相应接口将配置好的寄存器列表通过 IIC 发送到 sensor 中。

3. 摄像头调试中的几个问题点：

1.1 是否正确上电，是否有时钟波形输出。　　检测输出电压的电压值是否和上电时序以及 MCLK 是否符合 sensor 的要求。这部分可以用示波器和万用表测量。测量电压值和上电时序以及 MCLK 的时钟频率是否正确。

1.2 IIC 读写是否正常。调试 CPU 与 ISP 间的 I2C 通信。　　检测包括 IIC 地址是否正确，协议是否匹配。这部分也可以用示波器测量 IIC 的 SDA、CLK 的峰值、波形逻辑是否正确。

1.3 正确上电并初始化以后 sensor 模块是否正常工作。　　这部分主要通过用示波器测量 MIPI 线路的数据和时钟 PIN 是否正确，它的波形是否含有数据，是否标准波形，峰值有没有达到要求等。

1.4 如果以上都正确了以后，MIPI 控制器将接收到中断，并开始处理图像信号。此时如果出错，可以通过中断信号的出错值查看错误状态。除 CPU 端是否正常初始化工作的问题外，需要关注模组端设置的图像格式和 CPU 接收的默认图像格式和图像大小（SIZE）是否一致。模组中图片格式和图像大小通过寄存器值查看。CPU 端接收图片格式和图像大小在 HAL 部分的 s5k8aa 中设置，拍照源图像大小和预览源图像大小需要分别设置。

以上部分完成后，摄像头可以正确预览。