Android 的照相机系统

韩　超
@ Android 技术

Android 的 Camera 系统

- 第一部分 Camera 系统的结构
- 第二部分 移植和调试的要点
- 第三部分 Camera 实现方式

第一部分 Camera 系统的结构

照相机系统下层的硬件通常是摄像头设备，主要用于向系统输入视频数据。摄像头设备通常包括处理器中的数据信号处理相关的控制器和摄像头传感器。摄像头传感器又可以分为普通型和智能型的。摄像机硬件对软件部分主要提供视频数据。

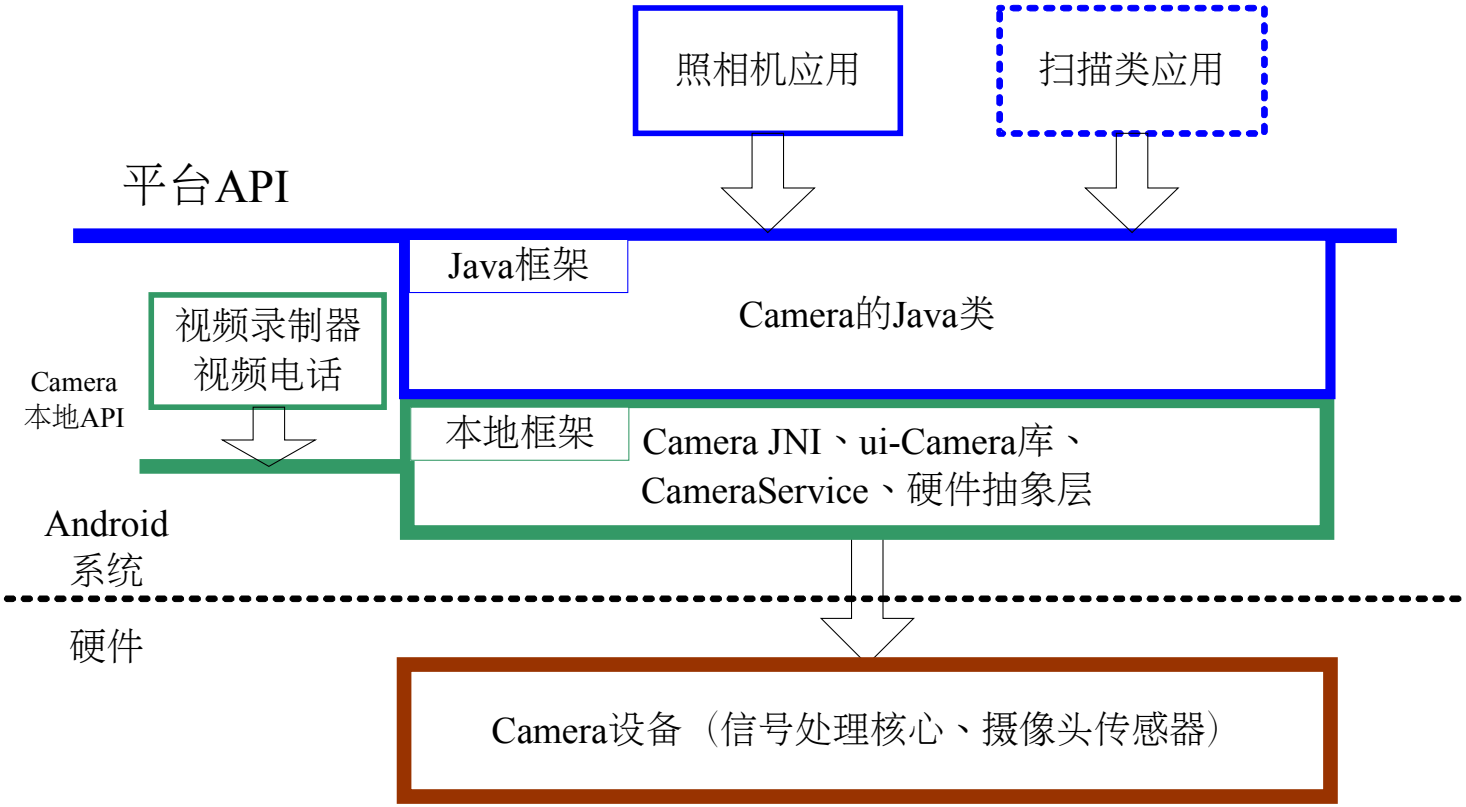
第一部分 Camera 系统的结构

照相机系统对上层的接口提供了取景器、视频录制、拍摄相片三个方面的主要功能，还有各种控制类的接口。照相机系统提供了 Java 层的接口和本地接口：一方面， Java 框架中 Camera 类，提供 Java 层照相机接口，为照相机和扫描类应用使用；另一方面， Camera 本地接口也可以给本地程序调用，通常作为视频的输入环节，在摄像机和视频电话中使用。

第一部分 Camera 系统的结构

在理论上，照相机的取景器、视频、照片等数据都可以传送到 Java 层，但是通常情况下，这些数据不需要传递到 Java 层。仅有少数情况需要在 Java 层获取数据流，例如通过摄像头进行扫面识别的时候，需要取景器的数据帧。

第一部分 Camera 系统的结构



第一部分 Camera 系统的结构

自下而上， Camera 系统分成了以下几个部分。
（1）摄像头驱动程序：通常基于 Linux 的 Video for Linux 视频驱动框架。
（2） Camera 硬件抽象层
frameworks/base/include/ui/
frameworks/base/include/camera/
主要的文件为 CameraHardwareInterface.h， 需要各个系统根据自己的情况实现。
（3） Camera 服务部分
frameworks/base/camera/libcameraservice/
Camera 服务是 Android 系统中一个单独部分，通过调用 Camera 硬件抽象层来实现。

第一部分 Camera 系统的结构

自下而上， Camera 系统分成了以下几个部分。
（1）摄像头驱动程序：通常基于 Linux 的 Video for Linux 视频驱动框架。
（2） Camera 硬件抽象层
frameworks/base/include/ui/
frameworks/base/include/camera/
主要的文件为 CameraHardwareInterface.h， 需要各个系统根据自己的情况实现。
（3） Camera 服务部分
frameworks/base/camera/libcameraservice/
Camera 服务是 Android 系统中一个单独部分，通过调用 Camera 硬件抽象层来实现。

第一部分 Camera 系统的结构

（4） Camera 的本地框架代码

头文件路径： frameworks/base/include/ui/ 或者 frameworks/base/include/camera/ 。

源代码路径： frameworks/base/libs/ui/ 或者 frameworks/base/libs/camera/ 。

Camera 系统是其中的一部分，这部分内容被编译成库 libui.so 或者 libcamera_client.so 。

（5） Camera 的 JNI 代码

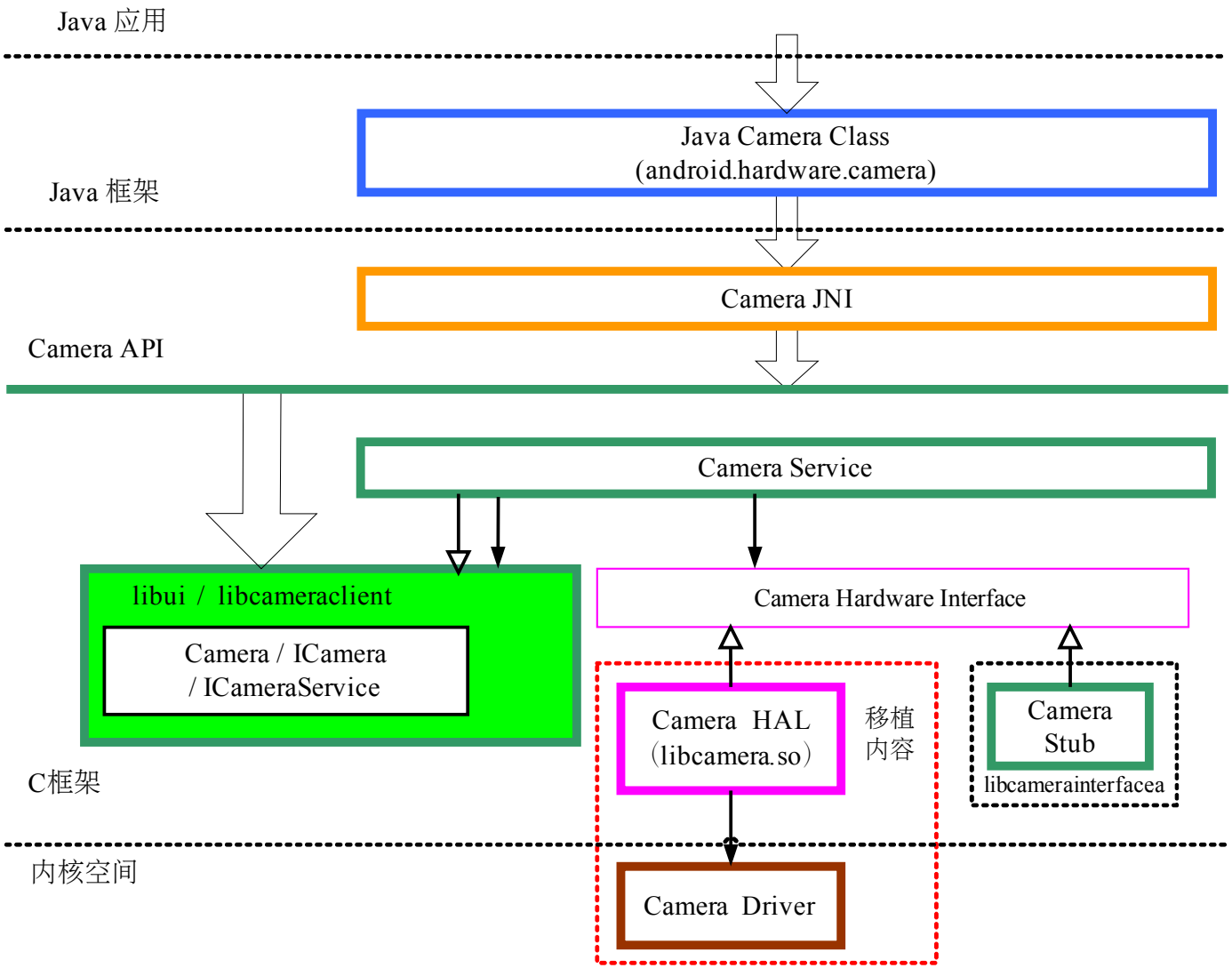
frameworks/base/core/jni/android_hardware_Camera.cpp 。

提供给 Java 类的本地支持，也包含了反向调用 Java 传递信息和数据功能。

（6） Camera 系统的 Java 类

frameworks/base/core/java/android/hardware/Camera.java 。

第四部分 Camera 系统与上层接口



第二部分 移植和调试的要点

2.1 Camera 驱动程序

2.2 硬件抽象层的内容

2.3 Camera 系统上层调用情况

2.1 Camera 驱动程序

摄像头（Camera）－视频驱动驱动通常使用 Video For Linux。

v4l2 驱动的设备节点：

/dev/video/videoX

主设备号为 81，次设备号 0-63。

v4l2 驱动主要头文件路径：

include/linux/videodev.h： v4l 第一版的头文件

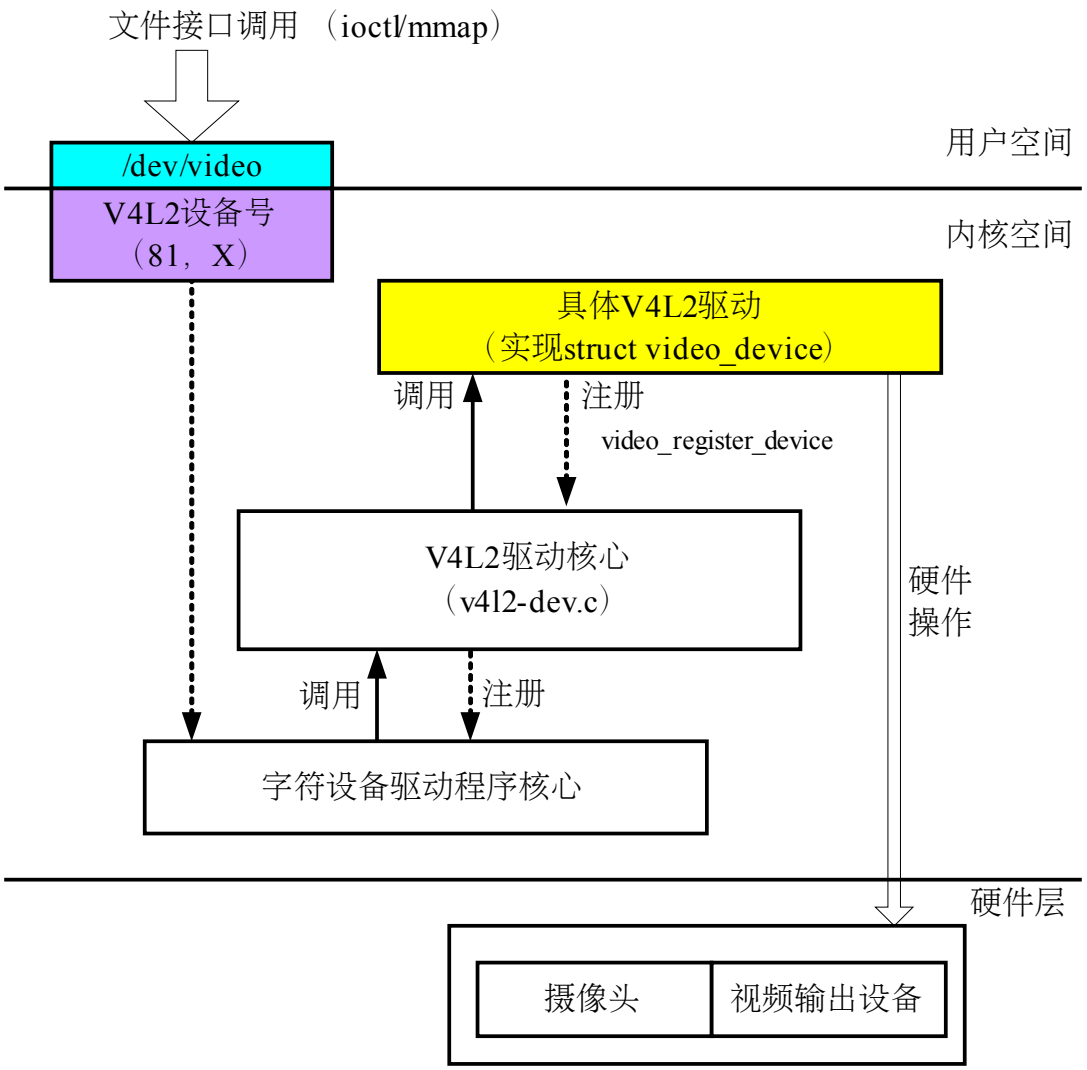
include/linux/videodev2.h： 定义主要的数据接口和常量

include/media/v4l2-dev.h： 设备头文件，具体设备使用其中的接口注册

v4l2 驱动核心实现路径：

driver/media/video/v4l2-dev.c

2.1 Camera 驱动程序



2.2 硬件抽象层的内容

Camera 的硬件抽象层的在 UI 库的头文件 CameraHardwareInterface.h 文件定义。

在这个接口中，包含了控制通道和数据通道，控制通道用于处理预览和视频获取的开始 / 停止、拍摄照片、自动对焦等功能，数据通道通过回调函数来获得预览、视频录制、自动对焦等数据。

Camera 的硬件抽象层中还可以使用 Overlay 来实现预览功能。

2.2 硬件抽象层的内容

CameraHardwareInterface.h 文件的定义：

```
typedef void (*notify_callback)(int32_t msgType,
                                int32_t ext1,
                                int32_t ext2,
                                void* user);

typedef void (*data_callback)(int32_t msgType,
                              const sp<IMemory>& dataPtr,
                              void* user);

typedef void (*data_callback_timestamp)(nsecs_t timestamp,
                                        int32_t msgType,
                                        const sp<IMemory>& dataPtr,
                                        void* user);
```

2.2 硬件抽象层的内容

CameraHardwareInterface 类的定义：

```
class CameraHardwareInterface : public virtual RefBase {
public:
    virtual ~CameraHardwareInterface() { }
    virtual sp<IMemoryHeap>         getPreviewHeap() const = 0;
    virtual sp<IMemoryHeap>         getRawHeap() const = 0;
    virtual void setCallbacks(notify_callback notify_cb,
                              data_callback data_cb,
                              data_callback_timestamp data_cb_timestamp,
                              void* user) = 0;
    virtual void        enableMsgType(int32_t msgType) = 0;
    virtual void        disableMsgType(int32_t msgType) = 0;
    virtual bool        msgTypeEnabled(int32_t msgType) = 0;
```

2.2 硬件抽象层的内容

```
    virtual status_t    startPreview() = 0;
    virtual bool         useOverlay() {return false;}
    virtual status_t     setOverlay(const sp<Overlay> &overlay)
                             {return BAD_VALUE;}
    virtual void        stopPreview() = 0;
    virtual bool        previewEnabled() = 0;
    virtual status_t    startRecording() = 0;
    virtual void        stopRecording() = 0;
    virtual bool        recordingEnabled() = 0;
    virtual void        releaseRecordingFrame(const sp<IMemory>& mem) = 0;
    virtual status_t    autoFocus() = 0;
    virtual status_t    cancelAutoFocus() = 0;
    virtual status_t    takePicture() = 0;
    virtual status_t    cancelPicture() = 0;
    virtual status_t    setParameters(const CameraParameters& params) = 0;
    virtual CameraParameters  getParameters() const = 0;
    virtual status_t sendCommand(int32_t cmd, int32_t arg1, int32_t arg2) = 0;
    virtual void release() = 0;
    virtual status_t dump(int fd, const Vector<String16>& args) const = 0;
};
```

2.2 硬件抽象层的内容

取景器预览的主要步骤如下所示：

- 在初始化的过程中，建立预览数据的内存队列（多种方式）；
- 在 startPreview() 的实现中，保存预览回调函数，建立预览线程；
- 在预览线程的循环中，等待视频数据的到达；
- 视频帧到达后调用预览回调函数，将视频帧送出。

如果使用 Overlay 实现取景器，则需要有以下两个变化：

- 在 setOverlay() 函数中，从 ISurface 接口中取得 Overlay 类；
- 在预览线程的循环中，不需要使用预览回调函数，直接将视频数据输入到 Overlay 上。

2.2 硬件抽象层的内容

对于 Linux 系统而言，摄像头驱动部分大多使用 Video for Linux 2 （V4L2）驱动程序，在此处主要的处理流程可以如下所示：

- 如果使用映射内核内存的方式（V4L2_MEMORY_MMAP），构建预览的内存 MemoryHeapBase 需要从 V4L2 驱动程序中得到内存指针；
- 如果使用用户空间内存的方式（V4L2_MEMORY_USERPTR），MemoryHeapBase 中开辟的内存是在用户空间建立的；
- 在预览的线程中，使用 VIDIOC_DQBUF 调用阻塞等待视频帧的到来，处理完成后 使用 VIDIOC_QBUF 调用将帧内存再次压入队列，等待下一帧的到来。

2.2 硬件抽象层的内容

录制视频的主要步骤如下所示：

- 在 startRecording() 的实现（或者在 setCallbacks ）中，保存录制视频回调函数；
- 录制视频可以使用自己的线程，也可以使用预览线程；
- 通过录制回调函数将视频帧送出；

releaseRecordingFrame() 被调用后，表示上层通知 Camera 硬件抽象层，这一帧的内存已经用完，可以进行下一次的处理。

如果在 V4L2 驱动程序中使用原始数据（ RAW ），则视频录制的数据和取景器预览的数据为同一数据。 releaseRecordingFrame() 被调用时，通常表示编码器已经完成了对当前视频帧的编码，对这块内存进行释放。在这个函数的实现中，可以设置标志位，标记帧内存可以再次使用。

2.3 Camera 系统上层调用情况

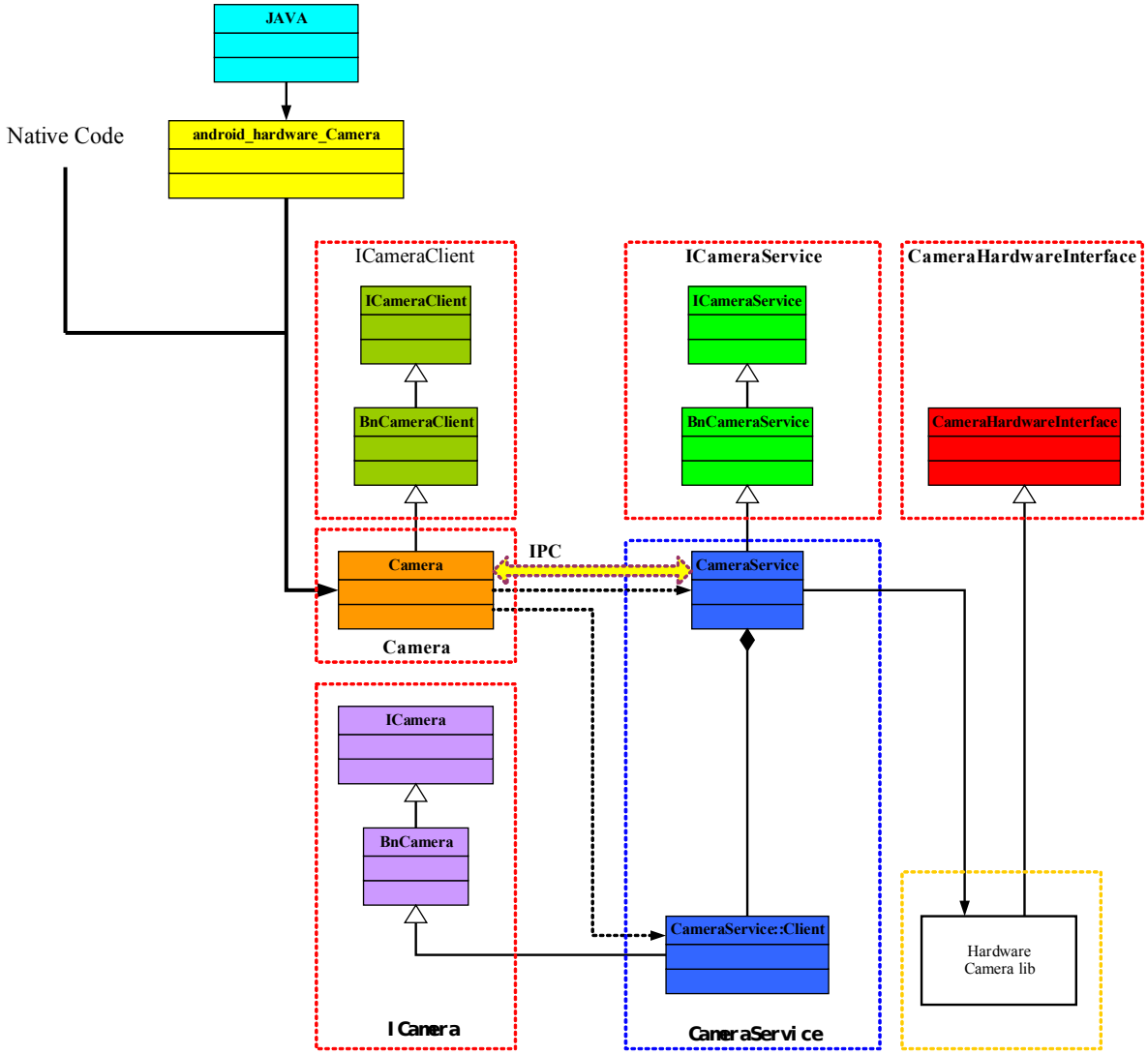
Camera 主要的头文件有以下几个：

- ICameraClient.h
- Camera.h
- ICamera.h
- ICameraService.h

ICameraService.h 、 ICameraClient.h 和 ICamera.h 三个类定义了 Camera 的接口和架构， ICameraService.cpp 和 Camera.cpp 两个文件用于 Camera 架构的实现， Camera 的具体功能在下层调用硬件相关的接口来实现。

Camera.h 是 Camera 系统对上层的接口。

2.3 Camera 系统上层调用情况



2.3 Camera 系统上层调用情况

Camera.h 是 Camera 系统对上层的接口；

ICameraService.h 、 ICameraClient.h 和 ICamera.h 三个类定义了 Camera 中间层实现的框架。它们接口的形式不同，但是具有 Camera 系统共同的几个方面：

- 预览功能（Preview）
- 视频获取功能（Recording）
- 拍照照片（takePicture）
- 参数设置

2.3 Camera 系统上层调用情况

CameraService 是继承 BnCameraService 的实现，在这个类的内部又定义了类 Client，CameraService::Client 继承了 BnCamera。

在运作的过程中 CameraService::connect() 函数用于得到一个 CameraService::Client，在使用过程中，主要是通过调用这个类的接口来实现完成 Camera 的功能，由于 CameraService::Client 本身继承了 BnCamera 类，而 BnCamera 类是继承了 ICamera，因此这个类是可以被当成 ICamera 来使用的。

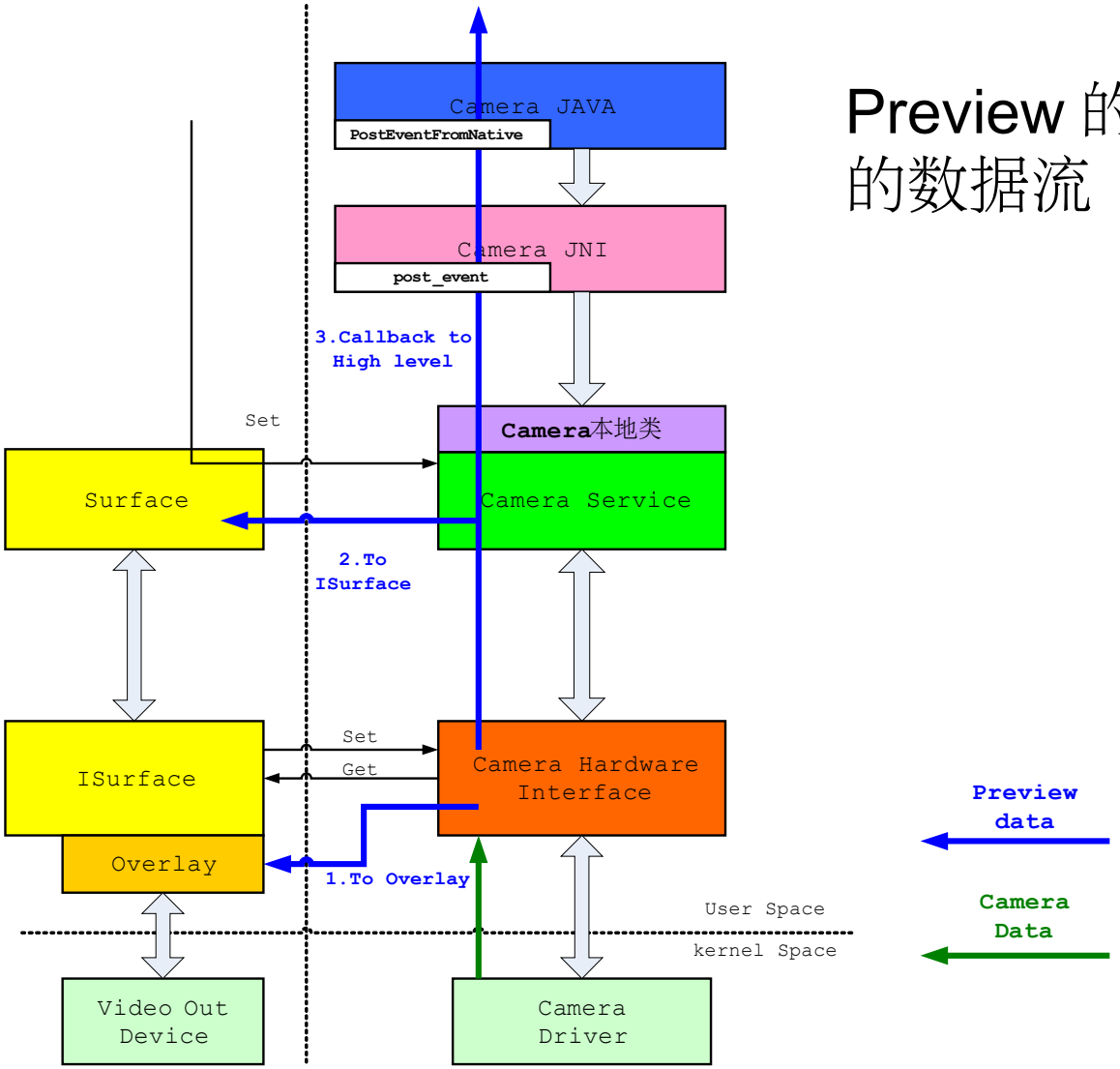
2.3 Camera 系统上层调用情况

照相机的 Preview 数据的走向：

1. 在 CameraHAL 中，直接送给 Overlay
2. 在 CameraService 中，调用 ISurface 的 postBuffer 接口，送出数据。
3. Camera 类通过 Callback 送给上层，由上层处理

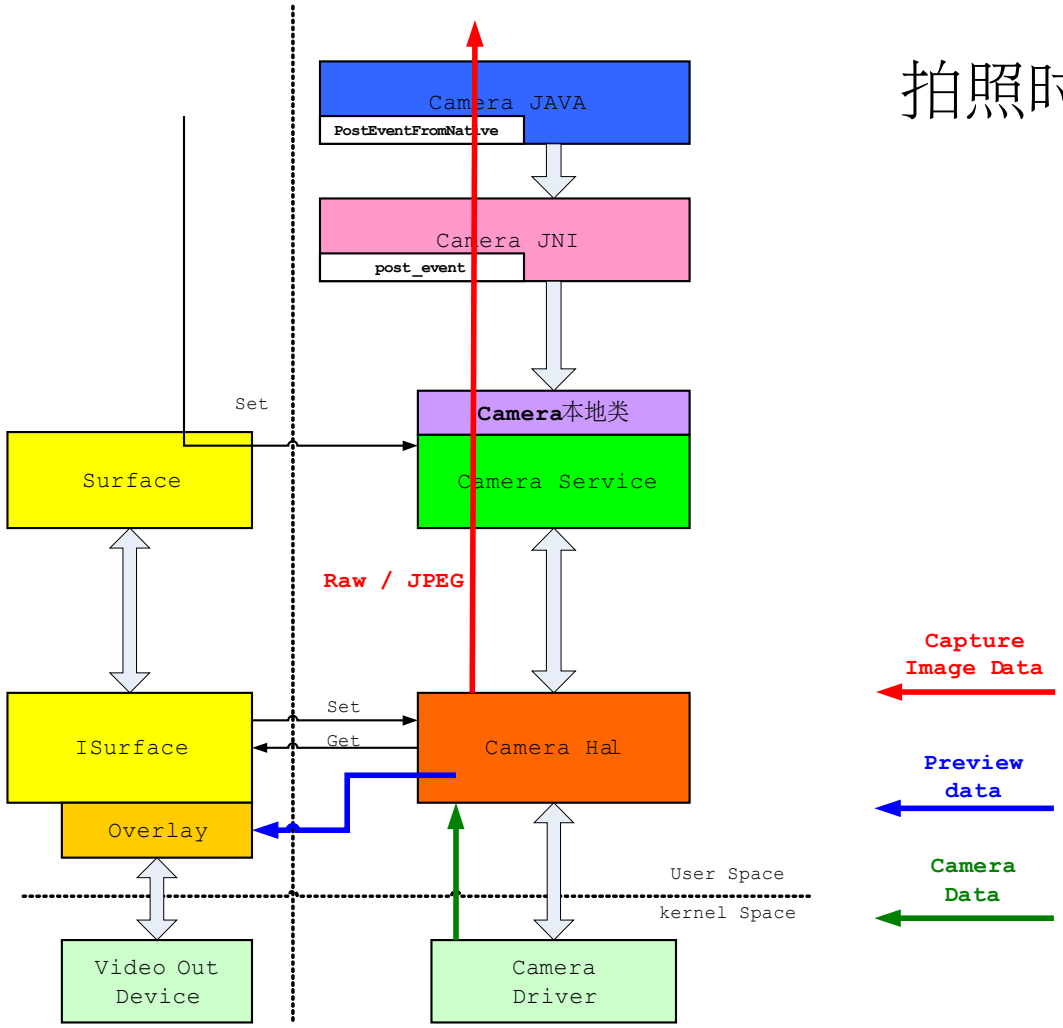
在 Android 照相机 / 摄像机应用程序中，使用的是 2 和 3 这两种方法，具体是 2 还是 3，由 CameraService 读取 CameraHAL 的 useOverlay 接口来实现。

2.3 Camera 系统上层调用情况



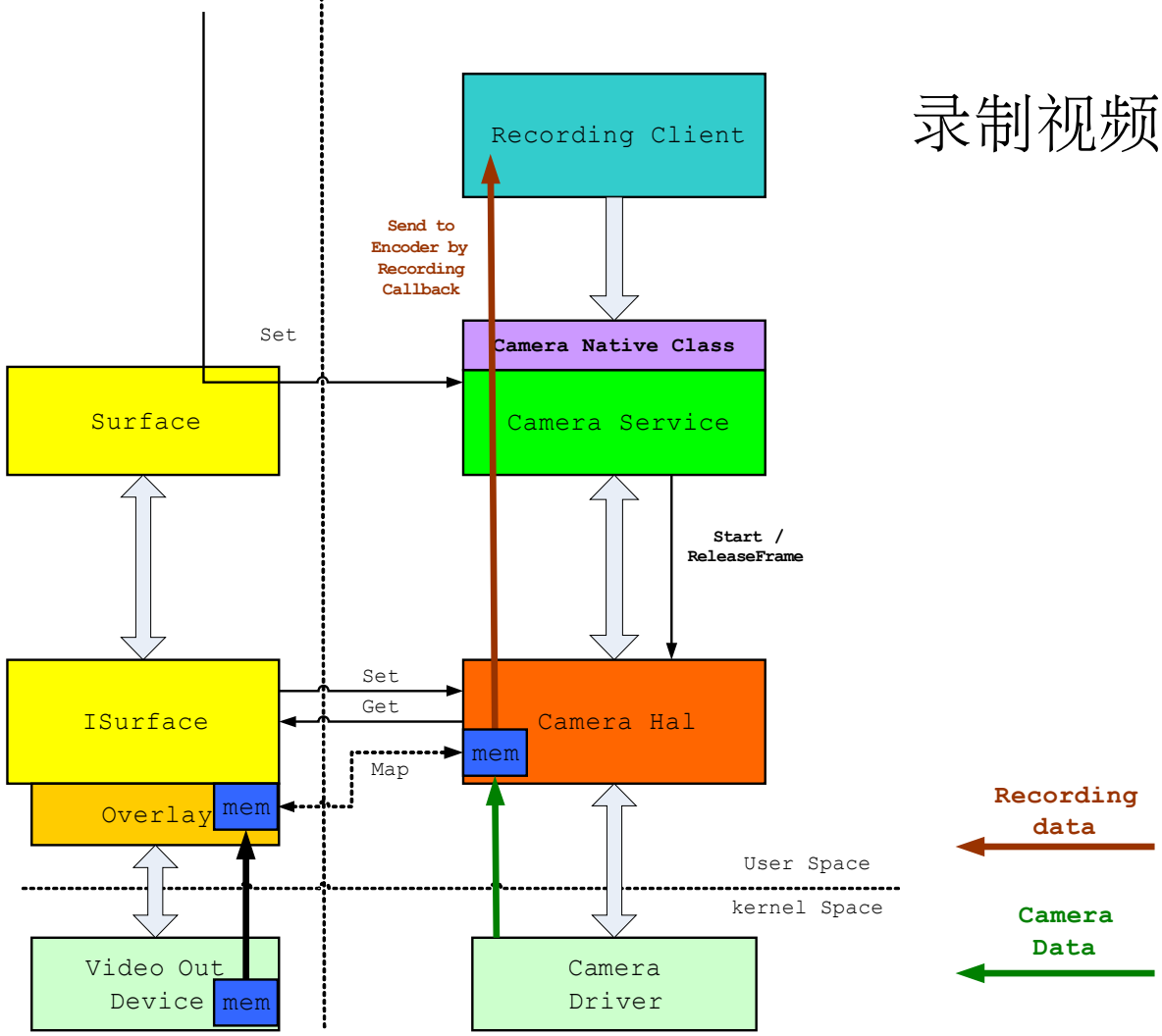
Preview 的三种方式的数据流

2.3 Camera 系统上层调用情况



拍照时的数据流

2.3 Camera 系统上层调用情况



录制视频时的数据流

2.3 Camera 系统上层调用情况

Android 的 Camera 使用 JNI 为向上层 JAVA 提供了接口。 Camera 在 JAVA 中的类是: android.hardware.Camera 。

Camera 的 JAVA 本地调用部分（JNI）：
frameworks/base/core/jni/android_hardware_Camera.cpp

Camera 的 JAVA 类:
frameworks/base/core/java/android/hardware/Camera.java

第三部分 Camera 系统的实现

2.1 Camera 的桩实现

2.2 MSM 平台的 Camera 实现

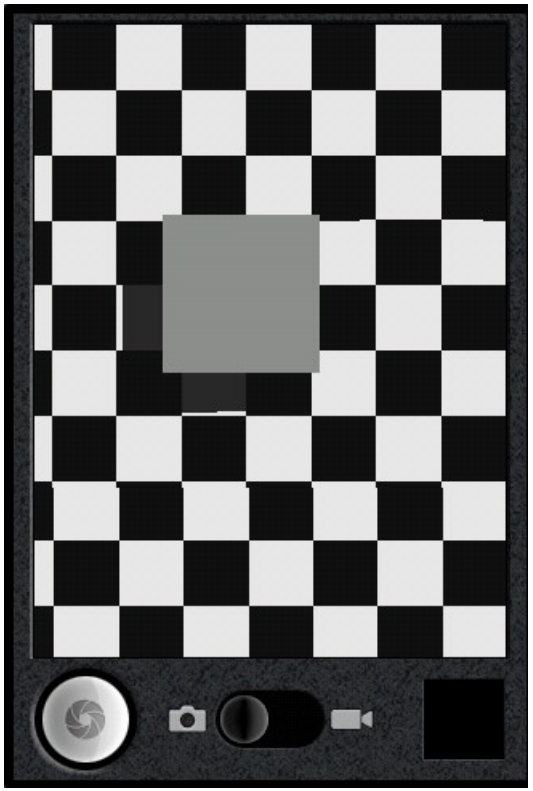
2.3 OMAP 平台的 Camera 实现

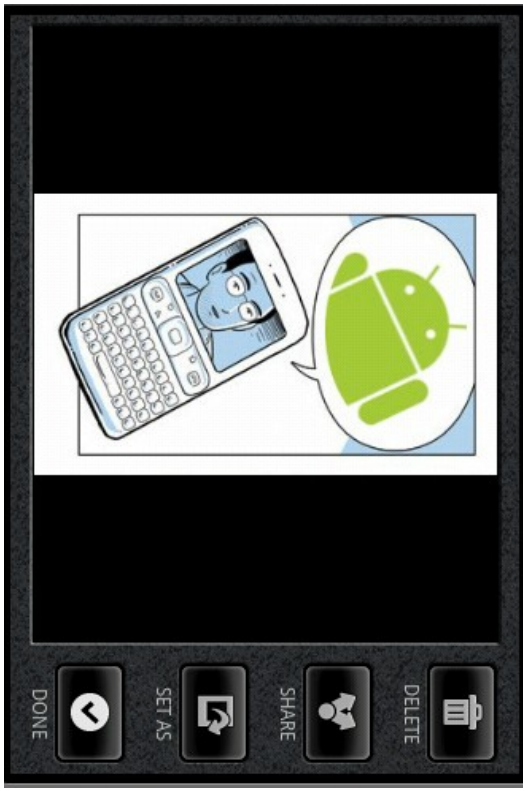
2.1 Camera 的桩实现

Android 中已经实现了一个 Camera 硬件抽象层的“桩（Stub）”，可以根据宏来进行配置。这个桩使用假的方式，可以实现一个取景器预览和拍摄照片等功能。

Camera 硬件抽象层桩实现使用黑白相间的格子代替实际来自硬件的视频流，这样可以在不接触硬件的情况下，让 Android 的 Camera 系统在没有硬件的情况下运行。显然由于没有视频输出设备， Camera 硬件抽象层桩实现是不使用 Overlay 的。

2.1 Camera 的桩实现





2.1 Camera 的桩实现

Camera 硬件抽象层桩实现包含的几个文件如下所示。

CameraHardwareStub.h： Camera 硬件抽象层桩实现的接口

CameraHardwareStub.cpp： Camera 硬件抽象层桩实现

FakeCamera.h 和 FakeCamera.cpp： 实现假的 Camera 的黑白格取景器效果

CannedJpeg.h： 包含一块 JPEG 数据，用于拍摄照片时作为 JPEG 数据。

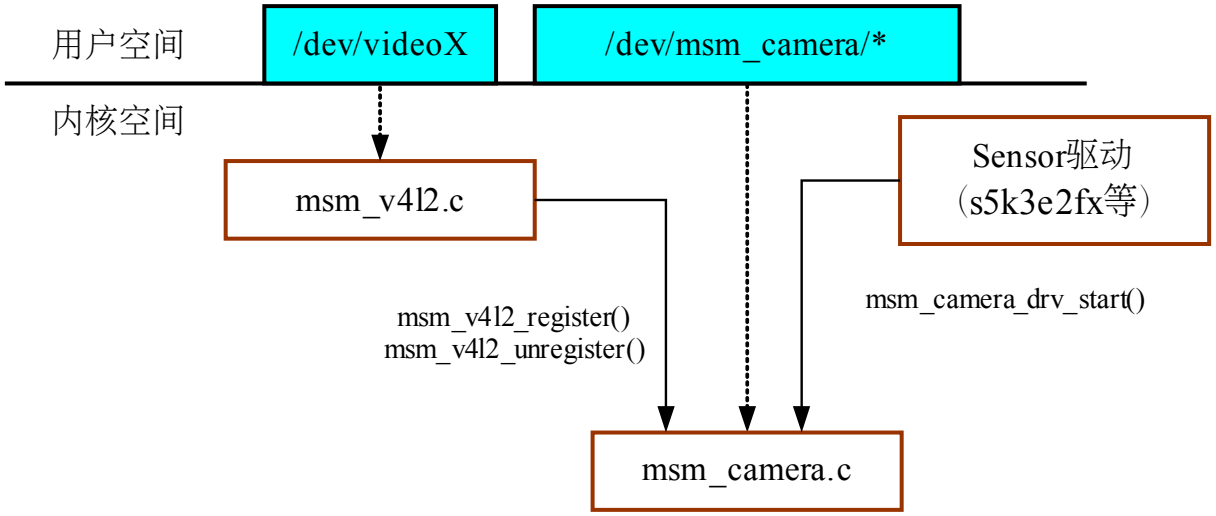
2.2 MSM 平台的 Camera 实现

MSM 的主要文件是在 drivers/media/video/msm/ 目录中。几个实现的文件的内容如下所示。

msm_v4l2.c： v4l2 驱动程序的入口文件

msm_camera.c： 公用的库函数

s5k3e2fx.c： 摄像头传感器驱动文件，使用 i2c 接口控制



2.2 MSM 平台的 Camera 实现

MSM 平台 Camera 的硬件抽象层已经包含在 Android 代码中，这部分内容路径为， hardware/msm7k/libcamera ， 其中包含了以下几个文件。

camera_ifc.h： Camera 接口中常量的定义

QualcommCameraHardware.h： 硬件抽象层的头文件

QualcommCameraHardware.cpp： 硬件抽象层的实现

2.3 OMAP 平台的 Camera 实现

OMAP 处理器内部包含了高级的 ISP （Image Signal Processing，图像信号处理）模块，通过外接（通常使用 i2c 的方式连接）的 Camera Sensor 即可以实现获取视频帧的功能。

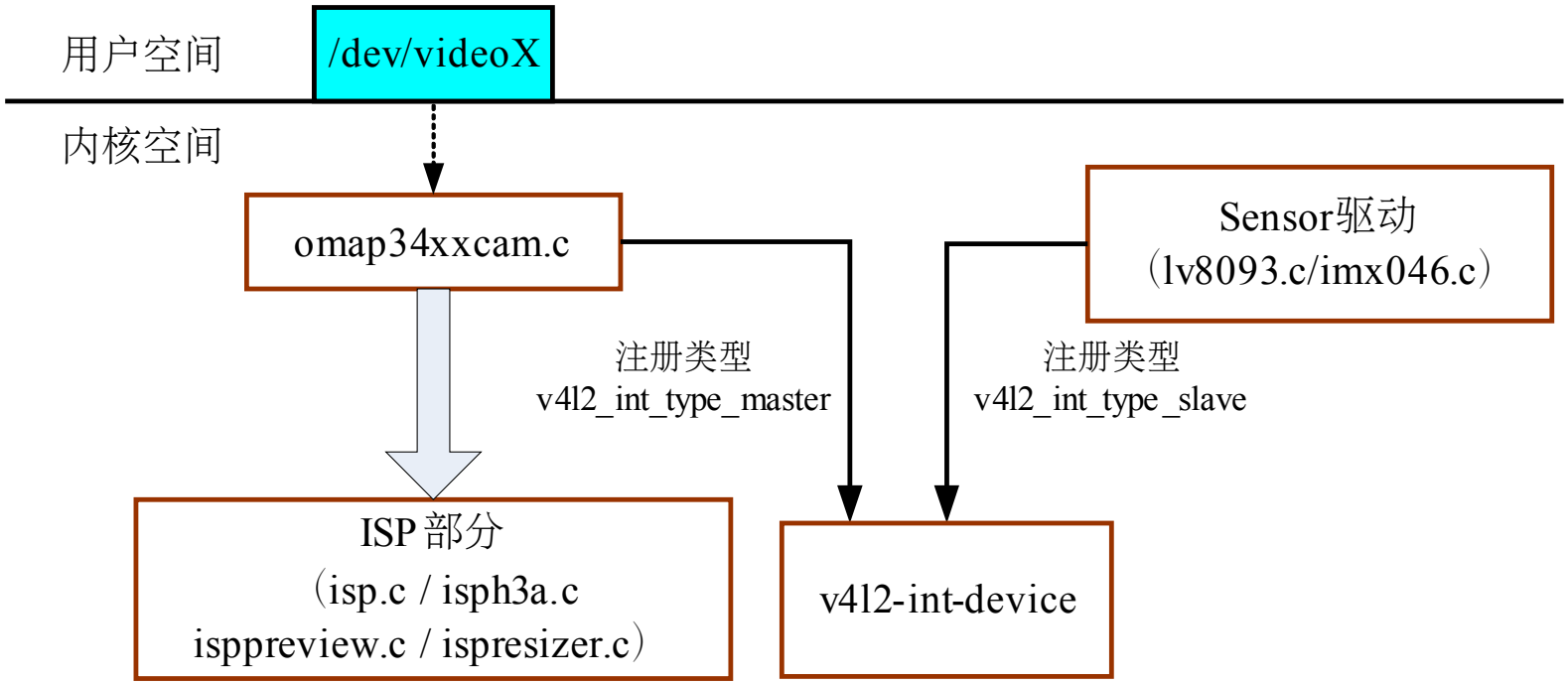
OMAP 平台的 Camera 部分驱动程序在 drivers/media/video/ 中，主要由以下三个部分组成：

Video for Linux 2 设备： omap34xxcam.h 和 omap34xxcam.c 文件

ISP 部分： isp 目录中的 isp.c ，isph3a.c ， isppreview.c ， ispresizer.c ，提供通过 ISP 进行的 3A 、预览、改变尺寸等功能

Camera Sensor 驱动： lv8093.c 或 imx046.c ，使用 v4l2-int-device 的结构注册。

2.3 OMAP 平台的 Camera 实现



2.3 OMAP 平台的 Camera 实现

OMAP 平台的 Camera 硬件抽象层基于 OMAP 的 V4L2 驱动程序实现，同时调用了 Overlay 系统作为视频输出。因此，其硬件抽象层的 useOverlay() 返回值为 true 。为了提高性能，直接映射 Overlay 中的内存，作为 Camera 输出的内存。因此，在 OMAP 的 Camera 硬件抽象层调用 V4L2 驱动程序的时候，使用 V4L2_MEMORY_USERPTR 标识，表示来自用户空间的内存。

OMAP 平台的 Camera 硬件抽象层还可以使用增强型功能。例如， 3A （自动对焦 AutoFocus 、自动增强 AutoEnhance 、自动白平衡 AutoWhiteBalance ）。 3A 功能由 OMAP SOC 内部的 ISP 模块提供基本机制，算法部分由用户空间的库支持。

谢谢！