# 初始化

## 客户端

## FW

\frameworks\av\services\camera\libcameraservice\device1\CameraHardwareInterface.h

mPreviewWindow和mHalPreviewWindow的区别和联系：mHalPreviewWindow的user字段就是mPreviewWindow！

**static** ANativeWindow \*\_\_to\_anw(**void** \*user)  
 {  
 CameraHardwareInterface \*\_\_this =  
 **reinterpret\_cast**<CameraHardwareInterface \*>(user);  
 **return** \_\_this->mPreviewWindow.get();  
 }  
#define anw(n) \_\_to\_anw(((**struct** camera\_preview\_window \*)n)->user)

**void** initHalPreviewWindow()  
{  
 mHalPreviewWindow.nw.cancel\_buffer = \_\_cancel\_buffer;  
 mHalPreviewWindow.nw.lock\_buffer = \_\_lock\_buffer;  
 mHalPreviewWindow.nw.dequeue\_buffer = \_\_dequeue\_buffer;  
 mHalPreviewWindow.nw.enqueue\_buffer = \_\_enqueue\_buffer;  
 mHalPreviewWindow.nw.set\_buffer\_count = \_\_set\_buffer\_count;  
 mHalPreviewWindow.nw.set\_buffers\_geometry = \_\_set\_buffers\_geometry;  
 mHalPreviewWindow.nw.set\_crop = \_\_set\_crop;  
 mHalPreviewWindow.nw.set\_timestamp = \_\_set\_timestamp;  
 mHalPreviewWindow.nw.set\_usage = \_\_set\_usage;  
 mHalPreviewWindow.nw.set\_swap\_interval = \_\_set\_swap\_interval;  
  
 mHalPreviewWindow.nw.get\_min\_undequeued\_buffer\_count =  
 \_\_get\_min\_undequeued\_buffer\_count;  
}

**struct** camera\_preview\_window {  
 **struct** preview\_stream\_ops nw;  
 **void** \*user;  
};

status\_t setPreviewWindow(**const** sp<ANativeWindow>& buf)  
{  
 ALOGV(**"%s(%s) buf %p"**, \_\_FUNCTION\_\_, mName.string(), buf.get());  
  
 mPreviewWindow = buf;  
 **if** (buf != **nullptr**) {  
 **if** (mPreviewScalingMode != NOT\_SET) {  
 setPreviewScalingMode(mPreviewScalingMode);  
 }  
 **if** (mPreviewTransform != NOT\_SET) {  
 setPreviewTransform(mPreviewTransform);  
 }  
 }  
 mHalPreviewWindow.user = **this**;  
 ALOGV(**"%s &mHalPreviewWindow %p mHalPreviewWindow.user %p"**, \_\_FUNCTION\_\_,  
 &mHalPreviewWindow, mHalPreviewWindow.user);  
 **return** mDevice->ops->set\_preview\_window(mDevice,  
 buf.get() ? &mHalPreviewWindow.nw : 0);  
  
}

## ANativeWindow来源

在mANativeWindow的类型为preview\_stream\_ops\_t，定义在

hardware/libhardware/include/hardware/camera.h

具体定义：

**int** (\*enqueue\_buffer)(**struct** preview\_stream\_ops\* w,  
 buffer\_handle\_t\* buffer);

初始化过程

camera\_device\_open

CameraHal\_Module.cpp

### set\_preview\_window

**int** camera\_device\_open(**const** hw\_module\_t\* module, **const char**\* name,  
 hw\_device\_t\*\* device)  
{

camera\_ops->set\_preview\_window = camera\_set\_preview\_window;

}

### camera\_set\_preview\_window

CameraHal\_Module.cpp

**int** camera\_set\_preview\_window(**struct** camera\_device \* device,  
 **struct** preview\_stream\_ops \*window)  
{  
 rv = gCameraHals[rk\_dev->cameraid]->setPreviewWindow(window);  
 **return** rv;  
}

gCameraHals

#### 发送消息

**int** CameraHal::setPreviewWindow(**struct** preview\_stream\_ops \*window)  
{  
 LOG\_FUNCTION\_NAME   
 Message\_cam msg;   
 Semaphore sem;  
 Mutex::Autolock lock(mLock);  
 **if** ((mCommandThread != NULL)) {  
 msg.command = CMD\_SET\_PREVIEW\_WINDOW;  
 sem.Create();  
 msg.arg1 = (**void**\*)(&sem);  
 msg.arg2 = (**void**\*)window;  
 setCamStatus(CMD\_SET\_PREVIEW\_WINDOW\_PREPARE, 1);  
 commandThreadCommandQ.put(&msg);  
 **if**(msg.arg1){  
 sem.Wait();  
 }  
 **if**(mCameraStatus&CMD\_SET\_PREVIEW\_WINDOW\_DONE)  
 LOG1(**"set preview window OK."**);   
 }  
 LOG\_FUNCTION\_NAME\_EXIT  
 **return** 0;  
}

#### 处理消息

**void** CameraHal::commandThread()

**case** CMD\_SET\_PREVIEW\_WINDOW:  
 {  
 LOGD(**"%s(%d):receive CMD\_SET\_PREVIEW\_WINDOW"**,\_\_FUNCTION\_\_,\_\_LINE\_\_);  
 mParameters.getPreviewSize(&app\_previw\_w,&app\_preview\_h);  
 mDisplayAdapter->setPreviewWindow((**struct** preview\_stream\_ops \*)msg.arg2);  
 prevStatus = mCameraAdapter->getCurPreviewState(&drv\_w,&drv\_h);  
  
 **if** ((mDisplayAdapter->getPreviewWindow()) && prevStatus) {  
 err=mDisplayAdapter->startDisplay(app\_previw\_w, app\_preview\_h);  
 **if**(err != -1)  
 setCamStatus(STA\_DISPLAY\_RUNNING, 1);  
 } **else** {  
 LOG1(**"%s(%d): not start display now"**,\_\_FUNCTION\_\_,\_\_LINE\_\_);  
 }  
  
setCamStatus(CMD\_SET\_PREVIEW\_WINDOW\_DONE, 1);   
 **if**(msg.arg1)  
 ((Semaphore\*)(msg.arg1))->Signal();  
 LOGD(**"%s(%d): CMD\_SET\_PREVIEW\_WINDOW out"**,\_\_FUNCTION\_\_,\_\_LINE\_\_);  
 **break**;  
 }

## DisplayAdapter.setPreviewWindow()

最后终于调用了

**int** DisplayAdapter::setPreviewWindow(**struct** preview\_stream\_ops\* window)  
{  
 **if**(mANativeWindow){  
 pauseDisplay();  
 }  
 mANativeWindow = window;  
 **return** 0;  
}

# enqueue\_buffer时序

从hal层请求enqueue\_buffer！

**static int** \_\_enqueue\_buffer(**struct** preview\_stream\_ops\* w,  
 buffer\_handle\_t\* buffer)  
{  
 ANativeWindow \*a = anw(w);  
 **return** a->queueBuffer(a,  
 container\_of(buffer, ANativeWindowBuffer, handle), -1);  
}

# displayThreadCommandQ.get(&msg);

取出一帧，填充数据信息

FramInfo\_s\* frame = (FramInfo\_s\*)msg.arg2;  
frame\_used\_flag = (**long**)msg.arg3;  
  
queue\_buf\_index = (**long**)msg.arg1;   
queue\_display\_index = CONFIG\_CAMERA\_DISPLAY\_BUF\_CNT;

# 找出queue\_display\_index

queue\_display\_index = CONFIG\_CAMERA\_DISPLAY\_BUF\_CNT;  
*//get a free buffer* **for** (i=0; i<CONFIG\_CAMERA\_DISPLAY\_BUF\_CNT; i++) {  
 **if** (mDisplayBufInfo && mDisplayBufInfo[i].buf\_state == 0)  
 **break**;  
 }  
 **if** (i<CONFIG\_CAMERA\_DISPLAY\_BUF\_CNT) {  
 queue\_display\_index = i;  
 }

# 格式转化

# setBufferState

标记为可用状态

setBufferState(queue\_display\_index, 1);  
mapper.unlock((buffer\_handle\_t)mDisplayBufInfo[queue\_display\_index].priv\_hnd);

# 送入窗口

err = mANativeWindow->enqueue\_buffer(mANativeWindow, (buffer\_handle\_t\*)mDisplayBufInfo[queue\_display\_index].buffer\_hnd);

# returnFrame

为啥要还？、

**if**(mFrameProvider)  
 mFrameProvider->returnFrame(frame->frame\_index,frame\_used\_flag);

# 取出一个显示buffer

queue\_cnt = 0;  
 **for** (i=0; i<mDislayBufNum; i++) {  
 **if** (mDisplayBufInfo[i].buf\_state == 1)   
 queue\_cnt++;// 统计目前用了多少个buffer  
 }  
  
 **if** (queue\_cnt > mDispBufUndqueueMin) {//高于了最小空闲  
err = mANativeWindow->dequeue\_buffer(mANativeWindow, (buffer\_handle\_t\*\*)&hnd, &stride);  
**if** (err == 0) {   
 *// lock the initial queueable buffers* bounds.left = 0;  
 bounds.top = 0;  
 bounds.right = mDisplayWidth;  
 bounds.bottom = mDisplayHeight;  
 mANativeWindow->lock\_buffer(mANativeWindow, (buffer\_handle\_t\*)hnd);  
 mapper.lock((buffer\_handle\_t)(\*hnd), CAMHAL\_GRALLOC\_USAGE, bounds, y\_uv);  
  
 phnd = (NATIVE\_HANDLE\_TYPE\*)\*hnd;  
 **for** (i=0; i<mDislayBufNum; i++) {  
 **if** (phnd == mDisplayBufInfo[i].priv\_hnd) {  
 dequeue\_buf\_index = i;  
 **break**;  
 }  
 }  
   
 **if** (i >= mDislayBufNum) {   
 LOGE(**"%s(%d): dequeue buffer(0x%x ) don't find in mDisplayBufferMap"**, \_\_FUNCTION\_\_,\_\_LINE\_\_,(**long**)phnd);   
 **continue**;  
 } **else** {  
 setBufferState(dequeue\_buf\_index, 0);//标记为空闲buffer  
 }  
   
 }