# 1 notify的种类？？

mm\_camera\_interface.h

## 1.1mm\_camera\_event\_notify\_t:

function definition for event notify handling

1.1原型：

**typedef** **void** (\*mm\_camera\_event\_notify\_t)(uint32\_t camera\_handle,mm\_camera\_event\_t \*evt,**void** \*user\_data);

## 1.2 mm\_camera\_buf\_notify\_t:

**function definition for frame notify handling** @mm\_camera\_super\_buf\_t : received frame buffers

**2.1原型(声明)：mm\_camera\_interface.h中声明**

**typedef** **void** (\*mm\_camera\_buf\_notify\_t) (mm\_camera\_super\_buf\_t \*bufs,**void** \*user\_data);

**2.2实例(定义)：QCamera中定义**

zsl\_channel\_cb、capture\_channel\_cb\_routine // Only zsl and capture channel need callback

DataNotifyCB  // QcameraStream中定义

**2.3使用（注册）：mm\_camera、mm\_channel和mm\_stream中注册**

addxxxChannel::pChannel->init::~::mm\_channel\_init(注册到mm\_channel中去)

my\_obj->bundle.super\_buf\_notify\_cb = channel\_cb;

addxxxChannel::addStreamToChannel::pStream->init::~::mm\_camera\_config（add\_stream后config\_stream时注册到mm\_stream中的DataNotifyCB：）

my\_obj->buf\_cb[0].cb = config->stream\_cb

**typedef** **struct** mm\_stream {

    // dataCB registered on this stream obj

mm\_stream\_data\_cb\_t buf\_cb[MM\_CAMERA\_STREAM\_BUF\_CB\_MAX];

};

**typedef** **struct** {

    mm\_camera\_buf\_notify\_t cb;

**void** \*user\_data;

} mm\_stream\_data\_cb\_t;

namespace qcamera

## 1.3 stream\_cb\_routine

**3.1原型：**

**typedef** **void** (\*stream\_cb\_routine)(mm\_camera\_super\_buf\_t \*frame,QCameraStream \*stream,**void** \*userdata);

**3.2实例：**

metadata\_stream\_cb\_routine、preview\_stream\_cb\_routine/nodisplay\_preview\_stream\_cb\_routine、*preview\_raw\_stream\_cb\_routine*；

video\_stream\_cb\_routine；snapshot\_stream\_cb\_routine；raw\_stream\_cb\_routine

**3.3使用：QCamera(Stream)中注册**

addStreamToChannel::pStream->init::注册到QCameraStream中

mDataCB = stream\_cb;

附：channel\_cb\_routine和stream\_cb\_routine在Channel中存在的情况

/表示或，<>表示可能不存在

​**MetadataChannel：**metadata\_stream\_cb\_routine、raw\_stream\_cb\_routine

**PreviewChannel：**

metadata\_stream\_cb\_routine、preview\_stream\_cb\_routine/nodisplay\_preview\_stream\_cb\_routine、*<preview\_raw\_stream\_cb\_routine>*；

**VideoChannel**：video\_stream\_cb\_routine；

**SnapshotChannel：**snapshot\_stream\_cb\_routine；

**ZSLChannel：**metadata\_stream\_cb\_routine、nodisplay\_preview\_stream\_cb\_routine/preview\_stream\_cb\_routine、zsl\_channel\_cb

**CaptureChannel:**metadata\_stream\_cb\_routine、*<preview\_stream\_cb\_routine>、<snapshot\_raw\_stream\_cb\_routine>、*capture\_channel\_cb\_routine

## 1.4 mm\_camera\_poll\_notify\_t

function ptr defined for poll notify CB,registered at poll thread with poll fd

**4.1原型（声明）：mm\_camera.h**

**typedef** **void** (\*mm\_camera\_poll\_notify\_t)(**void** \*user\_data);

4.2实例（定义）：

mm\_camera\_stream.c

mm\_camera.c

**static** **void** **mm\_camera\_event\_notify**(**void**\* user\_data)

\* DESCRIPTION: callback to handle event notify from kernel. This call will dequeue event from kernel.

\* PARAMETERS : @user\_data: user data ptr (camera object)

**static** **void** **mm\_stream\_data\_notify**(**void**\* user\_data);

\* callback to handle data notify from kernel

\* PARAMETERS : @user\_data : user data ptr (stream object)

4.3注册：

openCamera时，mm\_camera\_open：：mm\_camera\_evt\_sub：：mm\_camera\_poll\_thread\_add\_poll\_fd(,mm\_stream\_data\_notify,)

startChannel时，mm\_stream\_streamon：：mm\_camera\_poll\_thread\_add\_poll\_fd(,mm\_stream\_data\_notify,)

int32\_t mm\_camera\_poll\_thread\_add\_poll\_fd(,mm\_camera\_poll\_notify\_t notify\_cb,){

    poll\_cb->poll\_entries[idx].notify\_cb = notify\_cb;

}

调用：

Cmd\_thread\_launch中传递的dispatch函数原型？

Read code 函数实现？各dispatch，notify等之间的关系？？

貌似还少了HAL框图 QcameraStream 中的CBThread、Qcamera(Stream??)中的NotifyTh没找到对应关系

给线程标题具体化

# 2 Thread 类型

一般情况下addChannel的时候去创建notify，startChannel的时候创建相应thread。

Poll thread是个例外，也可以在add时起来，且start的时候去创建notify（start的时向其中添加fd，创建并添加相应的notify）

## 2.1 mm\_camera

### 2.1.1 注册mm\_camera\_event\_notify\_t(EvtHandle)

**openCamera：：mm\_camera\_register\_event\_notify\_internal**

QcameraHWI中调用register\_envent\_notify最终调用到mm\_camera.c中的上述函数

mCameraHandle->ops->register\_event\_notify(mCameraHandle->camera\_handle,

camEvtHandle,

(void \*) this);

### 2.1.2 创建线程

**openCamera：：mm\_camera\_open**（mm\_camera\_obj\_t \*my\_obj）

CDBG("%s : Launch evt Thread in Cam Open",\_\_func\_\_);

mm\_camera\_cmd\_thread\_launch(&my\_obj->evt\_thread,

mm\_camera\_dispatch\_app\_event,

(void \*)my\_obj);

//

CDBG("%s : Launch evt Poll Thread in Cam Open", \_\_func\_\_);

mm\_camera\_poll\_thread\_launch(&my\_obj->evt\_poll\_thread,

MM\_CAMERA\_POLL\_TYPE\_EVT);

/\* we will add evt fd into event poll thread upon user first register for evt \*/

mm\_camera\_evt\_sub(my\_obj, TRUE); // 注册**mm\_camera\_event\_notify（**mm\_camera\_poll\_notify\_t**）**

## 2.2 mm\_channel

### 2.2.1 注册mm\_camera\_buf\_notify\_t(channel\_cb)

**addChannel::mm\_channel\_init**

my\_obj(mm\_channel)->bundle.super\_buf\_notify\_cb = channel\_cb

CDBG("%s : Launch data poll thread in channel open", \_\_func\_\_);

mm\_camera\_poll\_thread\_launch(&my\_obj->poll\_thread[0],

MM\_CAMERA\_POLL\_TYPE\_DATA);

### 2.2.2 创建线程

**startChannel::mm\_channel\_start**

/\* launch cb thread for dispatching super buf through cb \*/

mm\_camera\_cmd\_thread\_launch(&my\_obj->cb\_thread,

mm\_channel\_dispatch\_super\_buf,

(void\*)my\_obj);

/\* launch cmd thread for super buf dataCB \*/

mm\_camera\_cmd\_thread\_launch(&my\_obj->cmd\_thread,

mm\_channel\_process\_stream\_buf,

(void\*)my\_obj);

## 2.3 mm\_stream

### 2.3.1 注册mm\_camera\_buf\_notify\_t(stream\_cb)

**addChannel::addStreamToChannel::** **mm\_stream\_config**

**/\* cd through intf always palced at idx 0 of buf\_cb \*/ why 0?**

原因:idx 不是0的在stream start时向poll thread中添加fd时创建？？貌似不是，添加fd到poll threrad时创建的notify是mm\_camera\_poll\_notify\_t型

既然定义了一个数组，就可以创建多个notify，那么其它的在哪里被创建（赋值）？

my\_ob j (mm\_stream) ->buf\_cb[0].cb = config->stream\_cb; // mm\_camera\_buf\_notify\_t

### 2.3.2 创建线程

**startChannel ：：MM\_STREAM\_EVT\_START**

/\* launch cmd thread if CB is not null \*/

mm\_camera\_cmd\_thread\_launch(&my\_obj->cmd\_thread,

mm\_stream\_dispatch\_app\_data,

(void \*)my\_obj);

rc = mm\_stream\_streamon(my\_obj); // add fd into thread并且注册**mm\_stream\_data\_notify（**mm\_camera\_poll\_notify\_t**）**

## 2.4 QCameraStream

### 2.4.1 注册stream\_cb\_routine(Qcamera:Stream\_cb)

**addChannel::addStreamToChannel::pStream->init即QcameraStream init()**

mDataCB = stream\_cb; // stream\_cb\_routine

### 2.4.2 创建线程

**startChannel::mStreams[i]->start 即QcameraStream start()**

rc = mProcTh.launch(dataProcRoutine, this);