|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  |  | | --- | --- | --- | | **Site**  IVSH Camera Lab | | Software Specifications | | **Originator**  Yajun Liu | | Software Technical Note | | Sensor Bring Up概要总结文档 | | | | **Contents:**  **Keywords:** | | | | Distribution list |  | | | SW Engineers | 刘亚军/杨亚坤 | | | SW Leader | 徐涵 | | | \* = mandatory reader |  | | | | |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | | AUTHOR | | APPROVALS | | QUALITY | | |  |  | | LEVEL 1 | | LEVEL 2 | |  | | NAME | 刘亚军 | | 徐涵 | | 刘心鹏 | | N/A | | FUNCTION |  | |  | |  | | N/A | | DATE | 2018/08/27 | |  | |  | |  | | SIGNATURE | 刘亚军 | |  | |  | |  | | |

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Comment** |
| 0.1 | 30/08/2018 | 刘亚军 | 文档架构作成 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[References 5](#_Toc227402444)

[1–前言 6](#_Toc1570814136)

[1.1目的 6](#_Toc1713816366)

[1.2 阅读对象 6](#_Toc1949217535)

[2– 概述 7](#_Toc763963821)

[2.1 CAMERA原理 7](#_Toc762181270)

[2.1.1 CAMERA工作原理 7](#_Toc1935545941)

[2.1.2 自动对焦原理 7](#_Toc2003471213)

[2.1.3 OIS防抖原理 8](#_Toc676804229)

[2.1.4 EEPROM&&OTP烧录原理 10](#_Toc1610923060)

[2.1.5 PDAF对焦原理 10](#_Toc366723644)

[2.1.6 Flash LED 12](#_Toc1384286636)

[2.2 模组解析 12](#_Toc1661121014)

[2.2.1 camera组成 12](#_Toc1422708899)

[2.2.2 VCM组成 14](#_Toc428895074)

[2.2.3 模组SPEC和PIN定义 15](#_Toc1760839042)

[2.3 项目经验 16](#_Toc1217469748)

[3 – BRING UP模块实现 17](#_Toc1678178955)

[3.1 软件框架 17](#_Toc1180424843)

[3.1.1 Kernel框架 17](#_Toc133993881)

[3.1.2 Camera Server Overview 17](#_Toc953510369)

[3.1.3 MCT Overview 17](#_Toc626054650)

[3.1.4 Sensor Module框架 18](#_Toc1535096050)

[3.1.5 Sensor Module代码路径 18](#_Toc1645215510)

[3.2 BRING UP实现流程及分析 19](#_Toc1252106437)

[3.2.1 Sensor poring 19](#_Toc1989054163)

[3.2.1.1 代码流程 19](#_Toc70103342)

[3.2.1.2 Porting流程 20](#_Toc160927350)

[3.2.1.3 验证流程 27](#_Toc1856496636)

[3.2.2 Eeprom poring 28](#_Toc782530385)

[3.2.2.1 代码流程 28](#_Toc749380744)

[3.2.2.2 Porting流程 29](#_Toc2083899081)

[3.2.2.3 验证流程 35](#_Toc205860873)

[3.2.3 Actuator porting 36](#_Toc315713462)

[3.2.3.1 代码流程 36](#_Toc1885632968)

[3.2.3.2 Porting流程 36](#_Toc969824694)

[3.2.3.3 验证流程 41](#_Toc1077894732)

[3.2.4 OIS porting 42](#_Toc1673695261)

[3.2.4.1 代码流程 42](#_Toc825812260)

[3.2.4.2 Porting流程 42](#_Toc1754698962)

[3.2.4.3 验证流程 46](#_Toc1137134673)

[3.2.5 PDAF porting 47](#_Toc1192535904)

[3.2.5.1 代码流程 47](#_Toc991501950)

[3.2.5.2 Porting流程 47](#_Toc650772039)

[1）Sensor端配置 47](#_Toc467761155)

[2）Eeprom数据 48](#_Toc1420397025)

[3.2.5.3 验证流程 49](#_Toc264127433)

[3.2.6 FLASH porting 51](#_Toc1685230903)

[3.2.6.1 代码流程 51](#_Toc951092332)

[3.2.6.2 Porting流程 51](#_Toc1444552276)

[3.2.6.3 验证流程 51](#_Toc1819224784)

[4 – BRING UP经验分享 53](#_Toc1904602701)

[4.1 SENSOR PORTING ISSUE 53](#_Toc2070606926)

[4.1.1 I2C相关 53](#_Toc1206837187)

[4.1.1.1 Read id failed 53](#_Toc1402334563)

[4.1.1.2 驱动电压 53](#_Toc1175229715)

[4.1.1.3 Pin脚虚焊 54](#_Toc1048407702)

[4.1.1.4 寄存器存储类型 54](#_Toc1472437905)

[4.1.1.5 寄存器读写超时 55](#_Toc1336157066)

[4.1.2 FpsRange 55](#_Toc757420690)

[4.1.2.1 Sensor FPS配置 55](#_Toc107484642)

[4.1.3 MIPI 相关 55](#_Toc2085537810)

[4.1.3.1 CAMIF 报错 56](#_Toc693836123)

[4.1.3.2 VFE 溢出报错 56](#_Toc313345515)

[4.1.3.3 CSID相关 56](#_Toc253767625)

[4.1.3.4 Settle\_cnt相关问题 57](#_Toc431985443)

[4.1.3.5 预览画面横纹较多 57](#_Toc1283170210)

[4.1.3.6 只能出一帧数据 58](#_Toc1331662357)

[4.1.4黑屏Issue 59](#_Toc2105680705)

[4.1.4.1 Hardware issue 60](#_Toc2108982470)

[4.1.4.2 Reading register issue 60](#_Toc938877671)

[4.1.4.3 App issue 60](#_Toc1095331730)

[4.1.4.4 HAL issue 61](#_Toc1154034726)

[4.2 EEPROM PORTING ISSUE 62](#_Toc1930379622)

[4.3 ACTUATOR PORTING ISSUE 62](#_Toc1746103770)

[4.3.1 DAC读写 62](#_Toc1621795882)

[4.3.1.1 寄存器数据使能问题 62](#_Toc1203292999)

[4.4 OIS PORTING ISSUE 63](#_Toc2010231203)

[4.5 PDAF PORTING ISSUE 63](#_Toc1159543137)

[4.6 FLASH PORTING ISSUE 63](#_Toc6901683)

References

|  |  |  |
| --- | --- | --- |
| **camera** | | *Wikipedia：<https://en.wikipedia.org/wiki/Camera>* |
| **PDAF** | | *Qualcomm 80-NV302-2 B* |
| **OIS** | | *Qualcomm* 80-P9301-97 C |
| **SENSOR** | | *Qualcomm 80-NU323-2SC* |
| **ACTUATOR** | | *Qualcomm 80-NU323-2SC* |
| **EEPROM** | | *Qualcomm 80-NU323-2SC* |
| **FLASH** | | *Qualcomm* 80-P9301-97 C |
|  | |  |
|  | |  |
|  | |  |
|  |  | |

# 1–前言

## 1.1目的

该文档主要是总结Sensor bring up的相关经验，包括Sensor，Eeprom，Actuator，OIS，PDAF， Flash等模块，为后续快速Bring up做好铺垫。

## 1.2 阅读对象

Camera driver研发人员。

# 2– 概述

## 2.1 CAMERA原理

### **2.1.1 CAMERA工作原理**

光线通过镜头Lens进入摄像头内部，然后经过IR Filter过滤红外光，最后到达sensor（传感器），senor分为按照材质可以分为CMOS和CCD两种，可以将光学信号转换为电信号，再通过内部的ADC电路转换为数字信号，然后传输给DSP（如果有的话，如果没有则以DVP的方式传送数据到基带芯片baseband，此时的数据格式Raw Data，加工处理后，转换成RGB、YUV等格式输出。

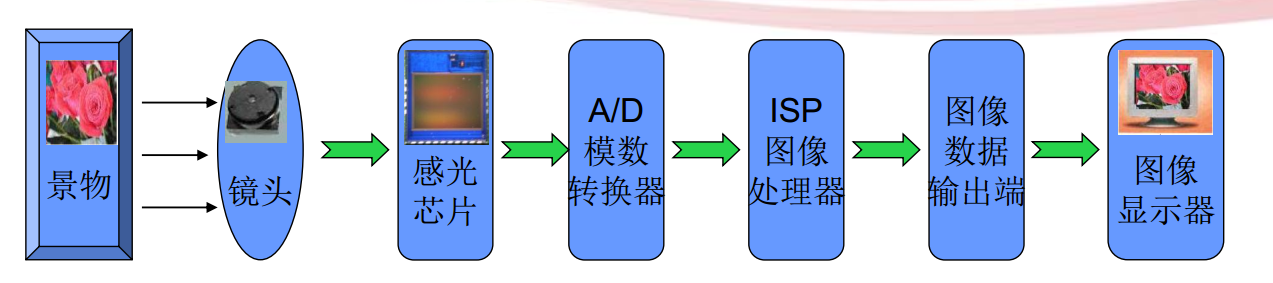


图2.1.1 camera原理图

### **2.1.2 自动对焦原理**

手机自动对焦功能是通过将摄像头锁入马达来实现的，马达种类较多，其中音圈马达是比较常用的，简称VCM（Voice Coil Motor），VCM和喇叭的工作原理一样，都是在固定磁场中加电流或电荷产生力的原理，从而产生运动的过程，即初中物理所谈左手定则。

左手定则：左手平展，让磁感线穿过手心，使大拇指与其余四指垂直，并且都跟手掌在一个平面内。把左手放入磁场中，让磁感线垂直穿入手心，手心面向N极(叉进点出)，四指指向电流所指方向，则大拇指的方向就是导体受力的方向。

手机摄像头的VCM需要Driver IC配合完成对焦，通过Driver IC控制VCM供电电流的大小，来确定VCM搭载的镜头移动的距离，从而调节到适当的位置拍摄清晰图像。

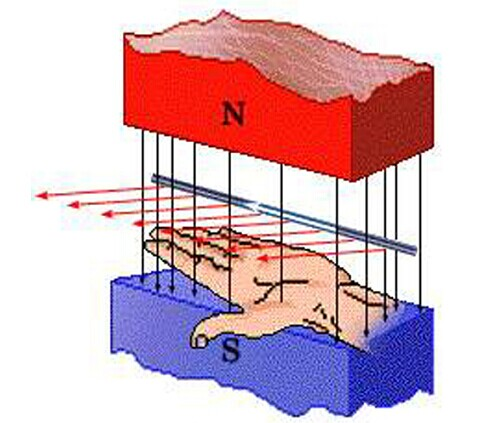


图2.1.2.a 左手定则

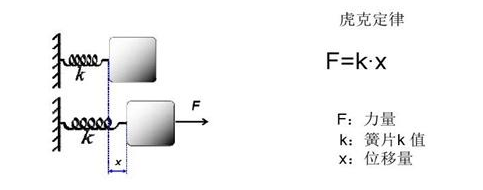


图2.1.2.b VCM原理图

### **2.1.3 OIS防抖原理**

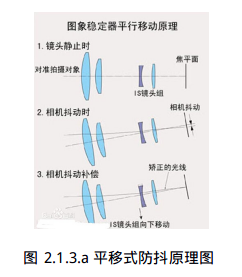
OIS的原理是：手持智能手机拍照时，手的抖动会造成相机的轻微倾斜（一般在+/-0.5度以内），该倾斜引起了镜头观察角度的变化，以镜头为参照物来说，相当于被拍摄的物体移动了，因此所成的像也会在图像传感器上相对于原位置发生偏移，结果造成图像始终随着手的抖动而处于不稳定状态。为解决这个问题，OIS技术应运而生。OIS是光学图像稳定系统的简称。它首先应包括一个可感测手抖的陀螺仪，该陀螺仪将手抖导致的相机倾斜角度测出，系统再根据该角度预测出倾斜导致的图像偏移量，然后系统控制镜头相对于图像传感器平移而产生相同大小但方向相反的图像偏移，由此将手抖造成的图像偏移抵消掉，保证相机在手抖环境中依然可保持成像稳定。

OIS目前有两种：平移式和移轴式防抖原理有所不同

当前智能手机OIS摄像模组有两个技术路线：分别为基于平移式（Pure shift）OIS对焦马达以及基于移轴式(Tilt-shift)对焦马达。前者以NokiaLumia 920的摄像模组为代表；后者以New HTC One的摄像模组为代表。不论是平移式还是移轴式，其基本原理都是一样的，即控制镜头相对于图像传感器平移而将手抖造成的图像偏移抵消补偿掉。但两者的具体实现方式有很大差别：

1）平移式OIS对焦马达是将一个独立的AF自动对焦马达固定在一个平移悬挂系统上（通常为4根等长的竖直向上的悬丝），该悬挂系统允许此AF马达在平行于图像传感器的平面上沿任意方向做微小平移（一般在正负100微米以内）。OIS马达的自动对焦功能由那个独立的AF马达来实现，而OIS功能则由AF马达连同镜头组成的镜头悬浮体相对于图像传感器平移而实现。由于镜头悬浮体的总重量较大，而悬挂系统的硬度又有限，因此，在不同的拍照姿态下（朝下、水平朝前、朝上等），镜头悬浮体会在自身重力的作用下相对于图像传感器产生平移，更严重的是，镜头悬浮体(相当于一个弹簧振子)会在外界扰动下产生平移振动，导致图像模糊！为了解决上述问题，马达内集成了两个霍尔传感元件，用于准确测出镜头悬浮体在两个正交平移方向上的位置，利用该位置信息结合闭环（Close-loop）控制便可将镜头悬浮体的位置锁定，令它只按照实际OIS的要求平移，而不受自身重力和拍照姿态的影响，也免于外界的机械扰动。

2）移轴式OIS马达的结构相对简单很多，非常接近普通AF马达。所不同的是，普通AF马达的弹簧悬挂系统只允许镜头沿对焦方向移动，而移轴式OIS马达的弹簧悬挂系统不但允许镜头沿对焦方向移动，还允许镜头相对于一个转轴支点做任意方向的微小侧倾（倾斜角度一般在正负1.5度以内）。由于转轴位于靠近镜头底部的位置，因此当镜头左右倾斜时，镜头顶部实际上在相对于图像传感器左右平移。利用这个平移，便可补偿手抖带来的图像抖动。由于一个固定的转轴支点的存在，马达运动部分的重量很大程度上被这个支点分担了，镜头的倾斜角度受拍照姿态的影响小到可以忽略，镜头也不易受外界机械干扰而摆动，镜头的倾斜角是稳定可预测的，完全受偏转电流大小的控制。因此，镜头顶部的平移量也是与马达所通的偏转电流成正比。移轴式OIS马达因此无需霍尔传感元件做闭环控制。另外，支点的存在还有一大好处，即施加较小的力便可令镜头倾斜，实际就是杠杆的省力原理。而平移式OIS马达需要克服整个镜头悬浮体的自身重量来实现镜头的平移，很费力。因此，移轴式OIS马达具有更低的理论功耗和更快速的机械响应，能补偿高达20Hz的快速抖动。

### **2.1.4 EEPROM&&OTP烧录原理**

EEPROM (Electrically Erasable Programmable read only memory)，带电可擦可编程只读存储器--一种掉电后数据不丢失的存储芯片。 EEPROM 可以在电脑上或专用设备上擦除已有信息，重新编程。

OTP(One Time Programmable)是MCU的一种存储器类型，意即一次性编程。多是采用融丝结构，编程过程是不可逆的破坏活动。

优缺点：相对于eeprom而言，OTP具有价格低廉的优势，这在竞争日益激烈的手机市场显得非常重要。而缺点不言而喻，一旦在OTP中烧录数据发生错误，就会导致整个手机模组报废，增大了生产成本。

进行OTP烧录的原因：模组中的Lens shading、AF等等参数由于各方面因素的影响，摄像头模组在各方面都存在一定的差异性，如果用同一套参数去校准，效果往往不尽人意。如果模组在出厂的时候，分别对每一个模组进行的校准，并且将这些校准参数烧入到OTP中，那么客户端在显示图像时只要从OTP中读取这些参数并且应用到图像上，他们得到的将是一致性非常好的成像效果。

### **2.1.5 PDAF对焦原理**

相位对焦——PDAF：它的全称是PhaseDetectionAutoFocus，字面意思就是“相位检测自动对焦”。通常而言相机的自动对焦方式有两种：反差对焦和相位对焦。反差对焦的原理是根据焦点处画面的对比度变化，寻找对比度最大时的镜头位置，也就是准确对焦的位置，如下图。

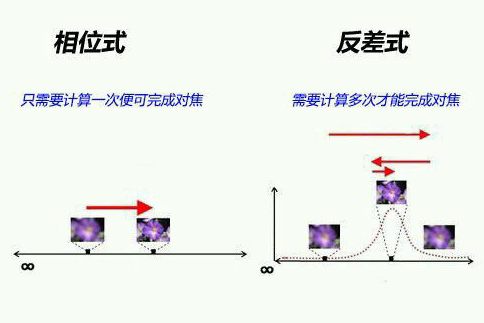


图2.1.5.a 相位式和反差式

相位对焦，它的原理是在感光元件上预留出一些遮蔽像素点，专门用来进行相位检测，通过提取校准区域的 L/R Shield Pixel 点用来生成当前的相位差(Phase Difference)。

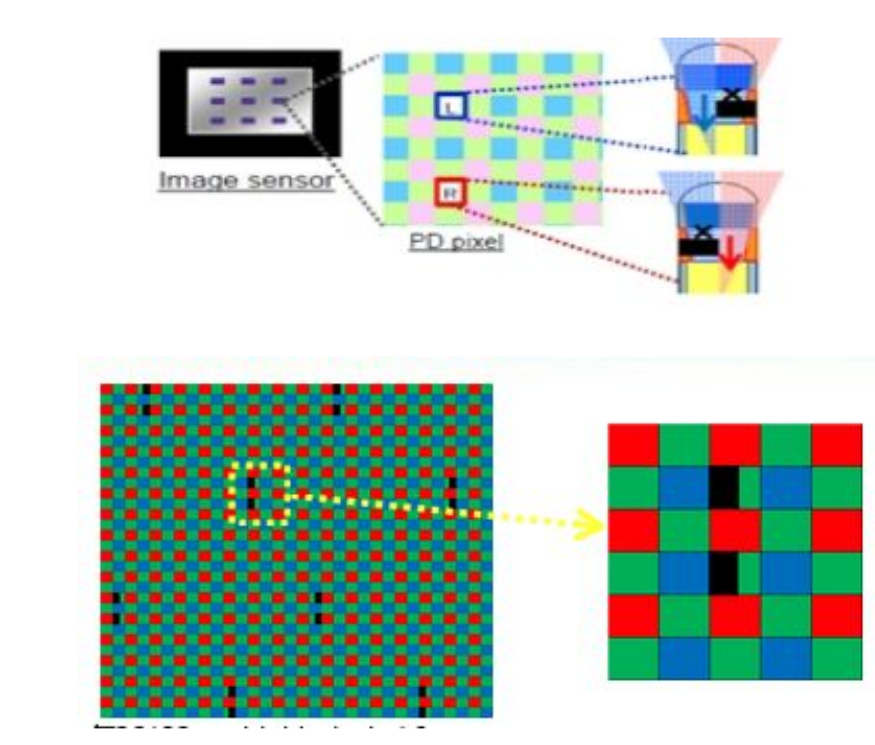


图2.1.5.b 像素点遮蔽

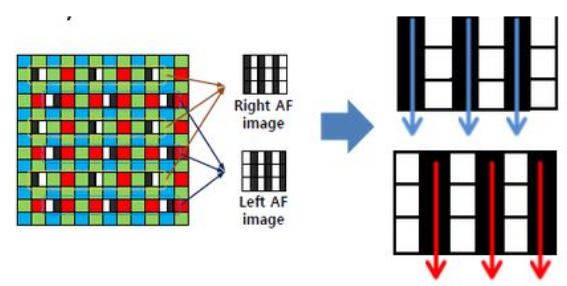


图2.1.5.c PD计算

PD 值有正负之分,用来决定马达的运动方向,其值决定马达的移动量。当 PD=0 时,表示此时图像是最清晰的。

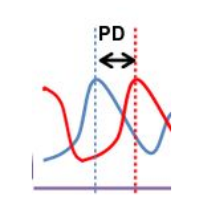


图2.1.5.d PD值

相比反差对焦，相位对焦不需要镜头的反复移动，对焦行程短了很多，对焦过程干净不犹豫。但另一方面，由于需要利用CMOS上的遮蔽像素点进行相位检测，故此相位对焦对光线强度的要求比较高。

相位式优点：

只需要计算一次就完成对焦，对焦速度极快，并且降低处理器计算负担。

相位式缺点：

在弱光环境下容易对不上焦。

### **2.1.6 Flash LED**

闪光灯是加强曝光量的方式之一，能在很短时间内发出很强的光线，多用于光线较暗的场合瞬

间照明，也用于光线较亮的场合给被拍摄对象局部补光，除此之外，还被用于一下的场景：

1：改变色温。

2：改善被摄体照明条件

3：反射用光，不破坏被摄体现场环境

4：添加眼神光

5：多灯组合，塑造被摄体形象

6：减小或加大反差

7：瞬间凝固被摄体

8：补光

一般相机上使用的灯是用惰性气体制成的灯，优点是亮度高，接近自然光，缺点是瞬间电压可以达到10kv，功率大，对电池及设备的损伤过大，综上考虑，手机上一般使用LED闪光灯/补光灯做增强曝光的方式。

## 2.2 模组解析

### **2.2.1 camera组成**

目前使用比较多的有定焦模组和自动变焦模组，它们的构成如下：

a：定焦模组

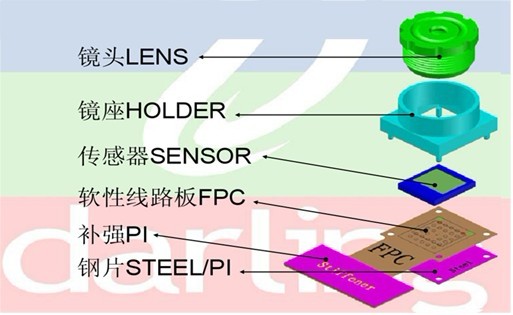


图2.2.1.a 定焦模组原理图

b：自动变焦模组

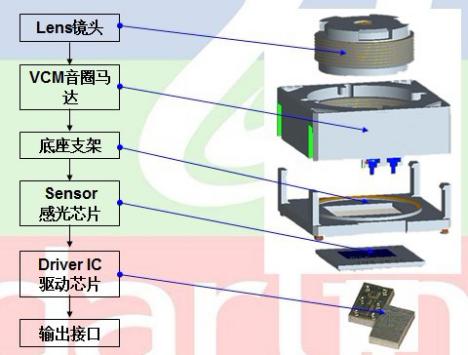


图2.2.1.b 自动变焦模组原理图

自动变焦模组和定焦模组最大的区别是多了一个对焦马达，各模块的介绍如下：

（1）镜头（lens）是相机的灵魂，镜头(lens)对成像的效果有很重要的作用，是利用透镜的折射原理，景物光线通过镜头，在聚焦平面上形成清晰的影像，通过感光材料CMOS或CCD感光器记录景物的影像。镜头厂家主要集中在台湾、日本和韩国，镜头这种光学技术含量高的产业有比较高的门槛，业内比较知名的企业如富士精机、柯尼卡美能达、大立光、Enplas等

（2）传感器（sensor）是CCM的核心模块，目前广泛使用的有两种：一种是广泛使用的CCD（电荷藕合）元件；另一种是CMOS（互补金属氧化物导体）器件。

A.电荷藕合器件图像传感器CCD（Charge Coupled Device），它使用一种高感光度的半导体材料制成，能把光线转变成电荷，通过模数转换器芯片转换成数字信号。CCD由许多感光单位组成，通常以百万像 素为单位。当CCD表面受到光线照射时，每个感光单位会将电荷反映在组件上，所有的感光单位所产生的信号加在一起，就构成了一幅完整的画面。CCD传感器模块以日本厂商为主导，全球规模市场有90%以上被日本厂商垄断，以索尼、松下、夏普为龙头。

B.互补性氧化金属半导体CMOS（Complementary Metal-Oxide Semiconductor）主要是利用硅和锗这两种元素所做成的半导体，使其在CMOS上共存着带N（带–电）和 P（带+电）级的半导体，这两个互补效应所产生的电流即可被处理芯片纪录和解读成影像。CMOS 传感器主要美国、台湾和韩国为主导，主要生产厂家有美国 OmniVision、Agilent、Micron,台湾的锐像、原相、泰视等，韩国的三星、现代。

（3）图像处理芯片（DSP）是CCM的重要组成部分，它的作用是将感光芯片获得的数据及时快速地传递中央处理器并刷新感光芯片，因此DSP芯片的好坏，直接影响画面品质（比如色彩饱和度，清晰度等）。

（4）FPC柔性电路板(柔性PCB): 简称"软板", 又称"柔性线路板",连接芯片和手机。起到电信号传输作用。

（5）手机自动对焦功能是通过将摄像头锁入马达来实现的，马达种类较多，其中音圈马达是比较常用的，简称VCM。

### **2.2.2 VCM组成**



图2.2.2 VCM模组

前弹片：用于承载载体，平衡力矩

磁铁：产生永久磁场

线圈：通电产生驱动力矩

载体：用于承载镜头

后弹簧：用于承载载体，平衡力矩

底座：用于固定部件。

### **2.2.3 模组SPEC和PIN定义**

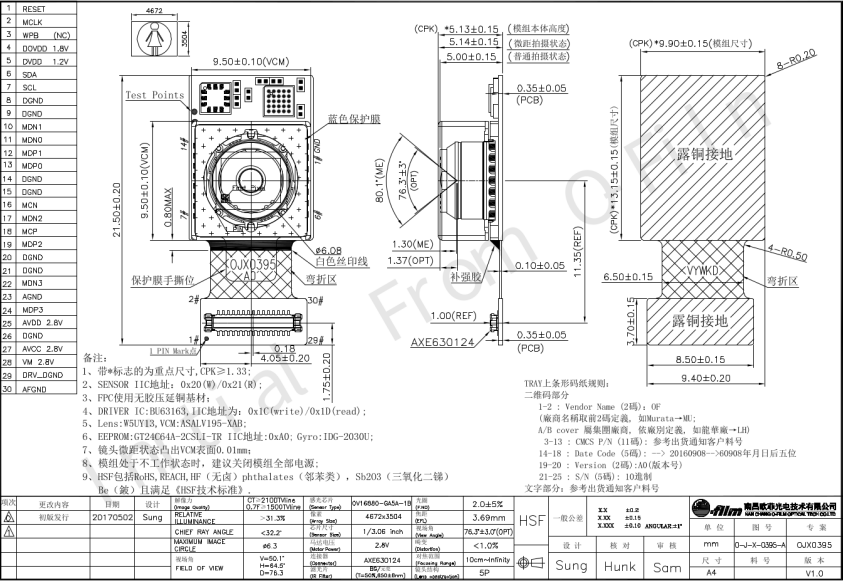


图2.2.3.a 模组SPEC

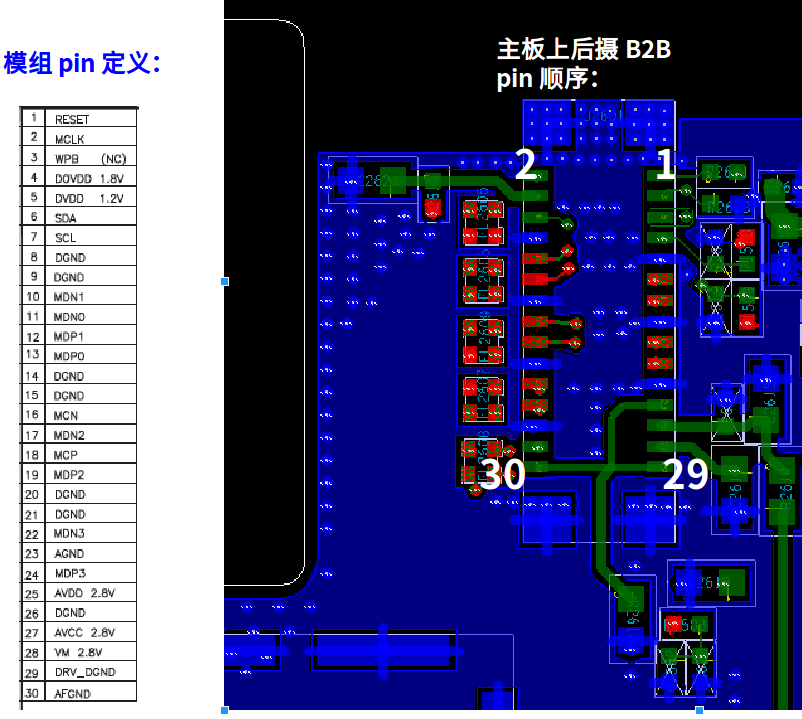


图2.2.3.b PCB图

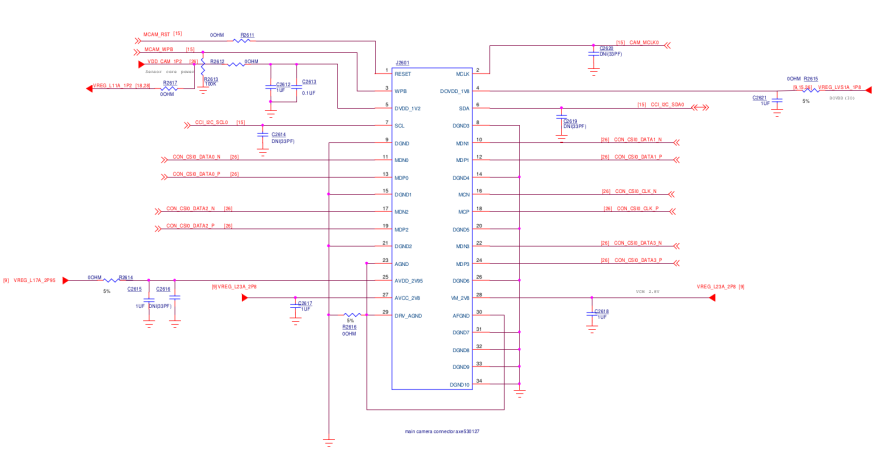


图2.2.3.c 原理图

。。。。

## 2.3 项目经验

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 项目名 | 平台 | 模组 | Sensor | Eeprom | Actuator | OIS | PDAF | FLASH |
| CI-164 | MSM8996 | OV16880 | OV16880 | ofilm\_ojx0395 | bu63163 | bu63163 |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

# 3 – BRING UP模块实现

## 3.1 软件框架

### **3.1.1 Kernel框架**



图3.1.1 Kernel框架

### **3.1.2 Camera Server Overview**

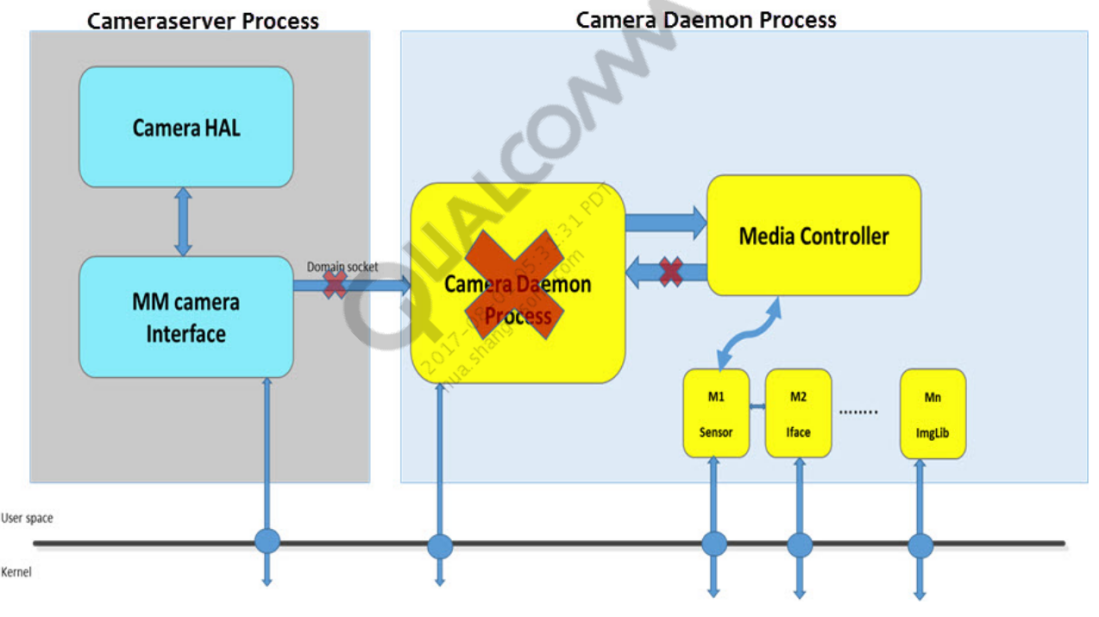


图3.1.2 Camera Server Overview

在8998平台里，camera daemon进程被取消了。

### **3.1.3 MCT Overview**

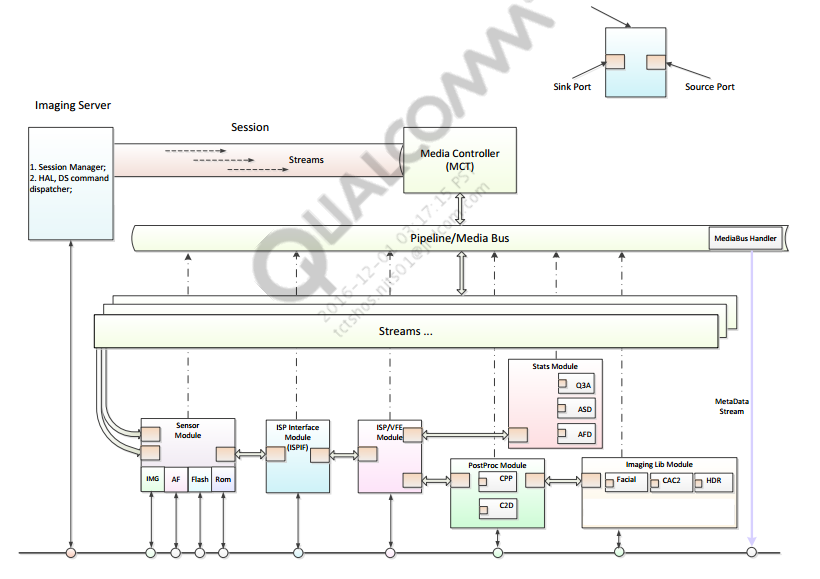


图3.1.3 MCT Overview框架

### **3.1.4 Sensor Module框架**

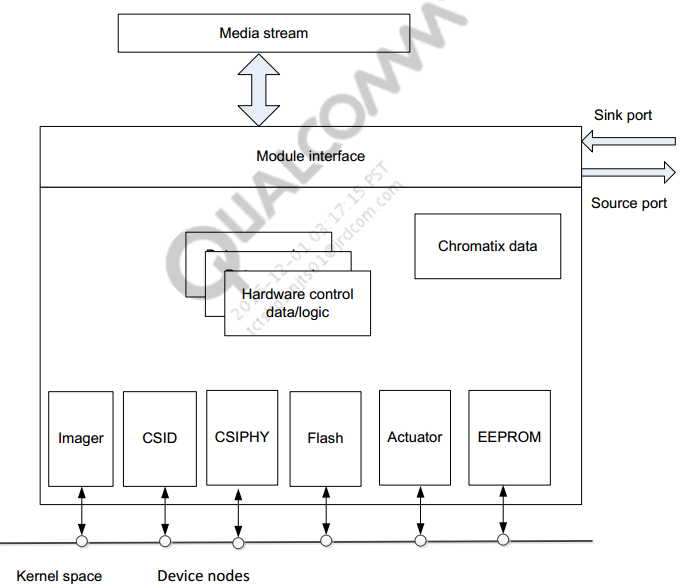


图3.1.4 Sensor Module框架

### **3.1.5 Sensor Module代码路径**

Sensor Module的代码路径如下，后面porting的大部分内容会在该目录下做修改：

vendor/qcom/proprietary/mmcamera/mm-camera2/media-controller/modules/sensors

## 3.2 BRING UP实现流程及分析

### **3.2.1 Sensor poring**

当一个项目确认后，我们需要让模组厂商提供sensor相关的资料，如：主板原理图、PCB图、模组SPEC、sensor datasheet等资料。同时我们需要确认该sensor是否通过PVL认证，如果有认证，和客户PVL的代码是否可以使用（使用PVL认证的代码会简化配置，但后期可能出现芯片批次不同引起的不能完全兼容或CTS测试时可能报错）。

#### 3.2.1.1 代码流程



图3.2.1.1.a dts信息流程

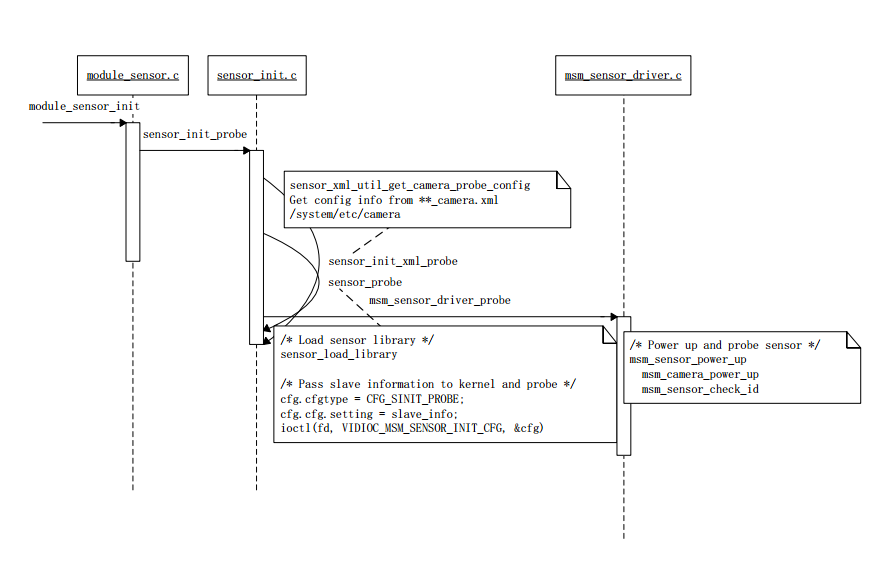


图3.2.1.1.b sensor&&XML\_probe流程

#### 3.2.1.2 Porting流程

我们需要根据厂商提供的模组信息，来正确的配置以下文件：

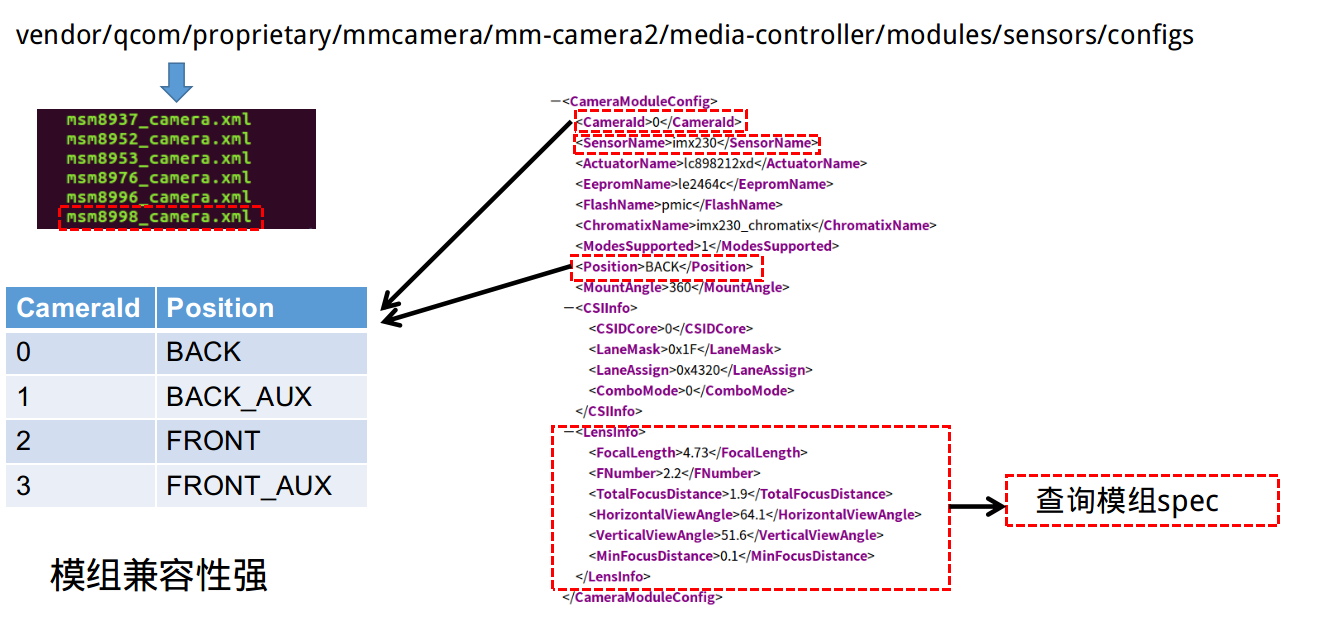
XML：.../mm-camera/mm-camera2/media-controller/modules/sensors/configs  
Sensor\_lib：.../mm-camera/mm-camera2/media-controller/modules/sensors/sensor/libs/

device-vendor.mk：vendor/qcom/proprietary/common/config/

Dtsi：kernel/arch/arm/boot/dts/qcom

具体配置：

**1）XML**



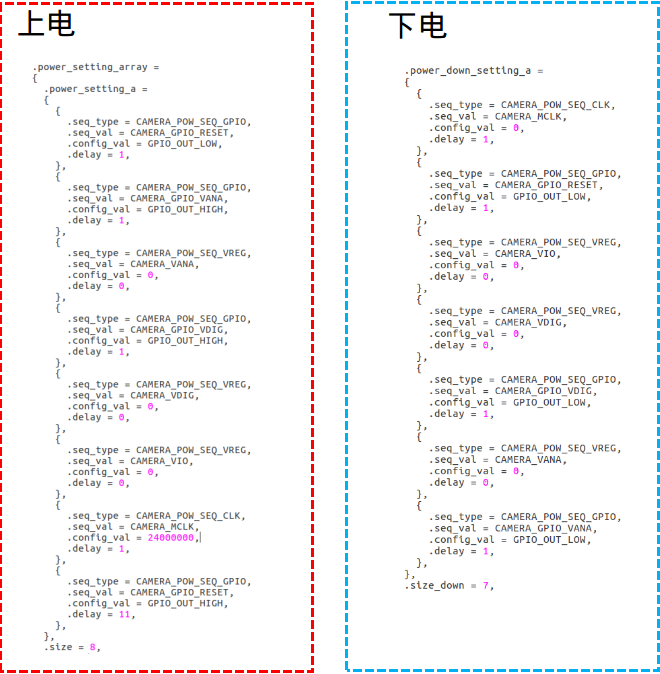
**2）编译文件**

在device-vendor.mk文件添加需要编译的sensor库，例如MM\_CAMERA += libmmcamera\_imx230

**3）lib库文件**

1. Init寄存器和resolution寄存器配置（厂商提供）
2. **上下电时序**

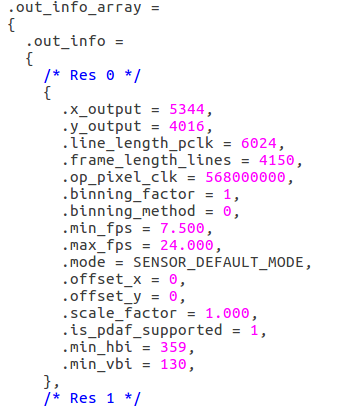
根据sensor datasheet里面的上电时序图配置power\_setting\_array。



上电配置

在更换新的模组时，可以优先使用原模组的上下电时序。

1. **Out info**



Out info配置

x\_output – 宽度

y\_output – 高度

line\_length\_pclk – 行间blanking的宽度

frame\_length\_lines – 帧间blanking的高度

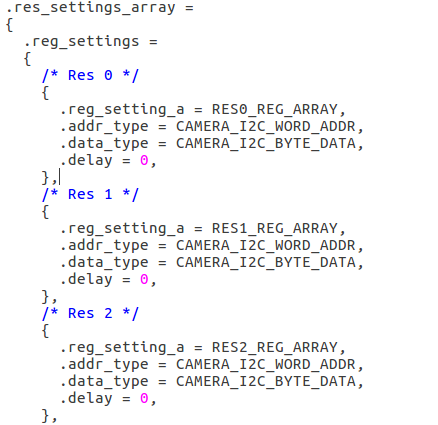
op\_pixel\_clk – 表示要设置 VFE 时钟

op\_pixel\_clk = (传感器总数据传输速率)/每个像素的位数

vt\_pixel\_clk时钟用于内部图像处理，计算曝光时间和帧率等，如下：

帧率：frame rate = vt\_pixel\_clk / (line\_lenth\_pclk \* frame\_length\_lines)。

1. **Resolution**



Res\_setting

根据配置的RES寄存器，正确的配置setting数组，后面会根据下发下来的stream\_info信息中包含的size来匹配对应的resolution（如果stream\_info有两种或以上的size，将会以其中最大的一组size去配置resolution，Res\_宽>size\_宽，Res\_长>size\_长）。

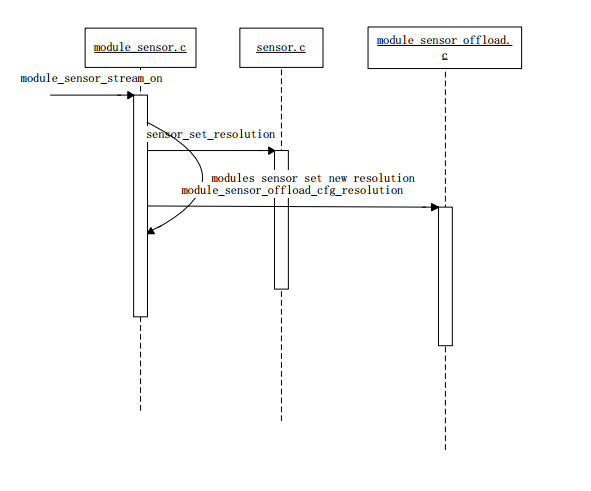
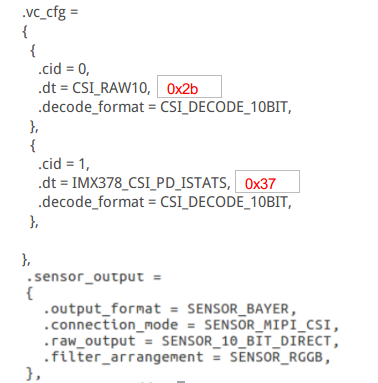
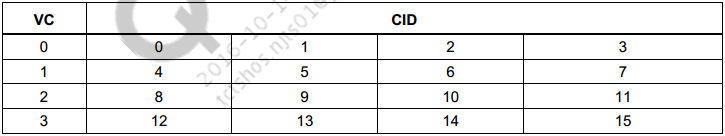


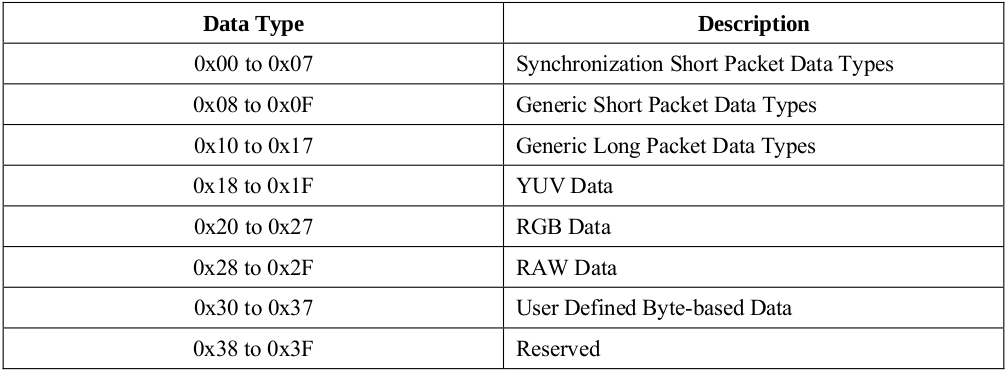
图3.2.1.2. Pick Resolution流程

1. **Sensor Stream Info**



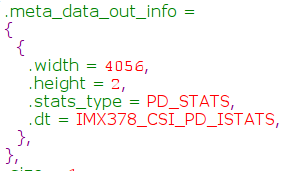
Stream 配置





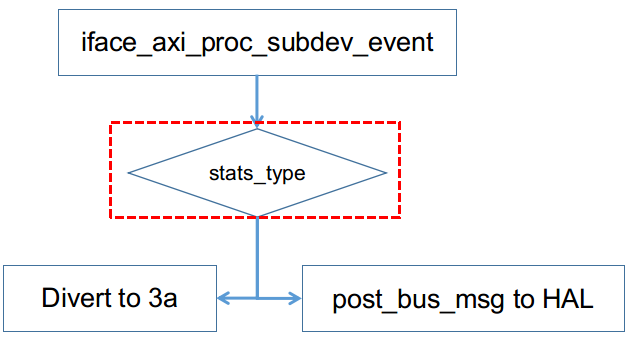
根据上面的表格以及sensor datasheet来配置VC信息，例如上面就是senser使用一个vc通道，传输了两种不同的数据，一组是raw数据，定义为0x2B，一组是自定义数据，定义为0x37。

1. **Meta\_Data**



Meta\_Data配置

配置的mata会作用在下面的流程判断中



1. **I2C**

I2C的配置信息用于寄存器读写，在sensor\_Probe过程中，会去做match\_id的动作，每一个sensor第一次failed后会再尝试2次，第3次失败后就会probe failed。在init过程或者设置模式时都会使用该I2C地址。



I2C配置

1. **MIPI配置**

这里配置的MIPI信息和xml配置的需要匹配。

Xml：

**<**CSIDCore**>**1**</**CSIDCore**>**

**<**LaneMask**>**0x1f**</**LaneMask**>**

**<**LaneAssign**>**0x4320**</**LaneAssign**>**

**<**ComboMode**>**0**</**ComboMode**>**

Sensor\_lib**:**

**.**csi\_params **=**

**{**

**.**lane\_cnt **=** 4**,**

**.**settle\_cnt **=** 0xb**,**

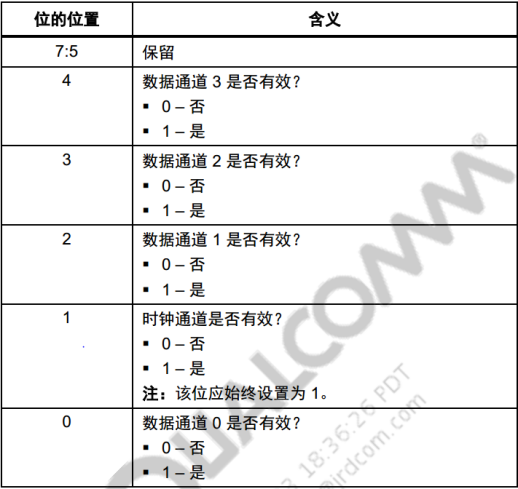
**.**is\_csi\_3phase **=** 0**,**

**},**

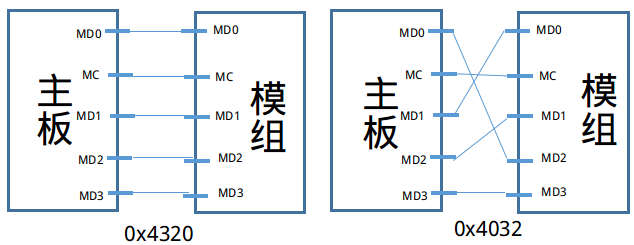
Lane\_cnt：使用的MIPI通道数量

Settle\_cnt：稳定计数

LaneMask按如下的配置：



LaneAssign按如下的配置：



1. **Kernel 配置**

qcom**,**camera@0 **{**

**......**

qcom**,**csiphy**-**sd**-**index **=** **<**0**>;**

qcom**,**csid**-**sd**-**index **=** **<**0**>;**

qcom**,**mount**-**angle **=** **<**270**>;**

qcom**,**led**-**flash**-**src **=** **<&**led\_flash0**>;**//LED索引

qcom**,**actuator**-**src **=** **<&**actuator0**>;**//actuator索引

qcom**,**eeprom**-**src **=** **<&**eeprom0**>;**//eeprom索引

cam\_vio**-**supply **=** **<&**pm8998\_lvs1**>;**//pin脚绑定

cam\_vana**-**supply **=** **<&**rear\_vana\_regulator**>;**//pin脚绑定

cam\_vdig**-**supply **=** **<&**pm8998\_s3**>;**//pin脚绑定

qcom**,**cam**-**vreg**-**name **=** "cam\_vio"**,** "cam\_vana"**,** "cam\_vdig"**;**//供电需要的

qcom**,**cam**-**vreg**-**min**-**voltage **=** **<**0 2800000 1352000**>;**//最小值

qcom**,**cam**-**vreg**-**max**-**voltage **=** **<**0 2800000 1352000**>;**//最大值

qcom**,**cam**-**vreg**-**op**-**mode **=** **<**0 80000 105000**>;**

qcom**,**gpio**-**no**-**mux **=** **<**0**>;**//索引表

pinctrl**-**names **=** "cam\_default"**,** "cam\_suspend"**;**//pin名称

pinctrl**-**0 **=** **<&**cam\_sensor\_mclk0\_active **&**cam\_sensor\_rear\_active**>;**//pin控制

pinctrl**-**1 **=** **<&**cam\_sensor\_mclk0\_suspend **&**cam\_sensor\_rear\_suspend**>;**//pin控制

gpios **=** **<&**tlmm 13 0**>,**//使用的gpio

**<&**tlmm 30 0**>,**//使用的gpio

**<&**pm8998\_gpios 20 0**>;**//使用的gpio

qcom**,**gpio**-**reset **=** **<**1**>;**//索引的gpio（30）

qcom**,**gpio**-**vdig **=** **<**2**>;**//索引的gpio（20）

qcom**,**gpio**-**req**-**tbl**-**num **=** **<**0 1 2**>;**

qcom**,**gpio**-**req**-**tbl**-**flags **=** **<**1 0 0**>;**

qcom**,**gpio**-**req**-**tbl**-**label **=** "CAMIF\_MCLK0"**,**"CAM\_RESET0"**,**"CAM\_VDIG"**;**//gpio对应的TYPE

qcom**,**sensor**-**position **=** **<**0**>;**

qcom**,**sensor**-**mode **=** **<**0**>;**

qcom**,**cci**-**master **=** **<**0**>;**

status **=** "ok"**;**

/\*CLK设置\*/

clocks **=** **<&**clock\_mmss clk\_mclk0\_clk\_src**>,**

**<&**clock\_mmss clk\_mmss\_camss\_mclk0\_clk**>;**

clock**-**names **=** "cam\_src\_clk"**,** "cam\_clk"**;**

qcom**,**clock**-**rates **=** **<**24000000 0**>;**

**};**

CAM\_VANA – 电源电压(模拟)

CAM\_VDIG – 电源电压(数字)

CAM\_VAF – 电源电压(致动器电压)

CAM\_VIO – 输入/输出电压(数字)

按照需求，对pin脚、电压、clk进行配置。

#### 3.2.1.3 验证流程

1. 启动设备后，通过adb命令进入设备，在sys/class/video4linux目录下，通过cat \*/name 获取节点名称，查看配置的sensor是否挂载。

2. 打开camera APP，看预览是否正常，能否正常获取图片。

3. 如果未挂载或不能正常打开camera，需要对硬件信息、dts配置、上电时序、I2C地址、GPIO的配置等等进行排查，参见第[4.1](#_4.1 SENSOR PORTING ISSUE)章节。

### **3.2.2 Eeprom poring**

Eeprom 的加载分为内核加载和用户空间加载，通过userspace\_probe这个变量来控制是内核加载还是用户空间加载。

在porting前，需要准备EEPROM Programming Guide、OTP data、模组spec等资料。

#### 3.2.2.1 代码流程

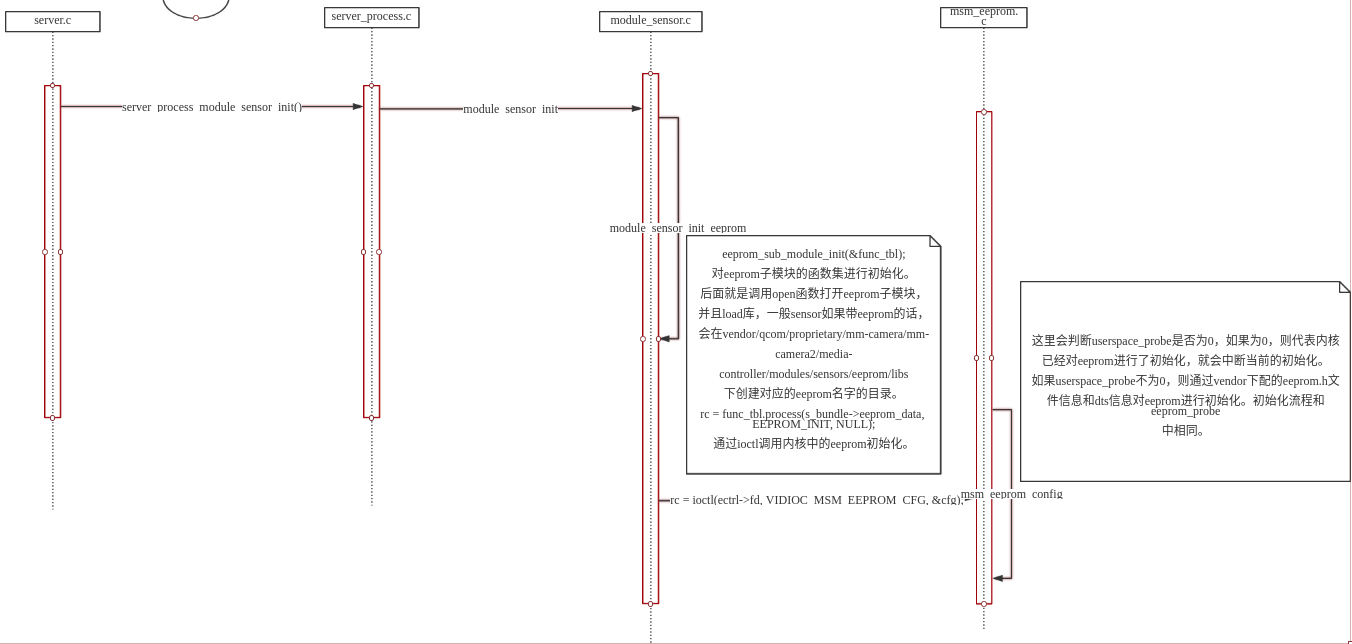


图3.2.2.1.a eeprom\_probe（userspace）流程

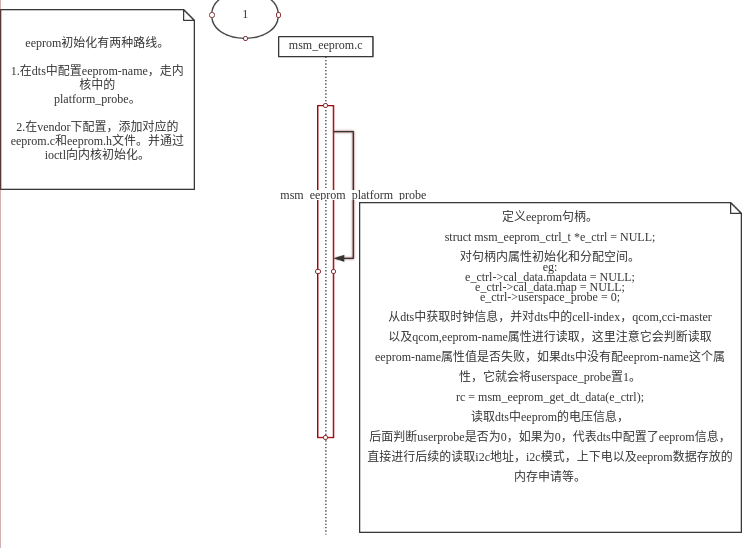


图3.2.2.1.b eeprom\_probe（kernel）流程

#### 3.2.2.2 Porting流程

由于eeprom可以在内核和用于空间加载，所以porting将分为两部分。

1. **Userspace加载**

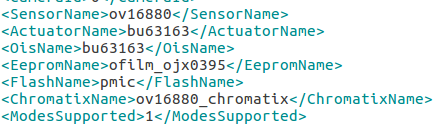
我们需要根据厂商提供的模组信息，来正确的配置以下文件：

XML：.../mm-camera/mm-camera2/media-controller/modules/sensors/configs  
Sensor\_lib：.../mm-camera/mm-camera2/media-controller/modules/sensors/sensor/libs/

device-vendor.mk：vendor/qcom/proprietary/common/config/

Dtsi：kernel/arch/arm/boot/dts/qcom

1. **XML**



Xml配置

EepromName:配置的eeprom名称，加载库文件时作为查找库文件的名称

1. **编译文件**

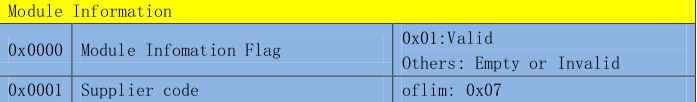
在device-vendor.mk文件添加需要编译的sensor库，例如MM\_CAMERA += libmmcamera\_ofilm\_ojx0395\_eeprom

1. **Lib库文件**
2. **定义宏**

由于eeprom是存储的各个模块的校验数据，后面的代码中会经常的使用，为方便理解和使用，建议将各个模块的起始地址定义为宏，如下：

#define MODULE\_INFO\_FLAG\_OFFSET 0x00 //模组信息起始地址

#define MODULE\_INFO\_OFFSET (MODULE\_INFO\_FLAG\_OFFSET + 1)



1. **上下电配置**

**.**power\_setting\_array **=**

**{**

**.**power\_setting\_a **=**

**{**

**{**

**.....**

**},**

**},**

**.**size **=** 1**,**

**.**power\_down\_setting\_a **=**

**{**

**{**

**......**

**},**

**},**

**.**size\_down **=** 1**,**

按照需求，正确的配置上下电时序。

1. **I2C设置**

**.**i2c\_freq\_mode **=** SENSOR\_I2C\_MODE\_STANDARD**,**

**.**mem\_map\_array **=**

**{**

**.**memory\_map **=**

**{**

**{**

**.**slave\_addr **=** 0xa0**,**

**.**mem\_settings **=**

**{**

**{** 0x0**,** CAMERA\_I2C\_WORD\_ADDR**,**

0x0B5A**,** CAMERA\_I2C\_BYTE\_DATA**,** CAMERA\_I2C\_OP\_READ**,** 0 **},**

**},**

**.**memory\_map\_size **=** 1**,**

**},**

**},**

**.**size\_map\_array **=** 1**,**

**},**

//0x0表示起始的地址，0x0B5A表示eeprom存储的个数。

1. **配置校准数据的使用方式**

**.**get\_calibration\_items **=** ofilm\_ojx0395\_eeprom\_get\_calibration\_items**,**

**.**format\_calibration\_data **=** ofilm\_ojx0395\_eeprom\_format\_calibration\_data**,**

**.**do\_af\_calibration **=** ofilm\_ojx0395\_eeprom\_autofocus\_calibration**,**

**.**do\_wbc\_calibration **=** eeprom\_whitebalance\_calibration**,**

**.**do\_lsc\_calibration **=** eeprom\_lensshading\_calibration**,**

**.**get\_raw\_data **=** **NULL,**

**.**get\_ois\_raw\_data **=** ofilm\_ojx0395\_eeprom\_get\_ois\_raw\_data**,**

按照定义的函数指针编写对应的模块校准函数。

1. **kernel配置**

根据使用的camera索引中对应的eeprom索引，来配置eeprom设备树，例如eeprom0: qcom,eeprom@0 的信息：

cell**-**index **=** **<**0**>;**

reg **=** **<**0**>;**

compatible **=** "qcom,eeprom"**;**

qcom**,**cci**-**master **=** **<**0**>;**

cam\_vio**-**supply **=** **<&**pm8994\_lvs1**>;**

qcom**,**cam**-**vreg**-**name **=** "cam\_vio"**;**

qcom**,**cam**-**vreg**-**min**-**voltage **=** **<**0**>;**

qcom**,**cam**-**vreg**-**max**-**voltage **=** **<**0**>;**

qcom**,**cam**-**vreg**-**op**-**mode **=** **<**0**>;**

1. **Kernel 加载**

一般情况下，我们都是默认在userspace加载，但在一些特殊情况下，例如无法使用XML做兼容而需要区分模组时，就可以在sensor加载前，先从kernel加载eeprom，根据模组寄存器来区分模组。

无论在Kernel加载还是在userspace加载，userspace配置是一样的，可参考userspace加载的配置，这里不再详述，下面介绍差异部分。

Kernel加载是在dtsi设备树里将eeprom的模组名称、I2C地址、上电时序、寄存器读写方式以及数据存储大小标示清楚，在msm\_eeprom\_platform\_probe函数加载dtsi文件时，根据是否配置了eeprom名称，对userspace\_probe 变量赋值，以便判定是否在kernel加载，或者直接写死在kernel加载，后续也需要改写代码，完成寄存器的初始化。

1. **dtsi配置**
2. **模组名称&&I2C设置**

cell**-**index **=** **<**0**>;**

reg **=** **<**0x00**>;**//eeprom 设备寄存器的偏移量

qcom**,**eeprom**-**name **=** "sony\_imx219"**;**

qcom**,**custom**-**eeprom**-**name **=** "sunny\_f8n01m"**;**//做兼容模组

compatible **=** "qcom,eeprom"**;**// eeprom 模块库的相关名称

qcom**,**slave**-**addr **=** **<**0x20**>;**//I2C地址

qcom**,**cci**-**master **=** **<**0**>;**//I2C通道

qcom**,**i2c**-**freq**-**mode **=** **<**1**>;**//i2c速率

**数据存储方式**

qcom**,**num**-**blocks **=** **<**2**>;**//eeprom的存储页数page0、page1等

/\*<page\_size,start\_address,address\_type,data, data\_type, delay(ms). >\*/

/\*size=0 stand for non-use; type : 1 byte, 2 word\*/

qcom**,**page0 **=** **<**1 0x3202 2 0x00 1 1**>;**//设置读取的页面

qcom**,**pageen0 **=** **<**1 0x3200 2 0x01 1 1**>;**//设置读或写

qcom**,**poll0 **=** **<**1 0x3201 2 0x01 1 1**>;**//check &0x3201==0x01？

qcom**,**mem0 **=** **<**64 0x3204 2 0x00 1 0**>;**//读出数据

qcom**,**page1 **=** **<**1 0x3202 2 0x01 1 1**>;**

qcom**,**pageen1 **=** **<**1 0x3200 2 0x01 1 1**>;**

qcom**,**poll1 **=** **<**1 0x3201 2 0x01 1 1**>;**

qcom**,**mem1 **=** **<**64 0x3204 2 0x00 1 0**>;**

**设置初始化寄存器**

qcm**,**setup**-**data**-**num **=** **<**12**>;**

// i2c slave addr, reg addr, write reg val, delay time [msec]

qcm**,**setup**-**data0 **=** **<**0x20 0x0100 2 0x00 1 1**>;** //standby

qcm**,**setup**-**data1 **=** **<**0x20 0x30EB 2 0x05 1 1**>;** //access command

qcm**,**setup**-**data2 **=** **<**0x20 0x30EB 2 0x0C 1 1**>;** //

qcm**,**setup**-**data3 **=** **<**0x20 0x300A 2 0xFF 1 1**>;** //

qcm**,**setup**-**data4 **=** **<**0x20 0x300B 2 0xFF 1 1**>;** //

qcm**,**setup**-**data5 **=** **<**0x20 0x30EB 2 0x05 1 1**>;** //

qcm**,**setup**-**data6 **=** **<**0x20 0x30EB 2 0x09 1 1**>;** //

qcm**,**setup**-**data7 **=** **<**0x20 0x3302 2 0x01 1 1**>;** //clock

qcm**,**setup**-**data8 **=** **<**0x20 0x3303 2 0x2C 1 1**>;** //clock

qcm**,**setup**-**data9 **=** **<**0x20 0x012A 2 0x13 1 1**>;** //frequency

qcm**,**setup**-**data10 **=** **<**0x20 0x012B 2 0x5B 1 1**>;** //frequency

qcm**,**setup**-**data11 **=** **<**0x20 0x3300 2 0x00 1 1**>;** //ecc

**Pinctrl 初始化**

用于设置camera状态的pin脚定义

pinctrl**-**names **=** "cam\_default"**,** "cam\_suspend"**;**

pinctrl**-**0 **=** **<&**qcm\_cam\_sensor\_mclk0\_default **&**qcm\_cam\_sensor\_rear\_default**>;**

pinctrl**-**1 **=** **<&**qcm\_sensor\_mclk0\_sleep **&**qcm\_cam\_sensor\_rear\_sleep**>;**

**GPIO配置**

qcom**,**gpio**-**no**-**mux **=** **<**0**>;**

gpios **=** **<&**msm\_gpio 26 0**>,**

**<&**msm\_gpio 35 0**>,**

**<&**msm\_gpio 90 0**>,** // OCAM\_PON28 -> +2.8V\_OCAMA\_KO

**<&**msm\_gpio 91 0**>,** // OCAM\_PON18 -> +1.8V\_OCAM\_KO

**<&**msm\_gpio 92 0**>,** // OCAM\_PON12 -> +1.2V\_OCAM\_A\_KO

**<&**msm\_gpio 78 0**>;**// 用作判定模组一二供

qcom**,**gpio**-**reset **=** **<**1**>;**//指定reset使用的gpio（即35）

qcom**,**gpio**-**vana **=** **<**2**>;**//指定vana 使用的gpio（即35）

qcom**,**gpio**-**vio **=** **<**3**>;**//指定vio 使用的gpio（即35）

qcom**,**gpio**-**vdig **=** **<**4**>;**//指定vdig 使用的gpio（即35）

qcom**,**gpio**-**custom1 **=** **<**5**>;**//指定custom1 使用的gpio（即35）

qcom**,**gpio**-**req**-**tbl**-**num **=** **<**0 1 2 3 4 5**>;**

qcom**,**gpio**-**req**-**tbl**-**flags **=** **<**0 0 0 0 0 0**>;**

qcom**,**gpio**-**req**-**tbl**-**label **=** "CAM\_MCLK0"**,**

"CAM\_RESET0"**,** "CAM\_AVDD"**,** "CAM\_IOVDD"**,** "CAM\_DVDD"**,**"CAM\_MAKER\_ID"**;**

//对应上面的gpio26、35、90、91、92、78的用于的名称

1. **上点时序**

　/\*上电\*/

qcom**,**cam**-**power**-**seq**-**type **=** "sensor\_gpio"**,**

"sensor\_gpio"**,**

"sensor\_gpio"**,**

"sensor\_clk" **,**

"sensor\_gpio"**;**

/\*下电\*/

qcom**,**cam**-**power**-**seq**-**val **=** "sensor\_gpio\_vdig"**,**

"sensor\_gpio\_vio"**,**

"sensor\_gpio\_vana"**,**

"sensor\_cam\_mclk"**,**

"sensor\_gpio\_reset"**;**

qcom**,**cam**-**power**-**seq**-**cfg**-**val **=** **<**1 1 1 19350000 1**>;**

qcom**,**cam**-**power**-**seq**-**delay **=** **<**0 0 1 0 1**>;**

1. **CLK pin配置**

　　clocks **=** **<&**clock\_gcc clk\_mclk0\_clk\_src**>,**

**<&**clock\_gcc clk\_gcc\_camss\_mclk0\_clk**>;**

clock**-**names **=** "cam\_src\_clk"**,** "cam\_clk"**;**

配置完成，如果配置额外的信息，需要在msm\_eeprom\_platform\_probe过程中添加对应的处理函数

#### 3.2.2.3 验证流程

1. 使用命令将eeprom的数据dump出来，和厂商提供的烧录数据作对比，看是否一致，dump命令如下：

adb root

adb remount

adb shell setprop persist.camera.cal.dump 1

adb reboot

1. 将数据手动修改成比较极限的值，例如修改abw的值，看画面是否出现偏色等现象。
2. 若修改后没有变化，需要添加log并打印以便debug，参见第[4.2](#_4.2 EEPROM PORTING ISSUE)章节。

### 

### **3.2.3 Actuator porting**

在porting之前，需要获取主板原理图、PCB图、模组SPEC、actuator datasheet等资料，正确的配置完参数后，可以在加载库文件的过程和参数调用的过程中加上相应的log，以便做debug用。

#### 3.2.3.1 代码流程

**1）Probe流程**



图3.2.3.1.a actuator\_probe流程

1. **工作流程**

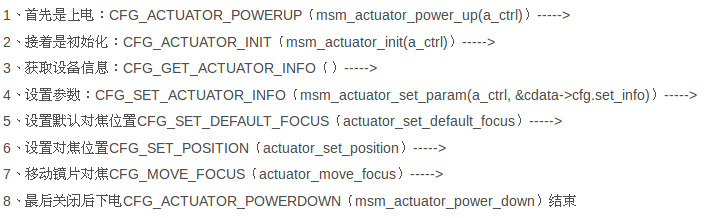


图3.2.3.1.b 工作流程

#### 3.2.3.2 Porting流程

我们需要根据厂商提供的模组信息，来正确的配置以下文件：

XML：.../mm-camera/mm-camera2/media-controller/modules/sensors/configs  
actuator\_lib：.../mm-camera/mm-camera2/media-controller/modules/sensors/actuator/libs/

device-vendor.mk：vendor/qcom/proprietary/common/config/

Dtsi：kernel/arch/arm/boot/dts/qcom

具体配置：

1. **XML**



XML配置

1. **编译文件**

在device-vendor.mk文件添加需要编译的actuator库，例如MM\_CAMERA += libactuator\_ak7371

1. **lib库文件**
2. **Init寄存器**

根据actuator datasheet或者模组厂商，正确配置初始化寄存器

**.**init\_settings **=**

**{**

**{**0xEC**,** CAMERA\_I2C\_BYTE\_ADDR**,** 0xA3**,**

CAMERA\_I2C\_BYTE\_DATA**,** ACTUATOR\_I2C\_OP\_WRITE**,** 0**},**

**......**

**},**

**I2C**

**.**module\_name **=** "abico"**,**

**.**actuator\_name **=** "dw9714"**,**

**.**i2c\_addr **=** 0x18**,**

**.**i2c\_data\_type **=** CAMERA\_I2C\_BYTE\_DATA**,**

**.**i2c\_addr\_type **=** CAMERA\_I2C\_BYTE\_ADDR**,**

**.**act\_type **=** ACTUATOR\_TYPE\_VCM**,**

配置马达名称和I2C信息。

1. **DAC配置**

**.**data\_size **=** 10**,**

**.**reg\_tbl **=**

**{**

**.**reg\_tbl\_size **=** 1**,**

**.**reg\_params **=**

**{**

**{**

**.**reg\_write\_type **=** ACTUATOR\_WRITE\_DAC**,**

**.**hw\_mask **=** 0x0000000F**,**

**.**reg\_addr **=** 0xFFFF**,**

**.**hw\_shift **=** 0**,**

**.**data\_shift **=** 4**,**

**},**

**},**

**},**

DAC（数模转换）将下发的code值转化为马达的驱动电流值，如果地址配置为0xFFFF代表没有寄存器地址。

1. **场景设置**

**.**scenario\_size **=**

**{**

1**,** /\* MOVE\_NEAR \*/

1**,** /\* MOVE\_FAR \*/

**},**

**.**ringing\_scenario **=**

**{**

/\* MOVE\_NEAR \*/

**{**

400**,**

**},**

/\* MOVE\_FAR \*/

**{**

400**,**

**},**

**},**

　　scenario\_size:向近景与远景方向移动部分的场景数。

ringing\_scenario: 各对应场景向近景与远景方向的移动步数最大值。

1. **VCM位置设置**

**.**initial\_code **=** 180**,**

**.**region\_size **=** 1**,**

**.**region\_params **=**

**{**

**{**

**.**step\_bound **=**

**{**

400**,** /\* Macro step boundary\*/

0**,** /\* Infinity step boundary\*/

**},**

**.**code\_per\_step **=** 1**,**

**.**qvalue **=** 128**,**

**},**

**},**

　　initial\_code:初始lens的位置，对应VCM的0~1023的范围，对应DAC的data\_size。

region\_size:对lens的移动进行多个区段划分，可实现更细化的设置(针对非线性)。

region\_params:对应每个区段的在移动步数表中的index范围,即bound。

step\_bound:对应具体的macro及infinity的bound边界值，所有区段合起来就是总 长度(对应上面提到的表的长度)。

code\_per\_step:在当前分区中每个step对应移动的实际lens长度。

qvalue:将浮点型数字转换为整数格式。

注：如果马达有对应的EEPROM补偿数据，最终的马达的位置信息以补偿计算后的为准。

1. **阻尼设置**

**.**damping **=**

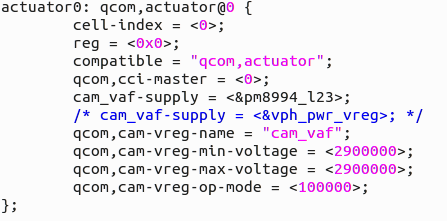
**{**

**......**

**},**

用于抑制对焦过程中马达的震荡。

1. **Kernel**



Dtsi 配置

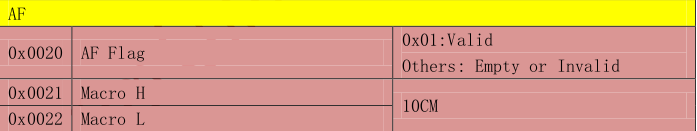
正确配置供电引脚和电压

1. **EEPROM**
2. **定义宏**

将关键的几个地址定义成宏，方便后面使用和阅读。

#define AF\_FLAG\_OFFSET 0x20

#define AF\_OFFSET (AF\_FLAG\_OFFSET + 1)



1. **读取校验数据**

编写函数，将AF的校验数据赋值到对应的结构体，以便后续调用。如：

static void ofilm\_ojx0395\_eeprom\_format\_afdata**(**sensor\_eeprom\_data\_t **\***e\_ctrl**)**

**{**

**......**

af **=** **(**af\_data\_t **\*)(**e\_ctrl**->**eeprom\_params**.**buffer **+** AF\_OFFSET**);**

e\_ctrl**->**eeprom\_data**.**afc**.**macro\_dac **=** **((**af**->**macro\_h **<<** 8**)** **|** af**->**macro\_l**);**

e\_ctrl**->**eeprom\_data**.**afc**.**infinity\_dac **=** **((**af**->**infinity\_h **<<** 8**)** **|** af**->**infinity\_l**);**

e\_ctrl**->**eeprom\_data**.**afc**.**starting\_dac **=** e\_ctrl**->**eeprom\_data**.**afc**.**infinity\_dac**;**

e\_ctrl**->**eeprom\_data**.**afc**.**infinity\_margin **=** INF\_MARGIN**;**

e\_ctrl**->**eeprom\_data**.**afc**.**macro\_margin **=** MACRO\_MARGIN**;**

**......**

**}**

#### 3.2.3.3 验证流程

1. 启动设备后，通过adb命令进入设备，在sys/class/video4linux目录下，通过cat \*/name 获取节点名称，查看配置的actuator是否挂载。

2. 设置命令，启动full\_sweep模式，看画面是否能够由模糊到清晰再到模糊最后跳至清晰点

3. 如果马达不能正常工作，需要查看log分析，文档汇总了一些遇到的情况和分析，参见第[4.3](#_4.3 ACTUATOR PORTING ISSUE)章节

### 

### **3.2.4 OIS porting**

OIS分为平移式和移轴式，从大类上分，属于马达的一种，是在传统对焦马上的基础上，通过同一个driver ic 进行控制，以抵消设备晃动导致的画面模糊或视频抖动。所以，VCM带有OIS（防抖）功能时，初始化寄存器一般会在OIS里配置。

OIS一般时在camera启动后写入对应的模式，不经过3A算法，除初始化寄存器外，一般还会有对应的校验数据需要写入，这部分数据在module\_sensor\_offload\_ois\_init\_calibrat中从eeeprom中读取。

Porting 前需要需要准备主板原理图、PCB图、模组SPEC、OIS datasheet等资料。

#### 3.2.4.1 代码流程



图3.2.4.1ois\_probe流程

#### 3.2.4.2 Porting流程

我们需要根据厂商提供的信息，来正确的配置以下文件：

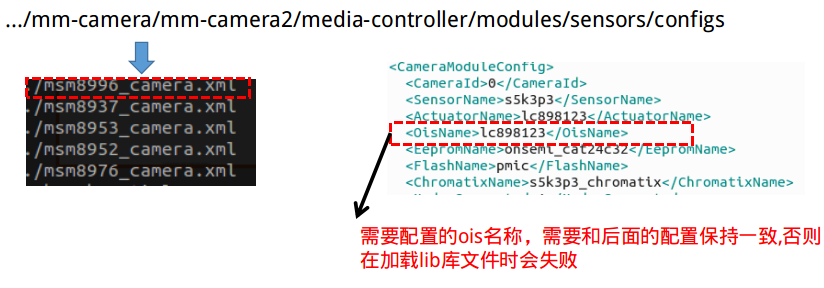
XML：.../mm-camera/mm-camera2/media-controller/modules/sensors/configs  
ois\_lib：.../mm-camera/mm-camera2/media-controller/modules/sensors/actuator/libs/

device-vendor.mk：vendor/qcom/proprietary/common/config/

Dtsi：kernel/arch/arm/boot/dts/qcom

具体配置：

1. **XML**



XML配置

1. **编译文件**

在device-vendor.mk文件添加需要编译的actuator库，例如MM\_CAMERA += libois\_bu63165

1. **lib库文件**
2. **I2C**

**.**module\_name **=** "rohm"**,** /\* module name \*/

**.**ois\_name **=** "bu63165"**,** /\* ois name \*/

**.**i2c\_addr **=** 0x1c**,** /\* I2C Address \*/

**.**i2c\_freq\_mode **=** SENSOR\_I2C\_MODE\_CUSTOM**,** /\* I2C frequency mode \*/

**INIT寄存器**

**.**init\_setting\_size **=** 725**,**

**.**init\_settings **=**

**{**

**......**

**}**

**使能和模式寄存器**

**.**enable\_ois\_setting\_size **=** 1**,**

**.**enable\_ois\_settings **=**

**{**

**......**

**},** //end ois settings

**.**disable\_ois\_setting\_size **=** 1**,**

**.**disable\_ois\_settings **=**

**{**

**......**

**},** //end disable ois settings

**.**movie\_mode\_ois\_setting\_size **=** 14**,**

**.**movie\_mode\_ois\_settings **=**

**{**

**......**

**},** // end movie\_mode\_ois\_settings

**.**still\_mode\_ois\_setting\_size **=** 14**,**

**.**still\_mode\_ois\_settings **=**

**{**

**......**

**},** //end still\_mode\_ois\_settings

**.**centering\_on\_ois\_setting\_size **=** 0**,**

**.**centering\_on\_ois\_settings **=**

**{**

**......**

**},** //end centering\_on\_ois\_settings

**.**centering\_off\_ois\_setting\_size **=** 0**,**

**.**centering\_off\_ois\_settings **=**

**{**

**......**

**},** //end centering\_off\_ois\_settings

**.**pantilt\_on\_ois\_setting\_size **=** 14**,**

**.**pantilt\_on\_ois\_settings **=**

**{**

**......**

**},** //end pantilt\_on\_ois\_setting\_size

以上寄存器包括Init寄存器的配置，需要模组厂商提供。

1. **Eeprom**
2. **定义宏**

将关键的几个地址定义成宏，方便后面使用和阅读。

#define OIS\_TAG\_OFFSET 0x0B31

#define OIS\_START\_OFFSET (OIS\_TAG\_OFFSET + 3)

#define OIS\_SETTING\_SIZE 17



1. **读取校验数据**

编写函数，将OIS的校验数据赋值到对应的结构体，以便后续调用。如：

static void ofilm\_ojx0395\_eeprom\_format\_oisdata**(**sensor\_eeprom\_data\_t **\***e\_ctrl**,**

unsigned int ois\_offset**)** **{**

**......**

g\_reg\_setting\_ois**.**addr\_type **=** CAMERA\_I2C\_WORD\_ADDR**;**

g\_reg\_setting\_ois**.**reg\_setting **=** **&**g\_reg\_array\_ois**[**0**];**

g\_reg\_setting\_ois**.**size **=** OIS\_SETTING\_SIZE**;**

g\_reg\_setting\_ois**.**delay **=** 0**;**

/\* CURDAT \*/

g\_reg\_array\_ois**[**0**].**reg\_addr **=** 0x8230**;**

g\_reg\_array\_ois**[**0**].**reg\_data**[**0**]** **=** buffer**[**ois\_offset**];**

g\_reg\_array\_ois**[**0**].**reg\_data**[**1**]** **=** buffer**[**ois\_offset **+** 1**];**

g\_reg\_array\_ois**[**0**].**reg\_data\_size **=** 2**;**

/\* HALOFS\_X \*/

g\_reg\_array\_ois**[**1**].**reg\_addr **=** 0x8231**;**

g\_reg\_array\_ois**[**1**].**reg\_data**[**0**]** **=** buffer**[**ois\_offset **+** 2**];**

g\_reg\_array\_ois**[**1**].**reg\_data**[**1**]** **=** buffer**[**ois\_offset **+** 3**];**

g\_reg\_array\_ois**[**1**].**reg\_data\_size **=** 2**;**

**......**

**}**

#### 3.2.4.3 验证流程

1. 用没有OIS功能的手机做对比，晃动时观看画面是否稳定。

### 

### **3.2.5 PDAF porting**

由于PDAF 是一个辅助对焦功能，并没有单独的硬件，所以不存在硬件的porting，是在sensor和马达成功porting后，进行的功能porting，pdaf分为TYPE\_1&&TYPE\_2和TYPE\_3，正常情况下，我们只需要将eeprom中pdaf的数据成功配置并且传给tuning使用，就可以。

鉴于上述，我们需要准备EEPROM Programming Guide、OTP data以及模组厂商提供的关于pdaf的pixel和镜像数据的头文件。

#### 3.2.5.1 代码流程

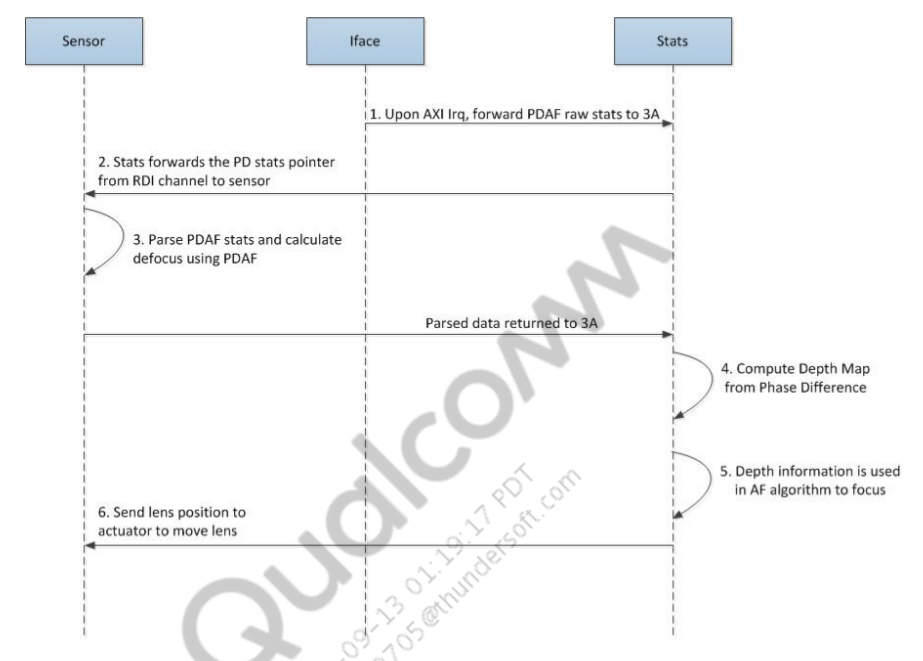


图3.2.5.1call flow

#### 3.2.5.2 Porting流程

#### 1）Sensor端配置

**A. Pdaf 使能**

**.**out\_info **=**

**{**

/\* Res 0 \*/

**{**

**......**

**.**offset\_y **=** 0**,**

**.**scale\_factor **=** 1.000**,**

**.**is\_pdaf\_supported **=** 1**,**//pdaf使能

**},**

需要在各个Res组中配置该属性，置1表示打开pdaf功能。.

1. **包含pdaf数据**

　　 #ifndef FLIP\_MIRROR

#include "ov16880\_pdaf.h"

#else

#include "ov16880\_pdaf\_mirror\_flip.h"

#endif

将厂商提供的pdaf的头文件包含到代码中。

#### 2）Eeprom数据

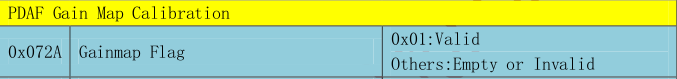
**A. 定义宏**

按照EEPROM Programming Guide中PDAF的信息，对PDAF有标志性的几个地址定义宏，方便后续使用。

#define PD\_GAIN\_MAP\_FLAG 0x072A

#define PD\_GAIN\_MAP\_SIZE 252

**......**





......

**B. 读取校验数据**

编写函数，将PDAF的校验数据赋值到PDAF对应的结构体，以便后续算法调用。如：

static void ofilm\_ojx0395\_format\_pdafgain**(**sensor\_eeprom\_data\_t **\***e\_ctrl**)**

**{**

**......**

pdaf\_ptr **=** e\_ctrl**->**eeprom\_params**.**buffer **+** PD\_GAIN\_MAP\_OFFSET\_X**;**

pdafc\_2d**->**OffsetX **=** **(**unsigned short**)(**pdaf\_ptr**[**0**]** **<<** 8**|** pdaf\_ptr**[**1**]** **);**

pdaf\_ptr **+=** 2**;**

pdafc\_2d**->**OffsetY **=** **(**unsigned short**)(**pdaf\_ptr**[**0**]<<** 8**|**pdaf\_ptr**[**1**]** **);**

pdaf\_ptr **+=** 2**;**

pdafc\_2d**->**RatioX **=** **(**unsigned short**)(**pdaf\_ptr**[**0**]<<** 8**|**pdaf\_ptr**[**1**]** **);**

pdaf\_ptr**+=**2**;**

pdafc\_2d**->**RatioY **=** **(**unsigned short**)(**pdaf\_ptr**[**0**]<<** 8**|**pdaf\_ptr**[**1**]** **);**

pdaf\_ptr**+=**2**;**

pdafc\_2d**->**MapWidth**=(**unsigned short**)(**pdaf\_ptr**[**0**]** **<<** 8 **|** pdaf\_ptr**[**1**]);**

pdaf\_ptr **+=** 2**;**

pdafc\_2d**->**MapHeight **=(**unsigned short**)(**pdaf\_ptr**[**0**]** **<<** 8 **|** pdaf\_ptr**[**1**]);**

pdaf\_ptr **+=** 2**;**

right\_gain\_map **=** e\_ctrl**->**eeprom\_params**.**buffer **+** PD\_LEFT\_GAIN\_MAP\_OFFSET**;**

left\_gain\_map **=** e\_ctrl**->**eeprom\_params**.**buffer **+** PD\_RIGHT\_GAIN\_MAP\_OFFSET**;**

**for** **(** i **=** 0**;** i **<** PD\_GAIN\_MAP\_SIZE**;** i**++)**

**{**

j **=** 2 **\*** i**;**

pdafc\_2d**->**Left\_GainMap**[**i**]** **=** **(**unsigned short**)(**left\_gain\_map**[**j**]** **<<** 8 **|** left\_gain\_map**[**j**+**1**]);**

pdafc\_2d**->**Right\_GainMap**[**i**]** **=** **(**unsigned short**)(**right\_gain\_map**[**j**]** **<<** 8 **|** right\_gain\_map**[**j**+**1**]);**

**}**

dcc\_map **=** e\_ctrl**->**eeprom\_params**.**buffer **+** PD\_DCC\_MAP**;**

pdafc\_2d**->**PD\_conversion\_coeff**[**0**]** **=** **(**int**)(**dcc\_map**[**2**]** **<<** 8 **|** dcc\_map**[**3**]);**

**}**

#### 3.2.5.3 验证流程

1. dumpOTP数据，看数据是否正确。

2. 在光线较好的情况下，触发对焦，看是否有对焦框，没有就是PDAF工作了。

3. 使用tuning方式，对PD值做判断。

4. 如不能正常工作，需要根据相应的log来debug，第[4.5](#_4.5 PDAF PORTING ISSUE)章节总结了一些遇到的问题，可作参考。

### **3.2.6 FLASH porting**

高通平台的LED\_FLASH有3种控制方式，PMIC、QUP/I2C 或 CCI（camera control interface，camera专用接口，包含GPIO和I2C），当然，还有一种LED，不论是否有自己的控制芯片，只要平台提供一个电压就可以工作，对于这种可以使用GPIO控制。

不同的控制方式有不同的控制方式，下面将会详细介绍。

#### 3.2.6.1 代码流程

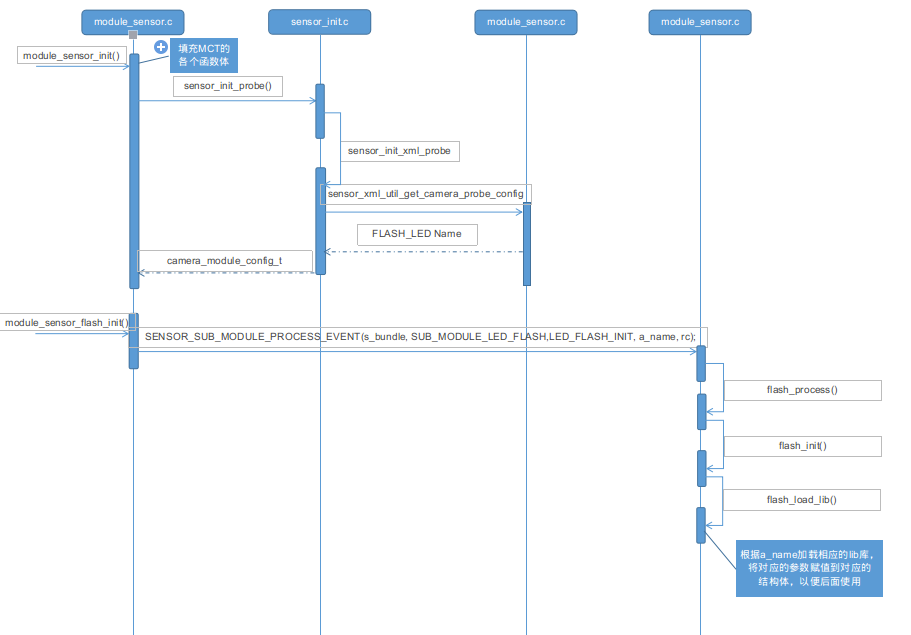


图3.2.6.1FLASH probe（userspace）

#### 3.2.6.2 Porting流程

**PMIC控制：**

**1）Kernel配置**

**A. 设备树配置**

1. 在对应的pmic设备树中定义LED的驱动管理，如在msm-pmi8998.dtsi配置如下信息：

　pmi8998\_flash0**:** qcom**,**flash\_0 **{**

label **=** "flash"**;**

qcom**,**led**-**name **=** "led:flash\_0"**;**

qcom**,**max**-**current **=** **<**1500**>;**//最大电流

qcom**,default-**led**-**trigger **=** "flash0\_trigger"**;**//默认的触发配置名称

qcom**,**id **=** **<**0**>;**

qcom**,**current**-**ma **=** **<**1000**>;**//正常电流

qcom**,**duration**-**ms **=** **<**1280**>;**//最大时间

qcom**,**ires**-**ua **=** **<**12500**>;**

qcom**,**hdrm**-**voltage**-**mv **=** **<**325**>;**

qcom**,**hdrm**-**vol**-**hi**-**lo**-**win**-**mv **=** **<**100**>;**

**};**

pmi8998\_flash1**:** qcom**,**flash\_1 **{**

**......**

**};**

pmi8998\_flash2**:** qcom**,**flash\_2 **{**

**......**

**};**

pmi8998\_torch0**:** qcom**,**torch\_0 **{**

**......**

**};**

pmi8998\_torch1**:** qcom**,**torch\_1 **{**

**......**

**};**

pmi8998\_torch2**:** qcom**,**torch\_2 **{**

**......**

**};**

pmi8998\_switch0**:** qcom**,**led\_switch\_0 **{**

label **=** "switch"**;**

qcom**,**led**-**name **=** "led:switch\_0"**;**

qcom**,**led**-**mask **=** **<**3**>;**

qcom**,default-**led**-**trigger **=** "switch0\_trigger"**;**

**};**

pmi8998\_switch1**:** qcom**,**led\_switch\_1 **{**

**......**

**};**

**}**

在对应的camera索引中添加led索引

qcom**,**led**-**flash**-**src **=** **<&**led\_flash0**>;**

1. 在对应的led索引添加PMIC配置

**&**soc **{**

led\_flash0**:** qcom**,**camera**-**flash@0 **{**

cell**-**index **=** **<**0**>;**

compatible **=** "qcom,camera-flash"**;**//此节点对应的数据类型

qcom**,**flash**-**source **=** **<&**pmi8998\_flash0 **&**pmi8998\_flash1**>;**

qcom**,**torch**-**source **=** **<&**pmi8998\_torch0 **&**pmi8998\_torch1**>;**

qcom**,switch-**source **=** **<&**pmi8998\_switch0**>;**//切换配置

status **=** "ok"**;**//是否启用或禁用此节点

**};**

**}**

配置的参数会按如下的流程调用：module\_init()->msm\_flash\_init\_module()->msm\_flash\_platform\_driver()->msm\_flash\_platform\_probe()->msm\_flash\_get\_dt\_data()->msm\_flash\_get\_pmic\_source\_info()，在msm\_flash\_get\_dt\_data中定义flash的type默认为：

fctrl->flash\_driver\_type = FLASH\_DRIVER\_DEFAULT;

后续在msm\_flash\_get\_pmic\_source\_info读取PMIC配置，如果成功读取，则将flash的type设置成：

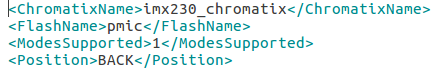
**if** **(**fctrl**->**flash\_driver\_type **==** FLASH\_DRIVER\_DEFAULT**)**

fctrl**->**flash\_driver\_type **=** FLASH\_DRIVER\_PMIC**;**

待所有的配置读取完成后将会创建LED\_Flash节点。

1. **Vendor**
2. **Xml**

在../mm-camera/mm-camera2/media-controller/modules/sensors/configs的xxx\_camera.xml配置如下信息：



1. **编译文件**

在device-vendor.mk文件添加需要编译的sensor库，例如MM\_CAMERA += libflash\_pmic

1. **库文件**

PMIC的库文件配置较简单，只需要配置type类型和跳帧就可以，如下：

static flash\_lib\_t flash\_lib\_ptr **=** **{**

**.**flash\_name **=** "pmic"**,**

**.**flash\_driver\_type **=** FLASH\_DRIVER\_TYPE\_DEFAULT**,**

**.**main\_flash\_on\_frame\_skip **=** 3**,**

**.**main\_flash\_off\_frame\_skip **=** 3**,**

**.**torch\_on\_frame\_skip **=** 2**,**

**.**torch\_off\_frame\_skip **=** 2**,**

**};**

Type类型配置为FLASH\_DRIVER\_TYPE\_DEFAULT表示LED各个模式下的参数（例如电流、电压等）从kernel端获取，也就是上面设备树配置的状态数据。

**CCI控制：**

1. **Kernel配置**
2. **设备树配置**

CCI的配置和PMIC的配置有相同点和不同点，kernel端的相同点是都要在对应的camera索引中添加led的索引，不同点如下：

1. 配置

　cci **{**

led\_flash0**:** qcom**,**led**-**flash@66 **{**

cell**-**index **=** **<**0**>;**

reg **=** **<**0x66**>;**

label **=** "bd7710"**;**//LED名称

qcom**,**flash**-**type **=** **<**1**>;**//闪光灯类型， LED flash、Strobe flash、Simple LED flash

qcom**,**gpio**-**no**-**mux **=** **<**0**>;**//1-不使用、0-使用

pinctrl**-**names **=** "cam\_flash\_default"**,** "cam\_flash\_suspend"**;**//使用的pinctrl的名称

pinctrl**-**0 **=** **<&**cam\_sensor\_flash\_default**>;**//pin0对应的索引

pinctrl**-**1 **=** **<&**cam\_sensor\_flash\_sleep**>;**//pin1对应的索引

gpios **=** **<&**msm\_gpio 36 0**>,**//使用的GPIO的端口

**<&**msm\_gpio 32 0**>,**

**<&**msm\_gpio 31 0**>;**

qcom**,**gpio**-**flash**-**reset **=** **<**0**>;**//reset对应的GPIO端口号（36）

qcom**,**gpio**-**flash**-**en **=** **<**1**>;**//enable对应的GPIO端口号（32）

qcom**,**gpio**-**flash**-**now **=** **<**2**>;**//now 对应的GPIO端口号（31）

qcom**,**gpio**-**req**-**tbl**-**num **=** **<**0 1 2**>;**//GPIO对应的索引号

qcom**,**gpio**-**req**-**tbl**-**flags **=** **<**0 0 0**>;**

qcom**,**gpio**-**req**-**tbl**-**label **=** "FLASH\_RST"**,**

"FLASH\_EN"**,**

"FLASH\_NOW"**;**

qcom**,**cci**-**master **=** **<**0**>;**//CCI端口

**};**

索引配置

配置上面pinctrl对应的索引

cam\_sensor\_flash\_default **{**

qcom**,**pins **=** **<&**gp 26**>;**

qcom**,**num**-**grp**-**pins **=** **<**1**>;**

qcom**,**pin**-**func **=** **<**1**>;**

label **=** "cam\_sensor\_flash\_default"**;**

/\* active state \*/

fj\_cam\_sensor\_mclk0\_default**:** **default** **{**

drive**-**strength **=** **<**4**>;** /\* 4 MA \*/

bias**-**disable **=** **<**0**>;** /\* No PULL \*/

**};**

配置的参数会按如下的流程调用：module\_init()->msm\_flash\_init\_module()->msm\_flash\_platform\_driver()->msm\_flash\_platform\_probe()->msm\_flash\_get\_dt\_data()->msm\_sensor\_driver\_get\_gpio\_data()，在msm\_flash\_get\_dt\_data通过一些判定，如果成功读取到CCI的配置信息，则fctrl->flash\_driver\_type = FLASH\_DRIVER\_I2C，后续在msm\_sensor\_driver\_get\_gpio\_data读取CCI的配置，待所有的配置读取完成后将会创建LED\_Flash节点。

1. **Vendor**

配置完kernel，同样需要配置xml、编译文件以及lib库，其中xml和编译文件配置相同，不再详述，下面是lib库文件配置：

1. **模组信息**

配置flash对应的名称等

**.**flash\_name **=** "bd7710"**,**

**.**flash\_type **=** FLASH\_TYPE\_LED**,**

**.**flash\_driver\_type **=** FLASH\_DRIVER\_TYPE\_I2C**,**

**I2C配置**

配置I2C地址，地址类型，match地址和数据类型等。

**.**slave\_addr **=** 0x66**,**

**.**i2c\_addr\_type **=** CAMERA\_I2C\_BYTE\_ADDR**,**

**.**flash\_init\_settings **=**

**{**

**.**reg\_setting\_a **=**

**{**

**{**0x00**,** 0x10**,** 0x00**},**

**},**

**.**size **=** 1**,**

**.**addr\_type **=** CAMERA\_I2C\_BYTE\_ADDR**,**//8位数据

**.**data\_type **=** CAMERA\_I2C\_BYTE\_DATA**,**//8位地址

**.**delay **=** 0**,**

1. **上电时序**

根据datasheet配置上下点时序。

**.**power\_setting\_array **=**

**{**

**.**power\_setting\_a **=**

**{**

**{**

**.**seq\_type **=** CAMERA\_POW\_SEQ\_GPIO**,**

**.**seq\_val **=** CAMERA\_GPIO\_RESET**,**

**.**config\_val **=** GPIO\_OUT\_LOW**,**

**.**delay **=** 1**,**

**},**

**......**

**},**

**},**

**.**size **=** 2**,**

**.**power\_down\_setting\_a **=**

**{**

**{**

**.**seq\_type **=** CAMERA\_POW\_SEQ\_GPIO**,**

**.**seq\_val **=** CAMERA\_GPIO\_RESET**,**

**.**config\_val **=** GPIO\_OUT\_HIGH**,**

**.**delay **=** 1**,**

**},**

**......**

**},**

**.**size\_down **=** 2**,**

**},**

**电压配置**

配置LED的数量，lash和torch模式下的最大电流以及最大连续工作时间。

**.**num\_of\_flash **=** 1**,**

**.**max\_flash\_current **=** **{**1000**,** 0**,** 0**},**

**.**max\_torch\_current **=** **{**200**,** 0**,** 0**},**

**.**max\_flash\_duration **=** **{**1200**,** 0**,** 0**},**

**寄存器配置**

根据datasheet或FAE提供的数据，配置各个模式下的寄存器设置

**.**flash\_off\_settings **=**

**{**

**.**reg\_setting\_a **=**

**{**

**{**0x05**,** 0x00**,** 0x00**},**

**{**0x02**,** 0x00**,** 0x00**},**

**},**

**.**size **=** 2**,**

**.**addr\_type **=** CAMERA\_I2C\_BYTE\_ADDR**,**

**.**data\_type **=** CAMERA\_I2C\_BYTE\_DATA**,**

**.**delay **=** 0**,**

**},**

**.**flash\_low\_settings **=**

**{**

**......**

**},**

**.**flash\_high\_settings **=**

**{**

**......**

**},**

**GPIO控制：**

GPIO的控制相对较简单，kernel端配置设备树，在userspace配置一个类似PMIC的库即可。

1. **Kernel**

在上述CCI的配置上，将led的索引做修改，如下：

qcom**,**led**-**flash@60 **{**

reg **=** **<**0x60**>;**

cell**-**index **=** **<**0**>;**

qcom**,**flash**-**type **=** **<**2**>;**

gpios **=** **<&**msmgpio 23 0**>,**

**<&**msmgpio 24 0**>;**

qcom**,**gpio**-**flash**-**en **=** **<**0**>;**

qcom**,**gpio**-**flash**-**now **=** **<**1**>;**

qcom**,**gpio**-**req**-**tbl**-**num **=** **<**0 1**>;**

qcom**,**gpio**-**req**-**tbl**-**flags **=** **<**0 0**>;**

qcom**,**gpio**-**req**-**tbl**-**label **=** "FLASH\_EN"**,**

"FLASH\_NOW"**;**

qcom**,**max**-**current **=** **<**750**>;**

qcom**,**max**-**duration **=** **<**1600**>;**

**};**

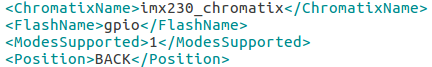
配置的参数会按如下的流程调用：module\_init()->msm\_flash\_init\_module()->msm\_flash\_platform\_driver()->msm\_flash\_platform\_probe()->msm\_flash\_get\_dt\_data()->msm\_sensor\_driver\_get\_gpio\_data()，在msm\_flash\_get\_dt\_data通过一些判定，如果不能读取到CCI的配置信息，则fctrl->flash\_driver\_type = FLASH\_DRIVER\_DEFAULT，后续在msm\_sensor\_driver\_get\_gpio\_data读取GPIO的配置，读取完成后判定fctrl->flash\_driver\_type的值，并做更改，如下：

**if** **(**fctrl**->**flash\_driver\_type **==** FLASH\_DRIVER\_DEFAULT**)**

fctrl**->**flash\_driver\_type **=** FLASH\_DRIVER\_GPIO**;**

1. **Vendor**
2. **Xml**

在.../mm-camera/mm-camera2/media-controller/modules/sensors/configs的xxx\_camera.xml配置如下信息：



1. **编译文件**

在device-vendor.mk文件添加需要编译的flash库，例如MM\_CAMERA += libflash\_gpio

1. **库文件**

PMIC的库文件配置较简单，只需要配置type类型和跳帧就可以，如下：

static flash\_lib\_t flash\_lib\_ptr **=** **{**

**.**flash\_name **=** "gpio"**,**

**.**flash\_driver\_type **=** FLASH\_DRIVER\_TYPE\_DEFAULT**,**

**.**main\_flash\_on\_frame\_skip **=** 3**,**

**.**main\_flash\_off\_frame\_skip **=** 3**,**

**.**torch\_on\_frame\_skip **=** 2**,**

**.**torch\_off\_frame\_skip **=** 2**,**

**};**

Type类型配置为FLASH\_DRIVER\_TYPE\_DEFAULT表示LED各个模式下的参数（例如电流、电压等）从kernel端获取，也就是上面设备树配置的状态数据。

**QUP/I2C：**

I2C的控制和CCI的类似，不一样的是需要提前在xxx\_mtp.dtsi文件里注册连接到&i2c 的 QUP 节点，如下：

i2c**:** i2c@f9928000 **{** /\* BLSP1 QUP6 \*/

cell**-**index **=** **<**6**>;**

compatible **=** "qcom,i2c-qup"**;**

#address-cells = <1>;

#size-cells = <0>;

reg**-**names **=** "qup\_phys\_addr"**;**

reg **=** **<**0xf9928000 0x1000**>;**

interrupt**-**names **=** "qup\_err\_intr"**;**

interrupts **=** **<**0 100 0**>;**

qcom**,**i2c**-**bus**-**freq **=** **<**100000**>;**

qcom**,**i2c**-**src**-**freq **=** **<**19200000**>;**

qcom**,**sda**-**gpio **=** **<&**msmgpio 16 0**>;**

qcom**,**scl**-**gpio **=** **<&**msmgpio 17 0**>;**

qcom**,**master**-**id **=** **<**86**>;**

**};**

注册完就可以使用I2C节点，配置如下：

**&**i2c **{**

led\_flash0**:** qcom**,**led**-**flash@60 **{**

cell**-**index **=** **<**0**>;**

reg **=** **<**0x60**>;**

compatible **=** "qcom,led-flash"**;**

qcom**,**flash**-**name **=** "adp1600"**;**

qcom**,**flash**-**type **=** **<**1**>;**

qcom**,**gpio**-**no**-**mux **=** **<**0**>;**

gpios **=** **<&**msmgpio 18 0**>,**

**<&**msmgpio 19 0**>;**

qcom**,**gpio**-**flash**-**en **=** **<**0**>;**

qcom**,**gpio**-**flash**-**now **=** **<**1**>;**

qcom**,**gpio**-**req**-**tbl**-**num **=** **<**0 1**>;**

qcom**,**gpio**-**req**-**tbl**-**flags **=** **<**0 0**>;**

qcom**,**gpio**-**req**-**tbl**-**label **=** "FLASH\_EN"**,**

"FLASH\_NOW"**;**

**};**

上面配置的信息的含义，可以参考CCI的配置，其余的配置和CCI的类似，不做详述。

#### 3.2.6.3 验证流程

# 

# 4 – BRING UP经验分享

## 4.1 SENSOR PORTING ISSUE

### **4.1.1 I2C相关**

#### 4.1.1.1 Read id failed

现象：打开camera，出现“Can not connect camera”的报错。

概率：100%

原因：slave address配置有误。

分析：从kernel 信息看，出现“read id failed”的信息，从I2C的波形上，没有响应信号：

波形：



Log：

**[**22.662594**]** msm\_cci\_init**:**1439**:** hw\_version **=** 0x10060000

**[**22.663099**]** msm\_cci\_irq**:**1797 MASTER\_0 error 0x10000000

**[**22.663140**]** msm\_cci\_i2c\_read**:**960 read\_words **=** 0**,** exp words **=** 1

**[**22.663141**]** msm\_cci\_i2c\_read\_bytes**:**1043 failed rc **-**22

**[**22.663143**]** msm\_camera\_cci\_i2c\_read**:** line 47 rc **=** **-**22

**[**22.663146**]** msm\_sensor\_match\_id**:** adv7181**:** read id failed

**[**22.663147**]** msm\_sensor\_check\_id**:**1410 match id failed rc **-**22

解决：高通的 slave address 是7位的地址+1位的读写标记位，将datasheet地址信息和客户确认，如果datasheet上的是7位地址，补上对应的标记位。

#### 4.1.1.2 驱动电压

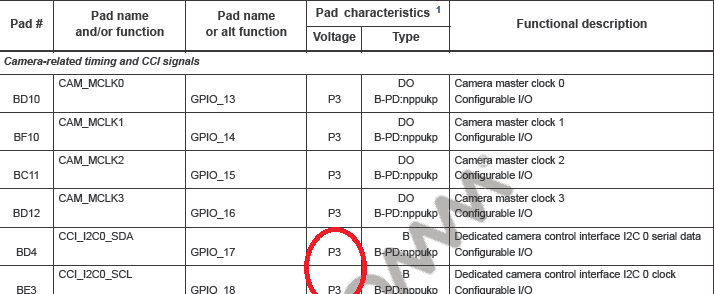
现象：打开camera，出现“Can not connect camera”的报错。

概率：100%

原因：高通平台的I2C电压为1.8V，sensor段的需求为3.3V，导致平台发出的高电平被sensor判定为低电平，不做应答。

分析：通过量取I2C的电压，和sensor的原理图做对比，看是否满足要求。另外，高通平台对电压的限制如下：





解决：联系硬件工程师，对I2C电路改造，使I2C的信号在到达sensor端时能被拉升到3.3V。

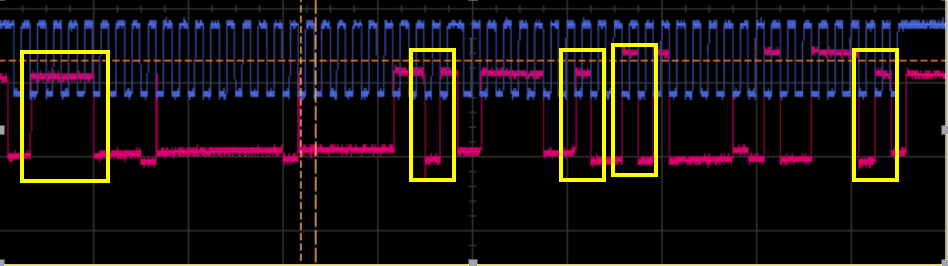
#### 4.1.1.3 Pin脚虚焊

现象：打开camera，出现“Can not connect camera”的报错。

概率：100%

原因：pin脚虚焊导致I2C的信号变形，不符合sensor接收端的需求

分析：通过I2C的波形查看，电压有跳动：



解决：联系硬件工程师对各个pin脚检查，将虚焊的pin脚重新焊接。

#### 4.1.1.4 寄存器存储类型

现象：打开camera，出现“Can not connect camera”的报错。

概率：100%

原因：高通代码里默认寄存器的存储方式是Word型，但寄存器本身是的存储方式是Byte型。

分析：通过log查看，会有math\_id failed的错误，根据打印的寄存器的值或I2C读出的数值位数，与datasheet中的寄存器作对比：

**[**22.137393**]** msm\_cci\_init**:**1439**:** hw\_version **=** 0x10060000

**[**22.138033**]** msm\_sensor\_match\_id chip id 1c9 does not match 1

**[**22.138079**]** msm\_sensor\_check\_id**:**1410 match id failed rc **-**19

解决：在配置寄存器时注意寄存器的类型，另外在下发IOCTL读写命令的代码处检查是否有类型判断，如果没有，将下发的TYPE类型手动修改为想要的类型。

#### 4.1.1.5 寄存器读写超时

### **4.1.2 FpsRange**

#### 4.1.2.1 Sensor FPS配置

现象：打开camera，直接闪退，界面显示“Snapdragon Camera Keeping stopping”。

概率：100%

原因：配置了不支持的FPS。

分析：查看log，发现报getPhotoPreviewFpsRange相关的信息，如下：

**---------** beginning of crash

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** FATAL EXCEPTION**:** Thread**-**2

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** Process**:** org**.**codeaurora**.**snapcam**,** PID**:** 3274

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** java**.**lang**.**NullPointerException**:** Attempt to invoke interface method 'int java.util.List.size()' on a null object reference

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** at com**.**android**.**camera**.**util**.**CameraUtil**.**getPhotoPreviewFpsRange**(**CameraUtil**.**java**:**1004**)**

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** at com**.**android**.**camera**.**util**.**CameraUtil**.**getPhotoPreviewFpsRange**(**CameraUtil**.**java**:**1000**)**

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** at com**.**android**.**camera**.**PhotoModule**.**updateCameraParametersInitialize**(**PhotoModule**.**java**:**3193**)**

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** at com**.**android**.**camera**.**PhotoModule**.**setCameraParameters**(**PhotoModule**.**java**:**4345**)**

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** at com**.**android**.**camera**.**PhotoModule**.**startPreview**(**PhotoModule**.**java**:**3152**)**

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** at com**.**android**.**camera**.**PhotoModule**.**access$100**(**PhotoModule**.**java**:**124**)**

08**-**06 08**:**32**:**31.006 3274 3301 E AndroidRuntime**:** at com**.**android**.**camera**.**PhotoModule$OpenCameraThread**.**run**(**PhotoModule**.**java**:**294**)**

解决：将sensor配置的FPS修改到合理范围。

### **4.1.3 MIPI 相关**

#### 4.1.3.1 CAMIF 报错

现象：打开camera会出现黑屏或闪退

概率：100%

原因：sensor的out\_info配置错误。

分析：查看log，发现报CAMIF的错误，错误显示我们配置的分辨率有问题有问题(我们需要的是0x148，但)，信息如下：

**[**283.771458**]** ispif\_process\_irq**:** PIX0 frame id**:** 4

**[**283.772802**]** msm\_vfe47\_process\_error\_status**:** camif error status**:** 0x148

**[**283.772812**]** 0x0000**:** 00000001 00000040 00000000 02cf013f

**[**283.772820**]** 0x0010**:** 0000013f 000002cf 00000000 00001f1f

**[**283.772828**]** 0x0020**:** ffffffff ffffffff 0014029d 00000148

**[**283.772835**]** 0x0030**:** 00000000 0000000f 00000000

解决：根据datasheet配置合理的分辨率。

#### 4.1.3.2 VFE 溢出报错

现象：打开canera黑屏后退出。

概率：100%

原因：配置的op\_pixel\_clk过小。

分析：导出log查看，有相关的overflow的报错。

解决：查找datasheet是否有相关的信息，根据如下的方式计算：

op\_pixel\_clk = (传感器总数据传输速率) / 每个像素的位数

例如,如果 MIPI DDR 时钟值(MIPI 摄像头传感器时钟通道的速度)为 300 MHz,且传感器在 4 条通道上传输数据,每个通道的数据传输速率为 600 MHz。从而,总的数据传输速率为 2400 MHz。对于每像素 10 位的 Bayer 数据,这相当于 op\_pixel\_clk 值为 2400/10 = 240 MHz。这些值必须按照传感器技术规范赋值。这些值可基于为摄像头传感器配置的寄存器设置进行计算。

#### 4.1.3.3 CSID相关

现象：打开canera黑屏后退出。

概率：100%

原因：某一路的MIPI传输没有SOT或EOT响应。

分析：在msm\_csid.c文件中打开相应的log，抓取log查看“CSID\_IRQ\_STATUS\_ADDR”的关键字，根据如下的解释判断是否有响应的响应：

bit 11: INFO\_RST\_DONE,bit 9: INFO\_SHORT\_PKT\_CAPTURED, bit 0-3: EOT on data lane 0-3, bit 4-7: SOT on data lane 0-3.

Log：

**[**32m**[**75.847195**]** **[**31mmsm\_csid\_irq CSID0\_IRQ\_STATUS\_ADDR **=** 0x200022

**[**32m**[**75.877222**]** **[**31mmsm\_csid\_irq CSID0\_IRQ\_STATUS\_ADDR **=** 0x200022

**[**32m**[**75.877269**]** **[**31mmsm\_csid\_irq CSID0\_IRQ\_STATUS\_ADDR **=** 0x200022

**[**32m**[**75.881645**]** **[**31mmsm\_csid\_irq CSID0\_IRQ\_STATUS\_ADDR **=** 0x200022

**[**32m**[**75.910774**]** **[**31mmsm\_csid\_irq CSID0\_IRQ\_STATUS\_ADDR **=** 0x200022

**[**32m**[**75.910867**]** **[**31mmsm\_csid\_irq CSID0\_IRQ\_STATUS\_ADDR **=** 0x200022

**[**32m**[**75.915203**]** **[**31mmsm\_csid\_irq CSID0\_IRQ\_STATUS\_ADDR **=** 0x200022

解决：如果没有对应的响应，联系sensor的硬件工程师排查。

#### 4.1.3.4 Settle\_cnt相关问题

现象：打开canera黑屏后退出。

概率：100%

原因：settle\_cnt的配置不符合要求。

分析：settle\_cnt配置错误，查看log和dmesg中是否有SOF freeze报错，如下：

Log:

08**:**50**:**27.422 813 3739 E mm**-**camera**:** **<**MCT**><**ERROR**>** 94**:** mct\_bus\_sof\_thread\_run**:** Session 4**:** Hinting SOF freeze to happen**.** Sending event to dump info

08**:**50**:**27.424 813 3742 E mm**-**camera**:** **<**MCT**><**ERROR**>** 57**:** server\_debug\_dump\_data\_for\_sof\_freeze**:** opendir clk fails

08**:**50**:**27.419 813 813 W CAM\_mct\_freeze**:** type**=**1400 audit**(**0.0**:**35**):** avc**:** denied **{** read **}** **for** name**=**"clk" dev**=**"debugfs" ino**=**1934 scontext**=**u**:**r**:**hal\_camera\_default**:**s0 tcontext**=**u**:**object\_r**:**debugfs**:**s0 tclass**=**dir permissive**=**0

08**:**50**:**29.923 813 3739 E mm**-**camera**:** **<**MCT**><**ERROR**>** 98**:** mct\_bus\_sof\_thread\_run**:** FATAL Session 4**:** SOF Freeze**!** Sending error message

Dmesg**:**

**[**32m**[**0.100075**]** **[**0mInitializing cgroup subsys freezer

**[**32m**[**51.740542**]** **[**31mmsm\_private\_ioctl**:**Notifying subdevs about potential sof freeze

解决：咨询厂商关于 settle\_cnt值的设置或者按如下的方式进行配置：

settle\_cnt(即稳定计数)– 必须根据传感器输出特性配置该值,以确保传感器的 PHY发送器与 MSM 的 PHY 接收器无障碍同步。

对于 28 nm 以及更小的 MSM 芯片,使用以下公式计算稳定计数: settle\_cnt = T(HS\_SETTLE)\_avg /T(TIMER\_CLK),

其中 T(HS\_SETTLE)\_avg = (T(HS\_SETTLE)\_min + T(HS\_SETTLE)\_max) / 2,如传感器数据表所指示。

对于 45 nm MSM 芯片,使用与 28 nm MSM 芯片相似的公式,其中的 T(TIMER\_CLK)替换为 T(DDR\_CLK)。

– DDR\_CLK 指摄像头传感器的 MIPI CLK 通道的工作频率,该值由通过传感器摄

像头驱动程序设置的摄像头传感器 PLL 配置确定。

– T(DDR\_CLK) 为工作频率等于 DDR\_CLK 时的时钟周期持续时间,

以纳秒为单位表示。例如,DDR\_CLK 200 MHz 的 T(DDR\_CLK)为 (1 \* (10^9)) / (200 \* (10^6)) = 5 ns。

#### 4.1.3.5 预览画面横纹较多

现象：打开canera后，预览和dump的图片有明显的横纹。

概率：100%

原因：一般横纹是由于电源噪声引起的。

分析：联系sensor硬件工程师一起分析，尝试不同电压的输入。

解决：修改输入电压。

#### 4.1.3.6 只能出一帧数据

Phenomenon: Open camera, Camera crash after 5 seconds.

Probability: 100%

Root cause: ISP module lead this issue

Analysis: From error log, VFE overflow error lead camera demon died， Detail log as below:

Kernel log:

**[**32m**[**496.840610**]** **[**33mmsm\_isp\_process\_overflow\_irq**[**31m**:** vfe 1 overflowmask 10**,**bus\_error 80

**[**32m**[**496.840688**]** **[**33mmsm\_vfe47\_axi\_halt**[**31m**:** VFE1 halt **for** recovery**,** blocking 0

**[**32m**[**496.842831**]** **[**31mCAM**-**SMMU cam\_smmu\_check\_vaddr\_in\_range**:**349 Cannot find vaddr**:**0000000000000000 in SMMU**.**\x0a vfe uses invalid virtual address

**[**32m**[**496.842837**]** **[**33mmsm\_vfe\_iommu\_fault\_handler**[**31m**:** fault address is 10052000

**[**32m**[**496.866223**]** **[**31moverflow processed

**[**32m**[**496.866333**]** **[**33mmsm\_isp\_process\_iommu\_page\_fault**[**31m**:** overflow detected during IOMMU

**[**32m**[**496.866429**]** **[**31mCAM**-**SMMU cam\_smmu\_check\_vaddr\_in\_range**:**349 Cannot find vaddr**:**0000000000000000 in SMMU**.**\x0a vfe uses invalid virtual address

**[**32m**[**496.866433**]** **[**33mmsm\_vfe\_iommu\_fault\_handler**[**31m**:** fault address is 10052b00

**[**32m**[**496.866440**]** **[**33mmsm\_isp\_process\_iommu\_page\_fault**[**31m**:** overflow detected during IOMMU

**[**32m**[** 98.911479**]** **[**31mmsm\_private\_ioctl**:**Notifying subdevs about potential sof freeze

Demon log:

07**-**25 03**:**52**:**57.528 816 3543 D mm**-**camera**:** **<**IFACE **><** HIGH**>** 2440**:** iface\_axi\_handle\_bus\_overflow\_error**:** iface\_axi\_handle\_bus\_overflow\_error**:** overflow\_dbg notify bus overflow**!**

07**-**25 03**:**52**:**57.528 816 3543 D mm**-**camera**:** **<**IFACE **><** HIGH**>** 6121**:** iface\_util\_process\_overflow\_recovery**:** iface\_util\_process\_overflow\_recovery**:**6121 overflow detected on VFE1

07**-**25 03**:**52**:**57.528 816 3543 E mm**-**camera**:** **<**IFACE **><**ERROR**>** 6149**:** iface\_util\_process\_overflow\_recovery**:** iface\_util\_process\_overflow\_recovery**:** can not take new recovery request**!** overflow state 107**-**25 03**:**53**:**02.047 816 3546 E mm**-**camera**:** **<**MCT **><**ERROR**>** 98**:** mct\_bus\_sof\_thread\_run**:** FATAL Session 4**:** SOF Freeze**!** Sending error message

07**-**25 03**:**53**:**02.047 816 3546 I mm**-**camera**:** **<**MCT **><** INFO**>** 133**:** mct\_bus\_sof\_thread\_run**:** Sending HW\_ERROR from MCT on session **=**4

07**-**25 03**:**53**:**02.048 816 3502 E mm**-**camera**:** **<**MCT **><**ERROR**>** 814**:** mct\_controller\_send\_cb**:** FATAL**:** Sending HW\_Error

07**-**25 03**:**53**:**02.049 816 3523 E QCamera **:** **<**HAL**><**ERROR**>** camEvtHandle**:** 807**:** Fatal**,** camera daemon died

We can dump just one raw frame from HAL, We can also dump multiple frames when enlarge the sensor\_num\_frame\_skip in sensor driver, but the sensor\_num\_frame\_skip is limited to 30 by platform.

#define MAX\_INIT\_FRAME\_DROP 31

**if** **(**stream\_cfg\_cmd**->**init\_frame\_drop **>=** MAX\_INIT\_FRAME\_DROP**)**

**{**

pr\_err**(**"%s: Invalid skip pattern\n"**,** \_\_func\_\_**);**

**return** rc**;**

**}**

From kernel, we can get that the first frame and the skip frame is no eof check.(/\*If frame\_id = 1 then no eof check is needed\*/)

VFE is related to isp module, TOF is just need raw data, check log, Found Analysis stream and callback stream created by itself.

Line 15042**:** 07**-**25 03**:**52**:**56.222 816 816 I QCamera **:** **<**HAL**><**INFO**>** processCaptureRequest**:** 4965**:** STREAM INFO **:** type 8**,** wxh**:** 328 x 744**,** pp\_mask**:** 0x0**,** Format**:**0 is\_type**:** 0 sync\_type 1 **--**Raw Stream

Line 15043**:** 07**-**25 03**:**52**:**56.222 816 816 I QCamera **:** **<**HAL**><**INFO**>** processCaptureRequest**:** 4965**:** STREAM INFO **:** type 11**,** wxh**:** 640 x 480**,** pp\_mask**:** 0x10100068e**,** Format**:**0 is\_type**:** 0 sync\_type 1 **--**Analysis Stream

Line 15044**:** 07**-**25 03**:**52**:**56.222 816 816 I QCamera **:** **<**HAL**><**INFO**>** processCaptureRequest**:** 4965**:** STREAM INFO **:** type 5**,** wxh**:** 640 x 480**,** pp\_mask**:** 0x100068e**,** Format**:**0 is\_type**:** 0 sync\_type 1 **--**Callback Stream

Raw**,** Analysis**,** callback stream modules linking as below**:**

raw\_bayer\_mod  **{**"sensor"**,**"iface"**,**"isp"**}**

analysis\_bayer\_mod**->** **{**"sensor"**,**"iface"**,**"isp"**,**"pproc"**,**"imglib"**}**

callback\_bayer\_mod**->** **{**"sensor"**,**"iface"**,**"isp"**,**"pproc"**,**"imglib"**}**

Try to remove stream as below, find that callback stream leads this issue.

a. Raw/RDI0 --OK

b. Raw/RDI0+analysis/Pix CAMIF --OK

c. Raw/RDI0+callback/Pix CAMIF --Fail

d. Raw/RDI0+analysis/Pix CAMIF+callback/Pix VIEW --Fail

From log, the differences between callback stream and analysis stream is that callback stream creates one more stats stream(MSM\_ISP\_STATS\_HDR\_BE).

08**-**17 08**:**39**:**52.762 812 3270 D mm**-**camera**:** **<**IFACE **><** DBG**>** 10166**:** iface\_util\_config\_axi**:** iface\_util\_config\_axi**:** stats\_stream found**:** stats\_stream id **=** 8001000c**,** stats type **=** 12**,** buf\_len **=** 73728**,** comp flag **=** 1

Try to remove HDR\_BE module, no result.

Try to remove isp function that update isp hardware configuration as below, Issue resolved.

.pipeline\_update\_module\_cfg = isp\_pipeline48\_update\_module\_cfg,

As above, we can locate It is isp module issue.

Resolution:

Remove callback stream or remove isp hardware configuration function:

//.pipeline\_update\_module\_cfg = isp\_pipeline48\_update\_module\_cfg.

### **4.1.4黑屏Issue**

This Black Screen Issue is Involved hardware and software.

#### 4.1.4.1 Hardware issue

Phenomenon: Open camera, Screen is black, then camera crash.

Probability: 6/10

Root cause: FPC Welding line

Analysis: From log, we got sof freeze error, No isp notify irq, locate hardware issue.

Error log:

01**-**01 00**:**04**:**59.978 1075 4025 E mm**-**camera**:** **<**MCT **><**ERROR**>** 94**:** mct\_bus\_sof\_thread\_run**:** Session 4**:** Hinting SOF freeze to happen**.** Sending event to dump info

01**-**01 00**:**04**:**59.982 1075 4056 E mm**-**camera**:** **<**MCT **><**ERROR**>** 57**:** server\_debug\_dump\_data\_for\_sof\_freeze**:** opendir clk fails

01**-**01 00**:**05**:**02.481 1075 4025 E mm**-**camera**:** **<**MCT **><**ERROR**>** 98**:** mct\_bus\_sof\_thread\_run**:** FATAL Session 4**:** SOF Freeze**!** Sending error message

01**-**01 00**:**05**:**02.482 1075 4025 I mm**-**camera**:** **<**MCT **><** INFO**>** 133**:** mct\_bus\_sof\_thread\_run**:** Sending HW\_ERROR from MCT on session **=**4

01**-**01 00**:**05**:**04.084 1075 3909 I mm**-**camera**:** **<**MCT **><** INFO**>** 188**:** stop\_sof\_check\_thread**:** Stopping SOF timeout thread session **=**4

01**-**01 00**:**05**:**04.179 1075 2444 E mm**-**camera**:** **<**MCT **><**ERROR**>** 179**:** stop\_sof\_check\_thread**:** Returning as SOF timer thread not yet initialized

01**-**01 00**:**05**:**15.319 1075 4165 I mm**-**camera**:** **<**IFACE **><** INFO**>** 9507**:** iface\_util\_dump\_camif\_cfg**:** camif irq subsample pattern **=** 0**,** period **=** 0 sof\_step 1

01**-**01 00**:**05**:**15.422 1075 4165 I mm**-**camera**:** **<**MCT **><** INFO**>** 156**:** start\_sof\_check\_thread**:** Starting SOF timeout thread session id **=**4

Resolution: Making new fpc.

#### 4.1.4.2 Reading register issue

Phenomenon: Open camera, Screen is black.

Probability: 5/10

Root cause: reading register error

Analysis: Raw data is black, Locate sensor problem. Reading register instability is a known issue.

Resolution:

Removing reading register action, writing the final register data to the sensor when stream on and stream off.

#### 4.1.4.3 App issue

Phenomenon: Open camera, Screen is black when in depth mode.

Probability: 4/10

Root cause: Depth frame is always waitting for ir or irbg frame in APP layer.

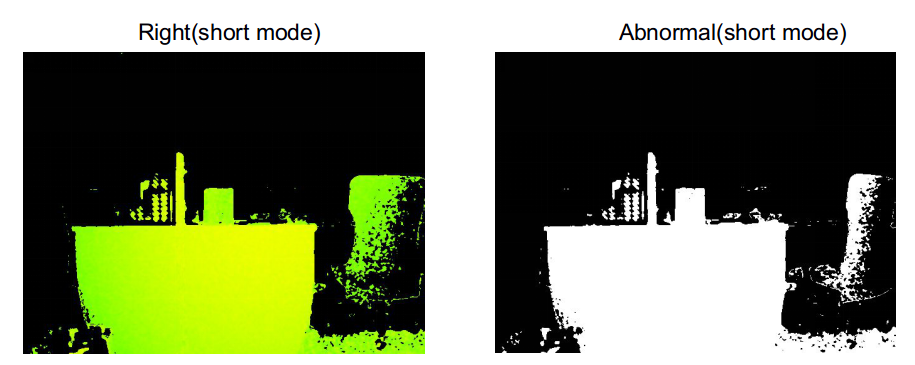
Analysis: Compare apk1(No AI) with apk2(has AI), full screen black not reproduce in apk1, locate apk issue.

Resolution: Removing waiting the timestamp action.

#### 4.1.4.4 HAL issue

Wrong test environment made us mistake this issue for a black screen issue before.

Phenomenon: The depth images sometimes are abnormal.



Probability: 3/10

Background:

Two streams (threads) would be created for getting raw 12 images from kernel, one for DEPTH images and the other for IR\_BG images.

These raw 12 images would be converted to be raw 16 images and then transferred to enhance algorithm.

Different depth\_unit values should be used for the depth images conversion according to current mode (long or short).

Root Cause:

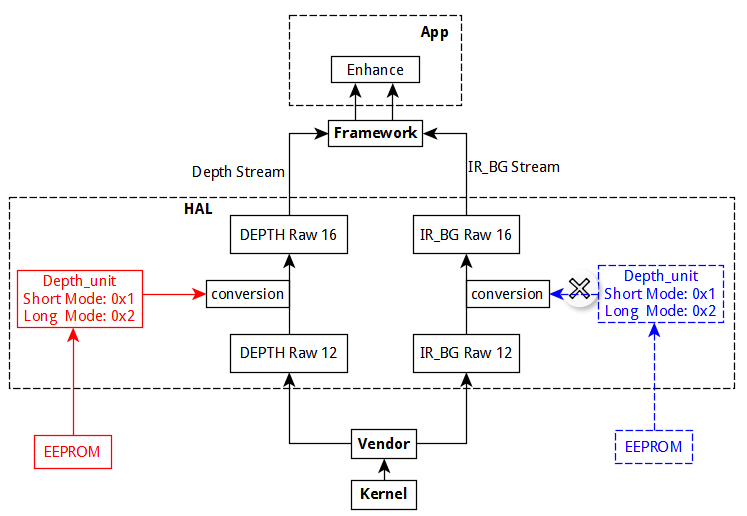
In our codes, the depth\_unit is set in Depth stream or IR\_BG stream randomly. It should be set in Depth stream all the time.

Analysis:

In abnormal cases, depth\_unit is set in IR\_BG stream, while Depth stream uses default value(=0) for depth images conversion.

Resolution:

Setting depth\_unit in Depth stream all the time.



## 4.2 EEPROM PORTING ISSUE

## 4.3 ACTUATOR PORTING ISSUE

### **4.3.1 DAC读写**

#### 4.3.1.1 寄存器数据使能问题

现象：触发自动对焦后，交替出现对焦清晰和。

概率：100%

原因：DAC配置了三组寄存器，两组数据寄存器，一组使能寄存器，使能寄存器和数据寄存器的顺序出现问题，导致写入的数据在下一次的对焦中生效，使得对焦清晰和模糊交替出现。

分析：在马达驱动中将写入的数据打印出来，发现数据的顺序不对（需要FAE一起分析）。

解决：修改DAC寄存器顺序。

## 4.4 OIS PORTING ISSUE

## 4.5 PDAF PORTING ISSUE

## 4.6 FLASH PORTING ISSUE