

Rockchip Camera Module OTP Calibration Guide

文件标识: RK-SM-YF-607

发布版本: V1.0.34

日期: 2023-10-08

文件密级: ☐绝密 ☐秘密 ☐内部资料 ☒公开

免责声明

本文档按“现状”提供, 瑞芯微电子股份有限公司(“本公司”, 下同)不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因, 本文档将可能在未经任何通知的情况下, 不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标, 归本公司所有。

本文档可能提及的其他所有注册商标或商标, 由其各自拥有者所有。

版权所有 © 2020 瑞芯微电子股份有限公司

超越合理使用范畴, 非经本公司书面许可, 任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部, 并不得以任何形式传播。

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址: 福建省福州市铜盘路软件园A区18号

网址: www.rock-chips.com

客户服务电话: +86-4007-700-590

客户服务传真: +86-591-83951833

客户服务邮箱: fae@rock-chips.com

前言

概述

本文旨在介绍相机模组的OTP标定流程, 指导模组厂进行正确的OTP标定工作。

产品版本

芯片名称

RV1109/RV1126/RK356X/RK3588

读者对象

本文档（本指南）主要适用于以下工程师：

模组厂相关的生产与调试工程师

ISP调试工程师

图像质量调试工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	陈煜	2021-07-12	发布初版，提供标定流程、数据封装格式的描述
v1.0.1	徐苏皖	2021-07-26	提供校准方案、示例代码
v1.0.2	徐苏皖	2021-07-27	修改LSC管控的函数接口、示例代码
v1.0.3	徐苏皖	2021-08-03	删除LSC管控的函数， 添加LSC验证接口及管控指标，修改示例代码
v1.0.4	徐苏皖	2021-08-06	添加pdaf gainmap 和dccmap 标定及验证功能
v1.0.5	徐苏皖	2021-08-11	修改AWB标定接口，输出四个通道均值
v1.0.6	徐苏皖	2021-08-16	OTP烧录数据添加full width、full height
v1.0.7	徐苏皖	2021-08-26	修改了OTP烧录数据的格式 以块的格式排列
v1.0.8	徐苏皖	2021-08-27	更新了pdaf 标定的相关内容及示例代码
v1.0.9	徐苏皖	2021-09-17	优化了pdaf 清晰度评价算法，更新pdaf dll
v1.0.10	徐苏皖	2021-10-12	OTP烧录格式添加ROCKCHIP 标识
v1.0.11	徐苏皖	2022-03-29	OTP MAP AF Code 模块添加中焦code
v1.0.12	徐苏皖	2022-04-15	更新了pdaf 标定部分，增加了坏点检测功能
v1.0.13	徐苏皖	2022-05-06	优化了pdaf dccmap 标定算法，修改dccmode为1
v1.0.14	徐苏皖	2022-05-13	优化了pdaf gainmap验证算法
v1.0.15	徐苏皖	2022-05-16	更新otp map pdaf 部分
v1.0.16	徐苏皖	2022-05-18	优化了pdaf gainmap、dccmap算法， 调整了gainmap 和dccmap每个block的大小
v1.0.17	徐苏皖	2022-05-23	调整pdaf dccmap 标定的阈值，增加log打印

版本号	作者	修改日期	修改说明
v1.0.18	徐苏皖, 黄若冰	2022-06-01	修改PDAF 标定流程的相关描述, 优化PDAF Dccmap标定及验证算法
v1.0.19	徐苏皖, 黄若冰	2022-06-01	调整pdaf dcc sharpness拍摄步长, 修改sharpness验证算法
V1.0.20	徐苏皖	2022-06-08	更新pdaf 计算清晰度算法, 调整otp gainmap和dccmap的内存大小
V1.0.21	徐苏皖	2022-06-20	优化pdaf计算清晰度算法
V1.0.22	徐苏皖	2022-07-07	优化pdaf计算清晰度算法, 增加DCC_SHARPNESS_COUNT 参数
V1.0.23	徐苏皖	2022-08-05	OTP map增加结束标志
v1.0.24	徐苏皖	2022-09-15	修改AWB标定方案, 调整LSC 验证的处理流程, 优化pdaf计算清晰度算法, 增加DCC_SHARPNESS_CHL、SENSOR_TYPE参数
v1.0.25	黄若冰, 徐苏皖	2022-11-08	LSC标定接口开放标定力度参数vig, 增加了sensor.ini相关参数定义, 增加了针对dual pd的sensor.ini定义
v1.0.26	徐苏皖	2022-11-28	调整DCCMap 标定数据 亮度管控的阈值
v1.0.27	徐苏皖	2022-12-06	更新了pdaf 清晰度算法
v1.0.28	徐苏皖	2022-12-08	针对lsc 四个暗角是否对称, LSC标定接口增加阈值卡控lsc 标定结果
v1.0.29	黄若冰	2023-04-28	增加了 all-pixel-ocl vbin类型sensor的标定
v1.0.30	徐苏皖	2023-05-08	修改了DCCMAP_BLKSZ_W的数据类型
V1.0.31	黄若冰	2023-06-02	增加了针对DUAL_PD类型的数据MIPI传输配置
V1.0.32	黄若冰	2023-07-06	更改了DUAL_PD类型NORMAL像素的计算方式
V1.0.33	黄若冰	2023-08-16	增加了PD_OFFSET的标定
V1.0.34	徐苏皖	2023-10-08	OTP map增加QSC项

目录

Rockchip Camera Module OTP Calibration Guide

- 1 LSC&AWB 标定流程
 - 1.1 标定方案
 - 1.2 管控方案
 - 2 PDAF 标定流程
 - 2.1 标定前准备
 - 2.2 Gain Map 标定
 - 2.3 DCC Map 标定
 - 3 标定方案
 - 3.1 AWB 标定
 - 3.2 LSC 标定
 - 3.2.1 LSC 标定
 - 3.2.2 LSC 验证
 - 3.3 PDAF 标定
 - 3.3.1 Get sensor config
 - 3.3.2 Gain Map 标定
 - 3.3.3 Gain Map 验证
 - 3.3.4 DCC Map 标定
 - 3.3.5 DCC Map 验证
 - 4 附录
 - 4.1 OTP 应用示例代码
 - 4.2 烧录打包数据格式
-

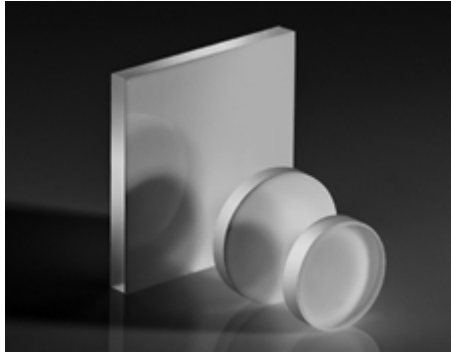
1 LSC&AWB 标定流程

1.1 标定方案

1. 光源环境与器材

D50 (5000K±100K) , 避免其他环境杂散光干扰。

均光片或满足上述光源的DNP灯箱, 均光片如下图所示:



2. 拍摄方式

1. 根据实际情况, 以下两个方法选择一种即可
 1. 使用均光片: 将均光片带涂层的一面朝向镜头紧贴平放, 保持模组端面、均光片、光源面板平行。
 2. 使用DNP灯箱: 将模组置于DNP光源面板前 1~2cm, 保持模组端面和光源面板平行。
2. 关闭mirror/flip等功能
3. 挑选曝光值, 让图像的G通道最大亮度 (8bit) 达到160~180之间
4. 使用该曝光拍摄Raw图并保存

3. AWB标定

1. 选择Raw图中心区域 (各取长宽的20%) , 计算R、Gr、Gb、B各通道的均值

$Gr_ave = Gr \text{ average of ROI} - \text{Black level}$

$Gb_ave = Gb \text{ average of ROI} - \text{Black level}$

$R_ave = Red \text{ average of ROI} - \text{Black level}$

$B_ave = Blue \text{ average of ROI} - \text{Black level}$

2. 计算R/Gb和B/Gb和Gr/Gb

$R/Gb = R_ave / Gb_ave$

$B/Gb = B_ave / Gb_ave$

$Gr/Gb = Gr_ave / Gb_ave$

3. 将R/Gb、B/Gb、Gr/Gb定点化至10bit

$R/Gb_hex = R/Gb * 1024$

$B/Gb_hex = B/Gb * 1024$

$Gr/Gb_hex = Gr/Gb * 1024$

4. LSC标定

1. 标定计算的目标值设置为70%
2. 分别计算四个通道的增益定点化至10bit

1.2 管控方案

1. LSC标定前

1. Y shading标准：

$$ROI = 1/5 \text{ Width} * 1/5 \text{ Height}$$

$$YShading_Corner = Y_Corner/Y_Center$$

$$30\% < YShading_Corner < 55\%$$

$$Ydiffer = YShading_Corner_Max - YShading_Corner_Min < 7\%$$

2. Color Shading 标准：

单颗模组color shading 均匀性（单颗模组每个block与中心block的差异管控）

ROI取框：1/5*1/5 计算24个block与中心block的差异

$$ROI = 1/5 \text{ Width} * 1/5 \text{ Height}$$

$$|(R/G_Corner)/(R/G_Center)-1| < 15\%$$

$$|(B/G_Corner)/(B/G_Center)-1| < 15\%$$

2. LSC标定

对于因为模组摆放问题导致抓取的raw数据上四周暗角不对称的问题，lsc_otp_Calibrate接口增加 ratio 参数卡控标定结果（默认ratio=1.5）；

若因模组本身组装导致的光心偏移，可将ratio调大，提高通过率。

3. LSC标定后

1. Y shading标准：

$$ROI = 1/5 \text{ Width} * 1/5 \text{ Height}$$

$$YShading_Corner = Y_Corner/Y_Center$$

$$Ydiffer = YShading_Corner_Max - YShading_Corner_Min < 5\%$$

2. Color Shading 标准：

单颗模组color shading 均匀性（单颗模组每个block与中心block的差异管控）

ROI取框：1/5*1/5 计算24个block与中心block的差异

$$ROI = 1/5 \text{ Width} * 1/5 \text{ Height}$$

$$|(R/G_Corner)/(R/G_Center)-1| < 5\%$$

$$|(B/G_Corner)/(B/G_Center)-1| < 5\%$$

2 PDAF 标定流程

PDAF标定分为 Gain Map标定 和 DCC Map 标定两部分，标定流程如下：

2.1 标定前准备

1、模组准备

- (1) 镜头脏污检测
- (2) VCM检测：测试线性、镜头的位置准确性和稳定性
- (3) AF基础校正：

确定DAC和对焦物距的关系-----标定模组镜头的马达在对焦Inf处和Marco处的位置，记录其对应的DAC_inf和Dac_marco值，确定DCC_LENS_BEGIN和DCC_LENS_END，当马达中置位置对焦最清晰时的对应物距为DCC Map标定物距 a；

马达中置位置DAC = (DAC_INF + DAC_MACRO)/2，对应标定物距a可以查询Lens景深表来确定。首先在lens景深表中确定inf和macro的lens shift，计算两者的中间距离，从景深表查找离中间距离最近的lens shift，即马达中置位置和对应的物距a。例如物距无限远时对应lens shift为0，最近物距选取10cm，对应lens shift为-0.183，则马达中置位置lens shift为-0.0915，和-0.091最接近，查表可知对应的DCC Map标定物距 a则为0.20米。

Object Distance (m)	Lens Shift (mm)	Far Field (m)	Near Field (m)
INF	0.000	-	-
10.0	-0.002	INF	4.33
7.6	-0.002	14417.449	3.81
5.0	-0.004	14.511	3.02
4.0	-0.004	8.406	2.63
3.0	-0.006	4.941	2.15
2.0	-0.009	2.71	1.59
1.9	-0.009	2.53	1.52
1.8	-0.010	2.35	1.46
1.7	-0.010	2.19	1.39
1.6	-0.011	2.02	1.32
1.5	-0.012	1.87	1.25
1.4	-0.013	1.71	1.18
1.3	-0.014	1.57	1.11
1.2	-0.015	1.42	1.04
1.1	-0.016	1.28	0.96
1.0	-0.018	1.15	0.88
0.9	-0.020	1.02	0.81
0.8	-0.022	0.89	0.72
0.7	-0.025	0.77	0.64
0.6	-0.029	0.65	0.56
0.5	-0.035	0.53	0.47
0.45	-0.039	0.478	0.425
0.40	-0.044	0.422	0.380
0.35	-0.051	0.366	0.335
0.30	-0.059	0.312	0.289
0.25	-0.071	0.258	0.242
0.20	-0.091	0.202	0.192
0.15	-0.120	0.153	0.147
0.14	-0.129	0.142	0.138

Object Distance (m)	Lens Shift (mm)	Far Field (m)	Near Field (m)
0.13	-0.140	0.132	0.128
0.12	-0.152	0.122	0.118
0.11	-0.166	0.111	0.109
0.10	-0.183	0.101	0.099
0.09	-0.205	0.091	0.089
0.08	-0.232	0.081	0.079
0.07	-0.267	0.071	0.069
0.06	-0.315	0.060	0.060
0.05	-0.383	0.050	0.050

Item	Detail	Note
测试图	厂商定义	建议棋盘格、diamond-chart等有高对比度，频率适中的测试图
测试图面照明亮度	大于400Lux	成像不可有flicker闪动
模组与测试图距离	Inf:2~5m Marco:10-20cm(具体参考景深表)	Inf距离可使用增距镜模拟远焦位置
模组校正方向	Inf/Macro校正时，模组光轴呈水平	若为open loop VCM,仅水平方向校正；若为Closed loop VCM可光轴朝上校正，注意Inf/Macro校正方向相同
Gain设置	同AWB	数字增益固定设置为1；模拟增益（传统PD或2x1OCL，Again设为1x;Dual-photodiode的PD,Again设定为2x）
Sensor设置	关闭sensor中的mirror/flip/OB/去坏点等设置	
影像亮度要求	同AWB	全尺寸显示时画面中心1/32W*1/32H的ROI区域的TOP 10%G通道的平均亮度值为 800LSB(raw10); dual pd G通道的均值为 700LSB(raw10);
误差检验样本数	范围检验：每个模组都要 精确度检验：大于10pcs	若不通过检验，则需修正校正方法，使得AF远近焦校正结果更精确

(4) 若模组为OIS-enabled类型，在标定lsc和pdaf前需先完成相机抖动校正。将马达移动到参考XY位置验证OIS 已经配置好。该步骤将生成平场图像用于lsc和pdaf标定；

(5) 完成黑电平和LSC校正，必要时可对场曲进行校正

2、模组AE设置

(1) 模组AE设置：数字增益固定设置为1；模拟增益（传统PD或2x1OCL，Again设为1x；Dual-photodiode的PD,Again设定为2x，防止出现pixel blooming现象）

(2) 曝光时间：为了防止dual-pd的charge blooming现象以及正常像素过曝（regular_pixel=dual_pd_l+dual_pd_r），全尺寸显示时，调整曝光时间，使得中心区域的G通道亮度为700(raw10)；对于OCL和metal-shield类型,保证中心区域 G通道亮度为 800(raw10)。sensor建议曝光时间设置为10ms的整数倍以避免工频干扰。

2.2 Gain Map 标定

1、光源环境与器材

D50（5000K±100K），避免其他环境杂散光干扰；

均光片或满足上述光源的DNP灯箱，积分球；

2、拍摄方式

1. 将模组放置在均匀面光源前，模组端面与光源面保持平行（eg.实验时采用的积分球，面板亮度数值调整为6.6），均光片置于镜头前端，使得面光源充满画面；
2. sensor的gain-对于 metal-shield和2x1OCL类型的sensor的again和dgain设置为1x，dual-pd类型的sensor，again设置为2x，dgain设置为1x，曝光时间设置为10的倍数；
3. 关闭sensor中的mirro/flip/OB等设置
4. 全尺寸显示时画面中心1/32w*1/32H的ROI区域的TOP 10%G通道的平均亮度值为 800LSB(raw10)或200LSB (raw8);dual pd G通道的均值为 700LSB(raw10)或175LSB (raw8)；
5. 打开camera预热3分钟，将马达移动到中置位置连续拍摄3张raw图，首先检测坏点个数是否超过允许的范围，若坏点个数超过允许范围，则需要降低画面亮度重新拍摄；若坏点检测通过，再得到平均图像数据；

3、标定

调用pdaf_gainmap_calibration(...)函数生成gainmap数据并获取gainmap 的大小，大小为gainmap_width*gainmap_height *2 byte（包含left gainmap 和 right gainmap）；

4、验证

1. 保证步骤2环境相同，拍摄一张测试图像，将标定得到的gainmap和测试图像输入到函数pdaf_gainmap_verification(...)，若返回为1，则验证成功，标定的gainmap可用；若返回为-1，则表示坏点个数是否超过允许的范围，需要降低画面亮度重新拍摄；

2. 验证标准

经gainmap校正后的left pd pixel 与right pd pixel 的差值小于5%，则验证成功；

2.3 DCC Map 标定

DCC(defocus conversion coefficient)表示马达位置与左右图像差的关系，将图像分成不同区域，将马达等步长移动，计算不同位置下的pd值，利用线性回归得到不同区域的DCC值---DCCMAP

$$DCC = \Delta lensposition[dac] / \Delta phasedisparity[pixel]$$

[optional]

对于部分模组，需要标定PD_OFFSET。例如：当马达为DAC_0时获得一张对焦模糊的图，计算出来的清晰对焦DAC=DAC_0+PD*DCC+PD_OFFSET *DCC

1、光源环境与器材

D50（5000K±100K），避免其他环境杂散光干扰。

均光片或满足上述光源的DNP灯箱，积分球，黑白相间标版；

标板：标板采用定制的透光菲林片，图案为等间隔的黑白条纹或者菱形格。

标定距离a在2.1.1 AF基础校正步骤中确定

标板宽度取决于镜头水平方向FOV和标定距离a，建议满足

$$S \geq a * \tan(FOV * \pi / 360) * 2 / 0.9$$

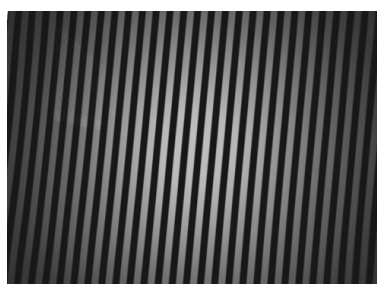
标板的黑白线宽W，镜头焦距f,标定距离a,像素尺寸u,有效像素水平方向raw_width,水平方向DCC_MAP大小dccmap_width之间的关系建议满足

$$\lambda[mm] = \frac{u[um]}{1000} * \frac{raw_width}{6 * dccmap_width}$$

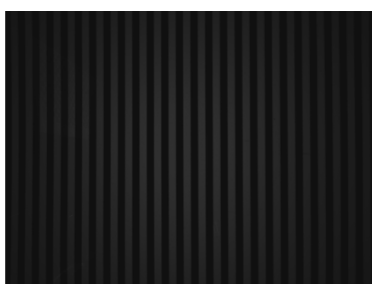
$$W[mm] = (a[mm] - f[mm]) / f[mm] * \lambda[mm]$$

对于IMX258,系数λ建议为0.11；对于S5KJN1,系数λ建议为0.088~0.099。

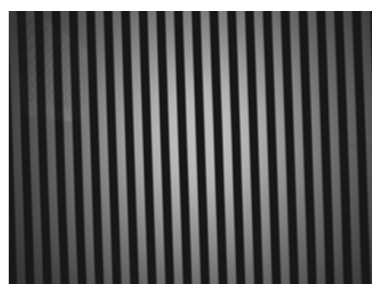
DCC标定要求正确定制并摆放标板，条纹尺寸不合适、摆放时发生旋转或变形、亮度过暗或过曝都会导致DCC标定失败，一些错误案例如下图所示。



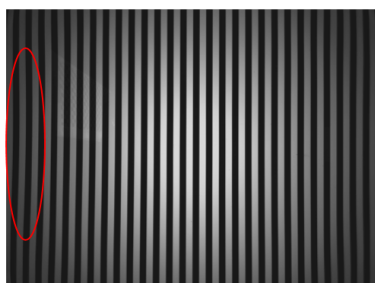
(a) 旋转



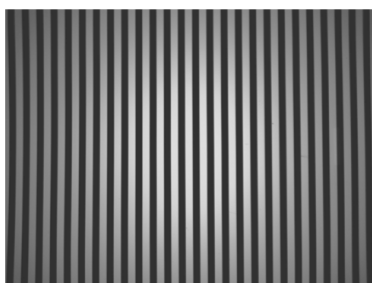
(b) 亮度过暗



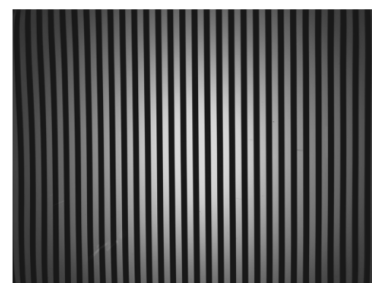
(c) 条纹过宽



(d) 反光与形变



(e) 过曝



(f) 条纹过密

2、拍摄方式

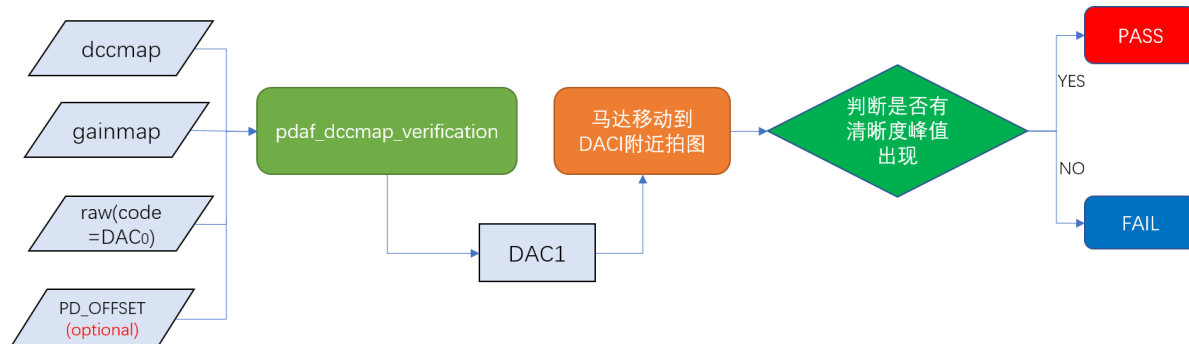
1. 将条纹菲林片放置于积分球前，模组放置于标版前方约对焦中段位置，该物距值由步骤2.1.1AF基础校正中确定。端面与标版保持平行，避免标板旋转、倾斜和扭曲，保证拍摄时标版充满图像；
2. sensor gain: 对于 metal-shield和2x10CL类型的sensor的again和dgain设置为1x, dual-pd类型的sensor, again设置为2x, dgain设置为1x;
3. 关闭sensor中的mirro/flip/OB/去坏点等设置;
4. 曝光: 保证标版画面中心1/32w*1/32h的ROI区域的白块g通道均值为800LSB(raw10), dual_pd类型的G通道 (L+R) 的均值为 700LSB(raw10), 避免亮度不够或过曝影响DCC标定精度; 对于metal-shield类型sensor, 保证non-shielded的像素没有过曝;
5. 将马达由DCC_LENS_BEGIN到DCC_LENS_END以DCC_LENS_INTERVAL采样间隔移动, 每移动一次待马达稳定后拍摄三张raw图, 得到每个位置的平均图像数据 (eg. 实验时以8为间隔, code从0移动到64共计9个位置, 拍摄了27张图像)

3、标定

将2.1节标定得到的gainmap、马达位置参数和每个位置的平均图像数据输入到函数 `pdaf_dccmap_calibration(...)`，得到dccmap 数据并获取dccmap 的大小，大小为`dccmap_width*dccmap_height *2 byte`（若选择标定PD_OFFSET,则同时输出该值大小为2bytes）；利用 `DCC_RSQ_THRESHOLD`评价每个Block拟合出来的系数线性度，如果不满足线性度阈值的Block大于一定数量，则标定失败，需要检查输入数据是否有误，调整标定环境重新拍摄。

4、验证

该步骤在所有步骤之后，目的是验证当前标定的dccmap是否正确 验证流程如下图所示：



1. 验证数据拍摄光照环境与DCCmap标定环境相同，镜头面与标板保持平行，保证拍摄时图像被标板图案充满；

2. 打开camera预热3分钟，调整马达至一个**成像模糊**的位置DAC0，

$$DAC_0 \in (DAC_{inf} + 0.2 * abs(DAC_{inf} - DAC_{macro}), DAC_{inf} + 0.8 * abs(DAC_{inf} - DAC_{macro}))]$$

待马达稳定后，抓取三张Raw图，得到平均图像数据将马达当前位置信息DAC0、平均图像数据以及标定得到的gainmap和dccmap输入到`pdaf_dccmap_verification()` 函数，计算得到对焦模糊处图像的理想对焦位置DAC1；

3. 然后将马达移动到DAC1，抓取在DAC1前后 **$tolerance_factor * abs(DAC_{inf} - DAC_{macro}) + \Delta$** （e.g. $\Delta=2$ ）内的五张图，`tolerance_factor`为验证时的容忍范围，不同类型的sensor该误差范围也不同(e.g.5%,10%,20%)。

验证时将马达从位置1移动到位置5，五个位置分别是(e.g.`tolerance_factor=10%, $\Delta=2$`)

1	$DAC1 - 10\% * abs(DAC_{inf} - DAC_{macro}) - \Delta$
2	$DAC1 - 10\% * abs(DAC_{inf} - DAC_{macro})$
3	DAC1
4	$DAC1 + 10\% * abs(DAC_{inf} - DAC_{macro})$
5	$DAC1 + 10\% * abs(DAC_{inf} - DAC_{macro}) + \Delta$

4. 将五张图传入到函数`pdaf_calc_sharpness(...)` 计算图像中心区域 `DCC_PERCENT_ROI_W* DCC_PERCENT_ROI_H`的清晰度(e.g.30%*30%)，判断是否有峰值出现，有则验证成功;若验证失败，放大`tolerance_factor`和 Δ 的范围（e.g. Δ 由2调为5），重新采集5张图，重复步骤4。计算清晰度最好选择pd像素比较少的通道。

3 标定方案

标定的模块包括 AWB(Auto White Balance)、 LSC(Lens Shading Correction)、 PDAF(Phase Detection Auto-focus)。

3.1 AWB 标定

【功能描述】

该函数用于计算 raw 图像中 ROI 的 R/G, B/G Gr/Gb。

【函数原型】

```
bool awb_otp_Calibrate(uint16_t* RawAddr, int width, int height, int roiw,
                      int roih, int bits, int bayer, int blc_r, int blc_gr,
                      int blc_gb, int blc_b, uint16_t* R, uint16_t* Gr,
                      uint16_t*Gb, uint16_t* B, uint16_t* R_G, uint16_t* B_G,
                      uint16_t* Gr_Gb)
```

【输入信息】

参数名称	描述
RawAddr	当前模组在产线拍摄的 raw 图像数据
width	raw 图像的宽度
height	raw 图像的高度
bits	每个pixel 的bit 数。10bit=10
roiw	ROI区域的宽度与图像宽度的比值的倒数
roih	ROI区域的高度与图像高度的比值的倒数
bayer	raw 图像的 Bayer 模式。 BGGR=0, GBRG=1, GRBG=2, RGGB=3
blc_r	R 通道 BLC
blc_gr	Gr 通道 BLC
blc_gb	Gb 通道 BLC
blc_b	B 通道 BLC

【输出信息】

参数名称	描述
R	ROI区域 R 通道均值
Gr	ROI区域 Gr 通道均值
Gb	ROI区域 Gb 通道均值
B	ROI区域 B 通道均值
R_G	ROI区域R通道的均值与Gb通道的均值的比值*1024
B_G	ROI区域B通道的均值与Gb通道的均值的比值*1024
Gr_Gb	ROI区域Gr通道的均值与Gb通道的均值的比值*1024

【返回值】

- =0：参数错误
- =1：标定成功

3.2 LSC 标定

3.2.1 LSC 标定

【功能描述】

该函数用于计算 raw 图像每个通道的增益。

【函数原型】

```
bool lsc_otp_calibrate(int vig,uint16_t* RawAddr, int width, int height,
                      int bits, int bayer, int blc_r, int blc_gr,
                      int blc_gb, int blc_b, float ratio, uint16_t* lsc_otp)
```

【输入信息】

参数名称	描述
vig	LSC校正力度，default vig=70(%)
RawAddr	当前模组在产线拍摄的 raw 图像数据
width	raw 图像的宽度
height	raw 图像的高度
bits	每个pixel的bit数。10bit=10
bayer	raw 图像的 Bayer 模式。BGGR=0，GBRG=1，GRBG=2，RGGB=3
blc_r	R 通道 BLC
blc_gr	Gr 通道 BLC
blc_gb	Gb 通道 BLC
blc_b	B 通道 BLC
ratio	衡量四周暗角是否对称的阈值，建议值ratio=1.5

【输出信息】

参数名称	描述
lsc_otp	LSC 增益： <ul style="list-style-type: none">• R 通道增益 (17 * 17)• Gr 通道增益 (17 * 17)• Gb 通道增益 (17 * 17)• B 通道增益 (17 * 17)

【返回值】

- =0: lsc 标定数据异常，暗角异常
- =1: 标定成功

3.2.2 LSC 验证

【功能描述】

该函数用于验证LSC标定的结果。

【函数原型】

```
bool lsc_otp_verify(uint16_t *RawAddr, int width, int height, int bits,
                    int bayer, int blc_r, int blc_gr, int blc_gb,
                    int blc_b, uint16_t *lsc_otp, int Ydiffer_down,
                    int Ydiffer_up, int ColorShading_down,
                    int ColorShading_up, float *fYdiffer,
                    float *fRGCcorner, float *fBGCcorner)
```

【输入信息】

参数名称	描述
RawAddr	当前模组在产线拍摄的 raw 图像数据
width	raw 图像的宽度
height	raw 图像的高度
bits	每个pixel 的bit 数。10bit=10
bayer	raw 图像的 Bayer 模式。BGGR=0, GBRG=1, GRBG=2, RGGB=3
b1c_r	R 通道 BLC
b1c_gr	Gr 通道 BLC
b1c_gb	Gb 通道 BLC
b1c_b	B 通道 BLC
lsc_otp	由lsc_otp_Calibrate计算输出的lsc_otp
Ydiffer_down	校准后的管控指标 YShading_Corner_Max - YShading_Corner_Min的下限, 百分比
Ydiffer_up	校准后的管控指标YShading_Corner_Max - YShading_Corner_Min的上限, 百分比
ColorShading_down	校准后的管控指标 $ (R/G_Corner)/(R/G_Center)-1 $ 、 $ (B/G_Corner)/(B/G_Center)-1 $ 的下限, 百分比
ColorShading_up	校准后的管控指标 $ (R/G_Corner)/(R/G_Center)-1 $ 、 $ (B/G_Corner)/(B/G_Center)-1 $ 的上限, 百分比

【输出信息】

参数名称	描述
fYdiffer	YShading_Corner_Max - YShading_Corner_Min的值
fRGCORner	每个block $ (R/G_Corner)/(R/G_Center)-1 $ 的值, 共5*5个block
fBGCORner	每个block $ (B/G_Corner)/(B/G_Center)-1 $ 的值, 共5*5个block

【返回值】

- =0: 不满足标定后的管控指标, 验证失败
- =1: 验证成功

3.3 PDAF 标定

3.3.1 Get sensor config

【功能描述】

配置 sensor.ini文件，并从sensor.ini文件中获取pd sensor 相关信息。

【配置内容】

[sensor config]

关键字	描述
RAW_WIDTH	raw image width
RAW_HEIGHT	raw image height
RAW_BITS	bit width of pixel value
RAW_BLACK_LEVEL	black level value
RAW_BAYER_PATTERN	raw image bayer pattern. 0:BGGR, 1:GBRG, 2:GRBG, 3:RGGB
SENSOR_TYPE	0: shieldPD or 1x2OCL 1: horizon and vertical (LR&TB) 2: reserved3:reserved
PD_DUAL_MODE	0:non-dual 1:dual PD
PD_BINNING_TYPE	calibration using binning PD 0:OFF 1:ON
MIPI_MIX_MODE	0:data contains only raw data or pd data 1:data contains both raw data and pd data,useful only when PD_DUAL_MODE=1
PD_OFFSET_X	x offset of PD block
PD_OFFSET_Y	y offset of PD block
PD_PITCH_X	x pitch of PD block
PD_PITCH_Y	y pitch of PD block
PD_DENSITY_X	x interval of 1 pair of L/R PD pixel
PD_DENSITY_Y	y interval of 1 pair of L/R PD pixel
PD_BLOCK_NUM_X	total PD block number in x direction
PD_BLOCK_NUM_Y	total PD block number in y direction
CALB_S_LEVEL	Saturation Level , default is 100
CALB_S_CNT	Saturation Count , default is 10
PD_POS_L	the position of L PD pixel in one PD block
PD_POS_R	the position of R PD pixel in one PD block

dual类型

[sensor config]

关键字	描述
RAW_WIDTH	raw image width
RAW_HEIGHT	raw image height
LR_WIDTH	pd image width
LR_HEIGHT	pd image height
RAW_BITS	bit width of pixel value
RAW_BLACK_LEVEL	black level value
RAW_BAYER_PATTERN	raw image bayer pattern. 0:BGGR, 1:GBRG, 2:GRBG, 3:RGGB
SENSOR_TYPE	0: shieldPD or 1x2OCL 1: horizon and vertical (LR&TB) 2:reseved 3:reserved
PD_DUAL_MODE	0:non-dual 1:dual PD
PD_BINNING_TYPE	calibration using binning PD 0:OFF 1:ON
MIPI_MIX_MODE	0 :data contains only raw data or pd data 1:data contains both raw data and pd data,useful only when PD_DUAL_MODE=1
PD_OFFSET_X	x offset of PD block
PD_OFFSET_Y	y offset of PD block
PD_PITCH_X	x pitch of PD block
PD_PITCH_Y	y pitch of PD block
PD_DENSITY_X	x interval of 1 pair of L/R PD pixel
PD_DENSITY_Y	y interval of 1 pair of L/R PD pixel
PD_BLOCK_NUM_X	total PD block number in x direction
PD_BLOCK_NUM_Y	total PD block number in y direction
CALB_S_LEVEL	Saturation Level , default is 100
CALB_S_CNT	Saturation Count , default is 10
PD_POS_L	the position of L PD pixel in one PD block
PD_POS_R	the position of R PD pixel in one PD block

[Gainmap_Calib]

关键字	描述
GAINMAP_BLKSZ_W	the width of one block of gainmap
GAINMAP_BLKSZ_H	the height of one block of gainmap
GAIN_CALIB_INPUT_MAX	Input raw image level-max :920(raw10)
GAIN_CALIB_INPUT_MIN	Input raw image level-min :800(raw10)
GAIN_VERIFY_DIFF_MAX	L/R after gain difference level(eg:=5 means 5%*1024)
CROSS_VER	using other imgs verify gainmap

[DCCmap_Calib]

关键字	描述
DCCMAP_BLKSZ_W	the width of one block of dccmap
DCCMAP_BLKSZ_H	the height of one block of dccmap
DCC_CALIBRATE_MODE	the mode of dcc calibrate, default is 1
DCC_LENS_BEGIN	the sampling start position of code
DCC_LENS_END	the sampling end position of code
DCC_LENS_INTERVAL	the sampling interval of code
DCC_RSQ_THRESHOLD	the threshold of rsq matrix
DCC_BORDER_RSQ_THRESHOLD	the threshold of rsq border matrix
DCC_BAD_RSQ_NUM_RATIO	the ratio of the number of bad rsq matrix
DCC_CALIB_INPUT_MAX	Input raw image level-max :640(raw10)or160(raw8)
DCC_CALIB_INPUT_MIN	Input raw image level-min :340(raw10)or85(raw8)
DCC_CALIB_PD_RANGE	pixel disparity search radius
DCC_VERIFY_CAF_DIFF_MAX	Threshold (%)fo DAC range
DCC_VERIFY_CAF_DIFF_DET	delta(DAC) for DAC range
CALIB_TARGET_PEAK_OFFSET_LOG	save the fv peak information to PDAFFVLog.txt.1:ON,0:OFF
CALIB_TARGET_PEAK_OFFSET	PD target offset in the fv peak criteria default:0
DCC_PERCENT_ROI_W	the roi width of test image
DCC_PERCENT_ROI_H	the roi height of test image
DCC_SHARPNESS_MODE	the mode of calc sharpness, default is 1
DCC_SHARPNESS_CHL	the channel of calc sharpness,0:origin raw;1:red,2:green(default),3:blue,4:demosaic

[optional params]

关键字	描述
QUALITY_VERIFY_ENABLE	enable quality test on gainmap calibration
QUALTI_LRDiff_L_MIN	L/R Difference criteria(block1,2,3)
QUALTI_LRDiff_L_MAX	L/R Difference criteria(block1,2,3)
QUALTI_LRDiff_C_MIN	L/R Difference criteria(block4,5,6)
QUALTI_LRDiff_C_MAX	L/R Difference criteria(block4,5,6)
QUALTI_LRDiff_R_MIN	L/R Difference criteria(block7,8,9)
QUALTI_LRDiff_R_MAX	L/R Difference criteria(block7,8,9)
QUALITY_SENSITIVITY_MIN	sensitivity criteria

【示例】

以IMX258 pd pixel分布为例，如下图所示：

根据图上的信息，填写的sensor.ini文件内容如下：

```
[sensor config]
RAW_WIDTH =4208
RAW_HEIGHT =3120
RAW_BITS =12
RAW_BLACK_LEVEL =64
RAW_BAYER_PATTERN =0
SENSOR_TYPE=0
PD_DUAL_MODE=0
PD_BINNING_TYPE=0
PD_OFFSET_X =24
PD_OFFSET_Y =24
PD_PITCH_X =32
PD_PITCH_Y =32
PD_DENSITY_X =16
PD_DENSITY_Y =16
PD_BLOCK_NUM_X =130
PD_BLOCK_NUM_Y =96
CALB_S_LEVEL=1000
CALB_S_CNT=10
PD_POS_L=
[26 29]
[42 29]
[33 48]
[49 48];          #end with a ";"
PD_POS_R=
[25 32]
[41 32]
[34 45]
[50 45];          #end with a ";"

[Gainmap_Calib]
```

```
GAINMAP_BLKSZ_W=16
GAINMAP_BLKSZ_H=16
GAIN_CALIB_INPUT_MAX=920
GAIN_CALIB_INPUT_MIN=800
GAIN_VERIFY_DIFF_MAX=5
CROSS_VER=1

[DCCmap_Calib]
DCCMAP_BLKSZ_W=32
DCCMAP_BLKSZ_H=32
DCC_CALIBRATE_MODE=1
DCC_LENS_BEGIN=0
DCC_LENS_END=64
DCC_LENS_INTERVAL=8
DCC_RSQ_THRESHOLD=0.985
DCC_BORDER_RSQ_THRESHOLD=0.97
DCC_BAD_RSQ_NUM_RATIO=0.05
DCC_CALIB_INPUT_MAX=640
DCC_CALIB_INPUT_MIN=340
DCC_VERIFY_CAF_DIFF_MAX=10
DCC_VERIFY_CAF_DIFF_DET=2
CALIB_TARGET_PEAK_OFFSET_LOG=0
CALIB_TARGET_PEAK_OFFSET=0
DCC_PERCENT_ROI_W=0.4
DCC_PERCENT_ROI_H=0.4
DCC_SHARPNESS_MODE=1
DCC_SHARPNESS_CHL=1

[optional params]
QUALITY_VERIFY_ENABLE=1
QUALTI_LRDiff_L_MIN=0.4
QUALTI_LRDiff_L_MAX=2.5
QUALTI_LRDiff_C_MIN=0.55
QUALTI_LRDiff_C_MAX=1.8
QUALTI_LRDiff_R_MIN=0.4
QUALTI_LRDiff_R_MAX=2.5
QUALITY_SENSITIVITY_MIN=0.45
```

【函数原型】

```
bool pda_f_initial(char* filename, sensor_cfg* psensor_cfg)
```

【输入信息】

参数名称	描述
filename	sensor.ini文件的完整路径

【输出信息】

参数名称	描述
psensor_cfg	存储sensor.ini文件内容的结构体指针

【返回值】

- =0: 获取sensor.ini失败
- =1: 获取sensor.ini成功

3.3.2 Gain Map 标定

【功能描述】

该函数用于pdaf gain map的标定。

【函数原型】

```
bool pdaf_gainmap_calibration(uint16_t *RawAddr, sensor_cfg* psensor_cfg,
                             uint16_t *gainmap_lut,int* gainmap_width,
                             int* gainmap_height)
```

【输入信息】

参数名称	描述	
RawAddr	当前模组在产线拍摄的 raw 图像数据	
psensor_cfg	存储sensor.ini文件内容的结构体指针	

【输出信息】

参数名称	描述
gainmap_lut	标定得到的gain map
gainmap_width	标定得到的gain map的宽
gainmap_height	标定得到的gain map的高

【返回值】

- =0: 参数错误
- =1: 标定成功

3.3.3 Gain Map 验证

【功能描述】

该函数用于验证标定得到的gain map是否满足要求。

【函数原型】

```
bool pdaf_gainmap_vertification(uint16_t *RawAddr, sensor_cfg* psensor_cfg,
                                uint16_t *gainmap_lut)
```

【输入信息】

参数名称	描述	
RawAddr	当前模组在产线拍摄的 raw 图像数据	
psensor_cfg	存储sensor.ini文件内容的结构体指针	
gainmap_lut	由pdaf_gainmap_calibration标定得到的gain map	

【返回值】

- =0: 验证失败，用于验证的gainmap 不满足要求。
- =1: 验证成功

3.3.4 DCC Map 标定

【功能描述】

该函数用于pdaf dcc map的标定。

【函数原型】

```
bool pdaf_dccmap_calibration(uint16_t *RawAddrAll , sensor_cfg* psensor_cfg,
                             uint16_t *gainmap_lut, uint16_t *dccmap_lut,
                             int* dccmap_width,int* dccmap_height,
                             uint8_t* dccSign,wchar_t*pd_offset)
```

【输入信息】

参数名称	描述
RawAddrAll	每个马达位置下拍摄的平均图像数据序列
psensor_cfg	存储sensor.ini文件内容的结构体指针
gainmap_lut	由pdaf_gainmap_calibration标定得到的gain map

【输出信息】

参数名称	描述
dccmap_lut	标定得到的dcc map
dccmap_width	标定得到的dcc map的宽
dccmap_height	标定得到的dcc map的高
dccSign	标定得到的dcc map的方向
pd_offset	标定得到的pd_offset

【返回值】

- =0: 标定失败
- =1: 标定成功

3.3.5 DCC Map 验证

用于验证标定得到的dcc map是否满足要求，包含两个函数接口pdaf_dccmap_vertification () 、pdaf_calc_sharpness () 。

【功能描述】

利用标定得到的dcc map获取模糊图像理想的对焦清晰的位置DAC1；

【函数原型】

```
uint16_t pdaf_dccmap_vertification(uint16_t *RawAddr, sensor_cfg* psensor_cfg,
                                   int cur_pos_id, uint16_t *gainmap_lut,
                                   uint16_t *dccmap_lut, uint8_t dccSign,
                                   wchar_t *pd_offset)
```

【输入信息】

参数名称	描述
RawAddr	拍摄的模糊图像的平均图像数据
psensor_cfg	存储sensor.ini文件内容的结构体指针
cur_pos_id	拍摄的模糊图像当前的马达位置
gainmap_lut	由pdaf_gainmap_calibration标定得到的gain map
dccmap_lut	由pdaf_dccmap_calibration标定得到的dcc map
dccSign	由pdaf_dccmap_calibration标定得到的dccSign
pd_offset	由pdaf_dccmap_calibration标定得到的pd_offset

【返回值】

模糊图像理想的对焦清晰时的马达位置DAC1。

【功能描述】

计算DAC1前后DCC_VERIFY_CAF_DIFF_MAX*abs(DACinf-DACmacro)+DCC_VERIFY_CAF_DIFF_DET内的图像中心区域的清晰度，判断是否出现峰值。

【函数原型】

```
bool pdaf_calc_sharpness(uint16_t *RawAddrAll, int img_num, sensor_cfg*
psensor_cfg)
```

【输入信息】

参数名称	描述
RawAddrAll	DAC1前后valid_toerance*abs(DACinf-DACmacro)+2内的平均图像数据序列
img_num	RawAddrAll包含的图像数量
psensor_cfg	存储sensor.ini文件内容的结构体指针

【返回值】

- =0: 未找到清晰度峰值，验证失败
- =1: 验证成功

4 附录

4.1 OTP 应用示例代码

```
#include <iostream>
#include "RKOTPDLL.h"
#include <stdlib.h>

int main()
{
    int ret = 0;
    #LSC AWB Otp Calibration
    ret =lsc_awbtest();

    #Pdaf Otp Calibration
    uint16_t gainmap[30*30 * 2];
    uint16_t dccmap[20 * 20 * 2];
    uint16_t code = 0;
    int gainmap_width = 0;
    int gainmap_height = 0;
    int dccmap_width = 0;
    int dccmap_height = 0;
    uint8_t dcc_Sign = 0;
    wchar_t pd_offset=0xffff;
    int curpos = 864;

    sensor_cfg sensor_param;
    if (pdaf_initial("D:\\testdll\\testdll\\sensor_OV16A10.ini", &sensor_param))
    {
        if (gainmaptest(&sensor_param, gainmap,&gainmap_width,&gainmap_height))
        {
            if (dccmaptest(&sensor_param, gainmap, dccmap, curpos, &code,
&dccmap_width, &dccmap_height, &dcc_Sign,&pd_offset))
            {
                ret = dcc_calc_sharpness(&sensor_param);
            }
        }
    }
}

bool lsc_awbtest()
{
    int vig=70;
    int height = 1944;
    int width = 2592;
    int bits =10;#10bit
    int bayer =0;#BGGR
    int roiw =5;#1/5 width
    int roih =5;#1/5 height
    int blc[4]={ 64, 64, 64, 64};
    uint16_t lsctable[17*17*4];
    uint16_t awb[3];
    uint16_t Rave =0;
    uint16_t Grave =0;
    uint16_t Gbave =0;
    uint16_t Bave =0;
```

```

int Ydiffer_down =0;##
int Ydiffer_up =5;##
int ColorShading_down =0;##
int ColorShading_up =5;##
float Ydiffer = 0;
float RGconer[25];
float BGconer[25];
bool ret =0;

uint16_t *Rawdata = NULL;
Rawdata = (uint16_t*)malloc(width * height * 2);
memset(Rawdata, 0, sizeof(uint16_t)* width * height);

FILE *fp = NULL;
fp = fopen("input.raw", "rb");
if (fp == NULL)
{
    return false;
}
fread(Rawdata, 1, height * width * 2, fp);
fclose(fp);
ret =lsc_otp_Calibrate(vig,Rawdata, width, height, bits, bayer, b1c[0],
b1c[1],b1c[2], b1c[3],lsctable);
ret = awb_otp_Calibrate(Rawdata, width, height,roiw,roih, bits, bayer,
b1c[0], b1c[1],b1c[2], b1c[3],&Rave,&Grave,&Gbave,&Bave, &awb[0], &awb[1],
&awb[2]);
ret = lsc_otp_verify(Rawdata, width, height, bits, bayer, b1c[0], b1c[1],
b1c[2], b1c[3], lsctable, Ydiffer_down, Ydiffer_up, ColorShading_down,
ColorShading_up, &Ydiffer, RGconer, BGconer);
if(!ret)
{
    return false;    #标定后的数据不满足管控标准，验证失败
}
if(Rawdata!=NULL)
{
    free (Rawdata);
    Rawdata =NULL;
}
return ret;
}

int gainmaptest(sensor_cfg* psensor_cfg, uint16_t *gainmap_lut,int *
gainmap_width, int *gainmap_height)
{
    int ret = 0;
    int height = psensor_cfg->height;
    int width = psensor_cfg->width;
    uint16_t maxval = (1 << psensor_cfg->bits) - 1;
    int gainmapfilenum = 3;
    int over_exp_cnt = 0;

    if (psensor_cfg->pd_dual_mode && psensor_cfg->mipi_mix_mode)
    {
        height = psensor_cfg->height + (psensor_cfg->sensor_type+1)*psensor_cfg-
>pd_height;
    }

    uint16_t *ave_pixelbuf = NULL;

```

```

ave_pixelbuf = (uint16_t*)malloc(width * height * 2);
memset(ave_pixelbuf, 0, sizeof(uint16_t)* width * height);

uint16_t *sum_pixelbuf = NULL;
sum_pixelbuf = (uint16_t*)malloc(width * height * 2);
memset(sum_pixelbuf, 0, sizeof(uint16_t)* width * height);

uint16_t *pixelbuf = NULL;
pixelbuf = (uint16_t*)malloc(width * height * 2);
memset(pixelbuf, 0, sizeof(uint16_t)* width * height);

FILE *fp = NULL;
char prename[20] = "gainRAW";
char endname[5] = ".raw";
for (int i = 0; i < gainmapfilenum; i++)
{
    char temp[20];
    char id[5];
    int idint = i + 1;
    strcpy(temp, prename);
    _itoa_s(idint, id, 10);
    strcat(temp, id);
    strcat(temp, endname);
    fp = fopen(temp, "rb");
    if (fp == NULL)
    {
        return false;
    }
    fread(pixelbuf, 1, height * width * 2, fp);
    fclose(fp);
    for (int j = 0; j < height; j++)
    {
        for (int i = 0; i < width; i++)
        {
            uint32_t tempbuf = *(pixelbuf + j*width + i);
            *(sum_pixelbuf + j*width + i) = tempbuf + *(sum_pixelbuf +
j*width + i);
            if (tempbuf >= maxval)
                over_exp_cnt++;
        }
    }
    for (int j = 0; j < height; j++)
    {
        for (int i = 0; i < width; i++)
        {
            *(ave_pixelbuf + j*width + i) = *(sum_pixelbuf + j*width + i) /
gainmapfilenum;
        }
    }

    ret = pdaf_gainmap_calibration(ave_pixelbuf, psensor_cfg, gainmap_lut,
gainmap_width, gainmap_height);
    ret = pdaf_gainmap_vertification(ave_pixelbuf, psensor_cfg, gainmap_lut);
    if (ret == -1)
        printf("gainmap image OVER exposure!!!");

    if (ave_pixelbuf != NULL)

```

```

    {
        free(ave_pixelbuf);
        ave_pixelbuf = NULL;
    }
    if (pixelbuf != NULL)
    {
        free(pixelbuf);
        pixelbuf = NULL;
    }
    if (sum_pixelbuf != NULL)
    {
        free(sum_pixelbuf);
        sum_pixelbuf = NULL;
    }
    return ret;
}

```

```

bool dccmaptest(sensor_cfg* psensor_cfg, uint16_t *gainmap_lut, uint16_t
*dccmap_lut, int curpos , uint16_t *code, int *dccmap_width, int *dccmap_height,
uint8_t *dcc_Sign, wchar_t *pdaf_offset)
{

```

```

    int ret = 0;
    int lens_begin = 0;
    int lens_end = 64;
    int lens_interval = 8;
    int height = psensor_cfg->height;
    int width = psensor_cfg->width;
    int lensnum = (lens_end - lens_begin) / lens_interval + 1;
    if (psensor_cfg->pd_dual_mode && psensor_cfg->mipi_mix_mode)
    {
        height = psensor_cfg->height + (psensor_cfg->sensor_type +
1)*psensor_cfg->pd_height;
    }
    uint16_t *ave_pixelbuf = NULL;
    ave_pixelbuf = (uint16_t*)malloc(width * height * 2);
    memset(ave_pixelbuf, 0, sizeof(uint16_t)* width * height);

    uint16_t *Rawdata = NULL;
    Rawdata = (uint16_t*)malloc(width * height * 2 * lensnum);
    memset(Rawdata, 0, sizeof(uint16_t)* width * height* lensnum);

    uint16_t *pixelbuf= NULL;
    pixelbuf = (uint16_t*)malloc(width * height * 2);
    memset(pixelbuf, 0, sizeof(uint16_t)* width * height);
    FILE *fp = NULL;
    char prename[20] = "dccRAW";
    char endname[5] = ".raw";
    for (int a = 0;a <lensnum;a++)
    {
        char temp[20];
        char id[5] ;
        int id1 = a + 1;
        strcpy(temp,prename);
        _itoa_s(id1, id, 10);
        strcat(temp, id);
    }

```

```

for (int b = 0; b < 3;b++)
{
    char name[20];
    strcpy(name, temp);
    int id2 = b + 1;
    _itoa_s(id2, id, 10);
    strcat(name, id);
    strcat(name, endname);
    fp = fopen(name, "rb");
    if (fp == NULL)
    {
        return false;
    }
    fread(pixelbuf, 1, height * width * 2, fp);
    fclose(fp);
    for (int j = 0; j < height; j++)
    {
        for (int i = 0; i < width; i++)
        {
            uint16_t tempbuf = *(ave_pixelbuf + j*width + i);
            *(ave_pixelbuf + j*width + i) = tempbuf + *(pixelbuf +
j*width + i) / 3;
        }
    }
    memcpy(Rawdata + a*height * width, ave_pixelbuf, sizeof(uint16_t)*height
* width);
    memset(ave_pixelbuf, 0, sizeof(uint16_t)* width * height);
}

ret = pdaf_dccmap_calibration(Rawdata, psensor_cfg, gainmap_lut, dccmap_lut,
dccmap_width, dccmap_height, dcc_Sign,pd_offset);

memset(pixelbuf, 0, sizeof(uint16_t)* width * height);
memset(ave_pixelbuf, 0, sizeof(uint16_t)* width * height);
char prename1[20] = "image";
for (int i = 0; i < 3; i++)
{
    char temp[20];
    char id[5];
    int idint = i + 1;
    strcpy(temp, prename1);
    _itoa_s(idint, id, 10);
    strcat(temp, id);
    strcat(temp, endname);

    fp = fopen(temp, "rb");
    if (fp == NULL)
    {
        return false;
    }
    fread(pixelbuf, 1, height * width * 2, fp);
    fclose(fp);

    for (int j = 0; j < height; j++)
    {
        for (int i = 0; i < width; i++)
        {

```

```

        uint16_t tempbuf = *(ave_pixelbuf + j*width + i);
        *(ave_pixelbuf + j*width + i) = tempbuf + *(pixelbuf + j*width +
i) / 3;
    }
}
}
*code = pdaf_dccmap_verification(ave_pixelbuf, psensor_cfg, curpos,
gainmap_lut, dccmap_lut, *dcc_Sign, pd_offset);
if (ave_pixelbuf != NULL)
{
    free(ave_pixelbuf);
    ave_pixelbuf = NULL;
}
if (pixelbuf != NULL)
{
    free(pixelbuf);
    pixelbuf = NULL;
}
if (Rawdata != NULL)
{
    free(Rawdata);
    Rawdata = NULL;
}
return ret;
}
bool dcc_calc_sharpness(sensor_cfg* psensor_cfg)
{
    int height = psensor_cfg->height;
    int width = psensor_cfg->width;
    int imgnum = 5;
    if (psensor_cfg->pd_dual_mode && psensor_cfg->mipi_mix_mode)
    {
        height = psensor_cfg->height + (psensor_cfg->sensor_type +
1)*psensor_cfg->pd_height;
    }

    uint16_t *Rawdata = NULL;
    Rawdata = (uint16_t*)malloc(width * height * 2 * imgnum);
    memset(Rawdata, 0, sizeof(uint16_t)* width * height* imgnum);

    uint16_t *pixelbuf = NULL;
    pixelbuf = (uint16_t*)malloc(width * height * 2);
    memset(pixelbuf, 0, sizeof(uint16_t)* width * height);

    FILE *fp = NULL;
    char prename[20] = "calimage";
    char endname[5] = ".raw";
    for (int i = 0; i < imgnum; i++)
    {
        char temp[20];
        char id[5];
        int idint = i + 1;
        strcpy(temp, prename);
        _itoa_s(idint, id, 10);
        strcat(temp, id);
        strcat(temp, endname);

        fp = fopen(temp, "rb");
    }
}

```



```

        if (fp == NULL)
        {
            return false;
        }
        fread(pixelbuf, 1, height * width * 2, fp);
        fclose(fp);
        memcpy(Rawdata + i*height * width, pixelbuf, sizeof(uint16_t)*height *
width);
    }

    bool ret=pdaf_calc_sharpness(Rawdata, imgnum, psensor_cfg);

    if (pixelbuf != NULL)
    {
        free(pixelbuf);
        pixelbuf = NULL;
    }
    if (Rawdata != NULL)
    {
        free(Rawdata);
        Rawdata = NULL;
    }
    return ret;
}

```

4.2 烧录打包数据格式

OTP 烧录格式统一以块的格式排列,如下方表格所示，用户可以根据需求烧录指定的某些块（注意块的ID禁止更改），烧录完所需的块后需要烧录0xff表示结束：

长度(Byte)	数据项	备注
8	Mark	ROCKCHIP
1	ID	ID=0, Sensor Info
4	Size	块数据的大小
	块数据	
1	ID	ID=1, AWB Calibration
4	Size	块数据的大小
	块数据	
1	ID	ID=2, LSC Calibration
4	Size	块数据的大小
	块数据	
1	ID	ID=3, PDAF Calibration
4	Size	块数据的大小
	块数据	
1	ID	ID=4, AF Code
4	Size	块数据的大小
	块数据	
1	ID	ID=5, QSC
4	Size	块数据的大小
	块数据	
...
1	0xFF	结束标志

块的具体内容如下，多字节的数据统一大端模式，高字节储存在低地址，低字节储存在高地址：

偏移地址	长度(Byte)	数据项	备注
0x0000	8	Mark	ROCKCHIP
0x0008	1	ID	ID=0, Sensor Info
0x0009	4	Size	Size=35
0x000D	2	Version	Sensor Info Version: v1.0.8=0x0108
0x000F	1	Supplier ID	用户指定
0x0010	1	Date: Year	如2021年, 写21
0x0011	1	Date: Month	如7月, 写7
0x0012	1	Date: Day	如12日, 写12
0x0013	1	Sensor ID	用户指定
0x0014	1	Lens ID	用户指定
0x0015	1	VCM ID	用户指定
0x0016	1	Driver ID	用户指定
0x0017	4	Module ID	用户指定
0x001B	1	mirror/flip	Bit[7:4]:Mirror Bit[3:0]:Flip ON: 1, OFF: 0
0x001C	1	Full Width H	Full Width High byte
0x001D	1	Full Width L	Full Width Low byte
0x001E	1	Full Height H	Full Height High byte
0x001F	1	Full Height L	Full Height Low byte
0x0020	15	Reserved	Reserved=0xff
0x002F	1	Checksum	Sensor Info Checksum Sum(0x0009~0x002E) % 255+1

偏移地址	长度(Byte)	数据项	备注
0x0000	1	ID	ID=1, AWB Calibration
0x0001	4	Size	Size=43
0x0005	2	Version	AWB Version: v1.0.9=0x0109
0x0007	1	R/G_H	Current R/G value High byte
0x0008	1	R/G_L	Current R/G value Low byte
0x0009	1	B/G_H	Current B/G value High byte
0x000A	1	B/G_L	Current B/G value Low byte
0x000B	1	Gr/Gb_H	Current Gr/Gb value High byte
0x000C	1	Gr/Gb_L	Current Gr/Gb value Low byte
0x000D	1	R/G_H	Golden R/G value High byte
0x000E	1	R/G_L	Golden R/G value Low byte
0x000F	1	B/G_H	Golden B/G value High byte
0x0010	1	B/G_L	Golden B/G value Low byte
0x0011	1	Gr/Gb_H	Golden Gr/Gb value High byte
0x0012	1	Gr/Gb_L	Golden Gr/Gb value Low byte
0x0013	28	Reserved	Reserved=0xff
0x002F	1	Checksum	AWB Calibration Checksum Sum(0x0001~0x002E) % 255+1

偏移地址	长度(Byte)	数据项	备注
0x0000	1	ID	ID=2, LSC Calibration
0x0001	4	Size	Size=2347
0x0005	2	Version	LSC Version: v1.0.b=0x010b
0x0007	2312	LSC Calibration Data	17x17x4matrix fixed to 1024, unpackaged, big endian ,存储的通道顺序为R、Gr、Gb、B
0x090F	32	Reserved	Reserved=0xff
0x092F	1	Checksum	LSC CalibrationChecksum Sum(0x0001~0x092E) % 255+1

偏移地址	长度 (Byte)	数据项	备注
0x0000	1	ID	ID=3, PDAF Calibration
0x0001	4	Size	Size=2603
0x0005	2	Version	PDAF Version: v1.1.12=0x011c
0x0007	1	Gainmap_width	Gainmap size, get from pdaf_gainmap_calibration()
0x0008	1	Gainmap_height	Gainmap size, get from pdaf_gainmap_calibration()
0x0009	2048	Gainmap	Actual size=Gainmap_width* Gainmap_height*2,big endian, 默认值: 0xff
0x0809	1	Checksum	Gainmap Checksum Sum(0x0007~0x00808) % 255+1
0x080A	1	mode value	DCC Map Fit mode (此版本默认为1), 与pd ini配置中 DCC_CALIBRATE_MODE保持一致
0x080B	1	direction	dccSign, get from pdaf_dccmap_calibration() 0: negative 1: positive
0x080C	1	DCCmap_width	DCCmap size, get from pdaf_dccmap_calibration()
0x080D	1	DCCmap_height	DCCmap size, get from pdaf_dccmap_calibration()
0x080E	512	Dccmap	Actual size=DCCmap_width* DCCmap_height*2,big endian, 默认值: 0xff
0x0A0E	1	Checksum	DCCmap Checksum Sum(0x080A~0x0A0D) % 255+1
0x0A0F	2	pd_offset	flag(1)sign(1)reserve(2)integer(4)8fix_decimals(8) ,flag=0:calib sign=1:positive
0x0A11	30	Reserved	Reserved=0xff
0x0A2F	1	Checksum	PDAF Calibration Checksum Sum(0x0001~0x0A2E) % 255+1

偏移地址	长度(Byte)	数据项	备注
0x0000	1	ID	ID=4, AF Code
0x0001	4	Size	Size=27
0x0005	2	Version	AF Version: v1.0.9=0x0109
0x0007	1	AF Infinite H	AF 远焦Code 值高位
0x0008	1	AF Infinite L	AF 远焦Code 值低位
0x0009	1	AF Macro H	AF 近焦Code值高位
0x000A	1	AF Macro L	AF 近焦Code值低位
0x000B	1	AF_Medium H	AF 中焦Code值高位
0x000C	1	AF_Medium L	AF 中焦Code值低位
0x000D	18	Reserved	Reserved=0xff
0x001F	1	Checksum	AF Code Checksum Sum(0x0001~0x001E) % 255+1

偏移地址	长度(Byte)	数据项	备注
0x0000	1	ID	ID=5, QSC
0x0001	4	Size	Size=4107
0x0005	2	Version	QSC Version: v1.0.0=0x0100
0x0007	2	QSC Size	QSC Size, big endian
0x0009	4096	Data	QSC Data, 实际写入大小以QSC Size为准, 默认值: 0xff
0x1009	6	Reserved	Reserved=0xff
0x100F	1	Checksum	QSC Checksum Sum(0x0001~0x100E) % 255+1