Rockchip Camera Module OTP Calibration Guide

ID：RK-SM-YF-607

Release Version：V1.0.28

Release Date：2022-12-08

Security Level：□Top-Secret □Secret □Internal ■Public

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

Rockchip Electronics Co., Ltd.

Address： No.18 Building, A District, No.89, software Boulevard Fuzhou, Fujian,PRC

Website: www.rock-chips.com

Customer service Tel: +86-4007-700-590

Customer service Fax: +86-591-83951833

Customer service e-Mail: fae@rock-chips.com

Preface

Overview

This article aims to introduce the OTP calibration process of camera modules and guide module manufacturers to perform correct OTP calibration work.

Product Version

Chip Name
RV1109/RV1126/RK356X

Reader Object

This document (this guide) is intended for the following engineers:

Production and commissioning engineer related to module factory

ISP Debugging Engineer

Image quality debugging engineer

Revision History

Version number	Author	Date modified	Modification instructions
V1.0.0	Chen Yu	2021-07-12	The first version was released, providing a description of the calibration process and data encapsulation format
v1.0.1	Xu Suwan	2021-07-26	Calibration scheme, sample code
v1.0.2	Xu Suwan	2021-07-27	Modify the function interface and sample code of LSC management
v1.0.3	Xu Suwan	2021-08-03	Delete LSC-controlled functions， add LSC verification interfaces and control metrics and modify the sample code
v1.0.4	Xu Suwan	2021-08-06	Add pdaf gainmap and dccmap calibration and verification functions
v1.0.5	Xu Suwan	2021-08-11	Modify the AWB calibration interface to output the average value of the four channels
v1.0.6	Xu Suwan	2021-08-16	Add full width and full height to the OTP burning data
v1.0.7	Xu Suwan	2021-08-26	Modified the format of the OTP programming data to arrange it in a block format
v1.0.8	Xu Suwan	2021-08-27	Updated PDAF calibration and sample code
v1.0.9	Xu Suwan	2021-09-17	Optimized the pdaf clarity evaluation algorithm and updated the pdaf dll
v1.0.10	Xu Suwan	2021-10-12	The ROCKCHIP logo is added to the OTP programming format
v1.0.11	Xu Suwan	2022-03-29	The OTP MAP AF Code module adds a mid-focus code
v1.0.12	Xu Suwan	2022-04-15	The PDAF calibration section has been updated to include dead pixel detection
v1.0.13	Xu Suwan	2022-05-06	The PDAF DCCMAP calibration algorithm is optimized, and the DCCMODE is changed to 1
v1.0.14	Xu Suwan	2022-05-13	Optimized the PDAF Gainmap verification algorithm
v1.0.15	Xu Suwan	2022-05-16	Update the OTP Map PDAF section
v1.0.16	Xu Suwan	2022-05-18	Optimized the PDAF gainmap and dccmap algorithms, Adjusted the size of each block of gainmap and dccmap

Version number	Author	Date modified	Modification instructions
v1.0.17	Xu Suwan	2022-05-23	Adjust the threshold of PDAF DCCMAP calibration and increase log printing
v1.0.18	Xu Suwan，Huang Ruobing	2022-06-01	Modify the description of the PDAF calibration process and optimize the PDAF Dccmap calibration and verification algorithm
v1.0.19	Xu Suwan，Huang Ruobing	2022-06-01	Adjust the PDAF DCC Sharpness shooting step size and modify the Sharpness verification algorithm
V1.0.20	Xu Suwan	2022-06-08	Updated the PDAF calculation definition algorithm and adjusted the memory size of OTP GainMap and DCCMAP
V1.0.21	Xu Suwan	2022-06-20	Optimized the PDAF calculation clarity algorithm
V1.0.22	Xu Suwan	2022-07-07	Optimized the PDAF calculation clarity algorithm and added DCC_SHARPNESS_COUNT parameters
V1.0.23	Xu Suwan	2022-08-05	The OTP map adds an end flag
v1.0.24	Xu Suwan	2022-09-15	Modify the AWB calibration scheme, adjust the processing process of LSC verification, optimize the PDAF calculation clarity algorithm, and add DCC_SHARPNESS_CHL and SENSOR_TYPE parameters
v1.0.25	Huang Ruobing, Xu Suwan	2022-11-08	The LSC calibration interface opens the calibration force parameter vig, adds sensor.ini related parameter definitions, and adds sensor.ini definitions for dual pd
v1.0.26	Xu Suwan	2022-11-28	Adjust the threshold of brightness control for DCCMap calibration data
v1.0.27	Xu Suwan	2022-12-06	Updated PDAF clarity algorithm
v1.0.28	Xu Suwan	2022-12-08	To determine whether the four vignetting angles of the LSC are symmetrical, the LSC calibration interface adds a threshold card to control the LSC calibration results

目录

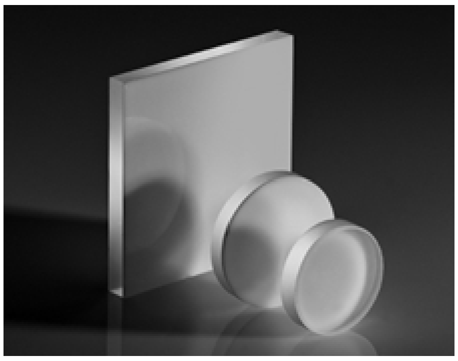
1. 1 LSC&AWB Calibration process

1.1 1.1 Calibration scheme

1. Light source environment and equipment

   D50 (5000K±100K) to avoid stray light interference from other environments.

   Homogenizer or DNP light box that satisfies the above light source, the homogenizer is shown in the following figure:

2. How to shoot

   ```
   1. According to the actual situation, you can choose one of the following two methods
      1. Use a homogenizer: Place the coated side of the homogenizer facing the lens close to the horizontal, keeping the end face of the module, the homogenizer, and the light source panel parallel.
      2. Use DNP light box: Place the module 1-2cm in front of the DNP light source panel, keeping the end face of the module parallel to the light source panel.
    2. Disable functions such as mirror/flip
    3. Select the exposure value so that the maximum brightness (8bit) of the G channel of the image reaches between 160~180
    4. Use that exposure to take a raw image and save it
   ```

3. AWB calibration

   1. Select the center area of the Raw image (20% of the length and width of each channel) and calculate the mean values of R, Gr, Gb, and B channels

      Gr_ave=Gr average of ROI - Black level

      Gb_ave=Gb average of ROI - Black level

      R_ave = Red average of ROI - Black level

      B_ave = Blue average of ROI - Black level

   2. Calculate R/Gb and B/Gb and Gr/Gb

      R/Gb=R_ave/Gb_ave

      B/Gb=B_ave/Gb_ave

    Gr/Gb=Gr_ave/Gb_ave

  3. Fixed-point R/Gb, B/Gb, and Gr/Gb to 10 bitsR/Gb_hex=R/Gb * 1024

     B/Gb_hex=B/Gb * 1024

     Gr/Gb_hex=Gr/Gb * 1024

4. LSC calibration

   1. Set the target value for the calibration calculation to 70%

   2. Calculate the gain of each of the four channels to 10 bits

1.2 1.2 Control schemes

1. Before LSC calibration

   1. Y shading standard :

      ROI = 1/5 Width * 1/5Height

      YShading_Corner = Y_Corner/Y_Center

      30% < YShading_Corner < 55%

      Ydiffer = YShading_Corner_Max - YShading_Corner_Min < 7%

   2. Color Shading standard：

      Color shading uniformity of a single module (control the difference between each block and the central block of a single module)

      ROI frame: 1/5*1/5 Calculate the difference between the 24 blocks and the central block

      ROI = 1/5Width * 1/5Height

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

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

2. LSC calibration

   For the problem of asymmetry of vignetting around the captured RAW data due to module placement problems, the ratio parameter card control calibration result is added lsc_otp_Calibrate interface (default ratio=1.5);

   If the optical center shifts due to the assembly of the module itself, the ratio can be increased to increase the pass rate.

3. After LSC calibration

   1. Y shading standard :

      ROI = 1/5 Width * 1/5Height

      YShading_Corner = Y_Corner/Y_Center

      Ydiffer = YShading_Corner_Max - YShading_Corner_Min < 5%

   2. Color Shading standard：

      Color shading uniformity of a single module (control the difference between each block and the central block of a single module)

      ROI frame: 1/5*1/5 Calculate the difference between the 24 blocks and the central block

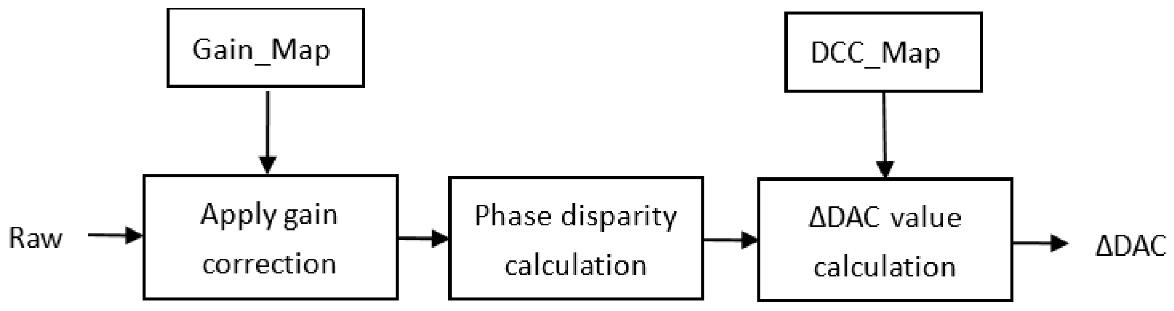
      ROI = 1/5Width * 1/5Height

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

$|(B/G\_Corner)/(B/G\_Center)-1| < 5\%$

2. 2 PDAF Calibration Process

PDAF calibration is divided into two parts: Gain Map calibration and DCC Map calibration, and the calibration process is as follows:



2.1 2.1 Preparation before calibration

1. Module preparation

(1) Lens dirt detection

(2) VCM detection: test linearity, lens position accuracy and stability

(3) AF basic correction:

To determine the relationship between DAC and focus distance----- calibrate the position of the motor of the module lens at the focus Inf and Marco, record the corresponding DAC_inf and Dac_marco values, and determine the DCC_LENS_BEGIN and DCC_LENS_END.

Item	Detail	Note
Test Chart	Vendor Definition	It is recommended that checkerboards, diamond-charts, etc. have high contrast and moderate frequency test charts
Test surface illumination brightness	>400Lux	Imaging without flicker
Distance between the module and the test chart	Inf:2~5m Marco:10-20cm	Inf distance can be simulated with a teleconverter
Module correction direction	When INF/MACRO is calibrated, the optical axis of the module is horizontal In the case of open loop VCM, only the horizontal direction is corrected; If it is a closed loop VCM, the optical axis can be corrected upwards, note that the Inf/Macro correction direction is the same	
Gain Settings	Same as AWB	
Sensor Settings	Disable the mirro/flip/OB settings in the sensor	
Image Brightness Requirements	Same as AWB	
Number of Error Test Samples	Range inspection: each module should be accuracy inspection: greater than 10pcs	If the test fails, the correction method needs to be modified to make the AF far and near focus correction results more accurate

(4) If the module is OIS-enabled, the camera shake correction needs to be completed before calibrating the lsc and pdaf. Move the motor to the reference XY position to verify that the OIS is configured. This step will generate flat field images for LSC and PDAF indexing;

(5) Complete the black level and LSC correction, and if necessary, the field curvature can be corrected

2、Module AE settings

(1) Module AE setting: the digital gain is fixed to 1; Analog gain (traditional PD or 2x1OCL, again set to 1x; Dual-photodiode's PD and again are set to 2x to prevent pixel blooming)

(2) Exposure time: In order to prevent the charge blooming phenomenon of dual-pd and the overexposure of normal pixels (regular_pixel=dual_pd_l+dual_pd_r), adjust the exposure time to make the brightness of the G channel in the center area 700 (raw10) when displaying the full size; For OCL and metal-shield types, a G-channel brightness of 800 (raw10) is guaranteed in the center area. The sensor recommends that the exposure time be set to an integer multiple of 10ms to avoid power frequency interference.

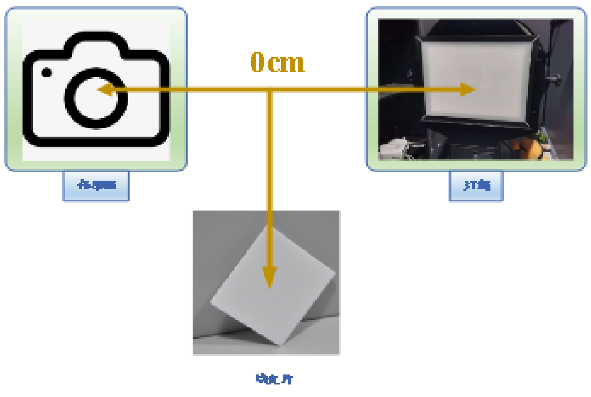
2.2 2.2 Gain Map calibration

1、Light source environment and equipment

D50 (5000K±100K) to avoid stray light interference in other environments；

Homogenizer or DNP light box that meets the above light source, integrating sphere；

2、Shooting method



```
  1. Place the module in front of the uniform surface light source, keep the end
face of the module parallel to the light source surface (eg. the integrating
sphere used in the experiment, the panel brightness value is adjusted to 6.6),
and the homogenizer is placed at the front end of the lens to make the surface
light source fill the picture;
  2. Sensor's gain - for metal-shield and 2x1OCL type sensors, set again and
dgain to 1x, dual-pd type sensors, again set to 2x, dgain set to 1x, and exposure
time set to multiples of 10;
  3. Turn off settings such as mirro/flip/OB in the sensor
  4. The average brightness value of the TOP 10% G channel in the ROI area of
1/32W*1/32H in the center of the screen is 800LSB (raw10) or 200LSB (raw8) when
displayed in full size. The mean value of the dual pd G channel is 700LSB (raw10)
or 175LSB (raw8);
  5. Turn on the camera to warm up for 3 minutes, move the motor to the center
position to shoot 3 RAW images in a row, first check whether the number of dead
pixels exceeds the allowable range, if the number of dead pixels exceeds the
allowable range, you need to reduce the brightness of the picture and shoot
again; If the dead pixel detection passes, the average image data is obtained;
```

3、Calibration

Calling pdaf_gainmap_calibration (...) The function generates the gainmap data and obtains the size of the gainmap, which is gainmap_width*gainmap_height *2 bytes (including left gainmap and right gainmap).

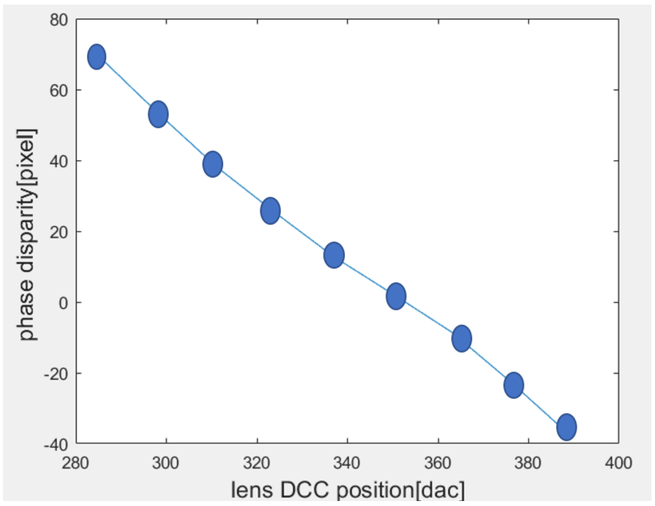
4、Verification

1. Ensure that the environment in step 2 is the same, take a test image, and input the calibrated gainmap and test image into the function pdaf_gainmap_vertification (...), if the return is 1, the verification is successful, and the calibrated gainmap is available; If the return value is -1, it indicates whether the number of dead pixels exceeds the allowable range, and you need to reduce the brightness of the image and reshoot.
2. Validation criteria

   If the difference between the left PD pixel and the right PD pixel corrected by gainmap is less than 5%, the verification is successful.

2.3 2.3 DCC Map calibration

DCC (defocus conversion coefficient) represents the relationship between the motor position and the left and right image differences, divides the image into different regions, moves the motor in equal steps, calculates the pd value at different positions, and uses linear regression to obtain the DCC value of different regions ---DCCMAP

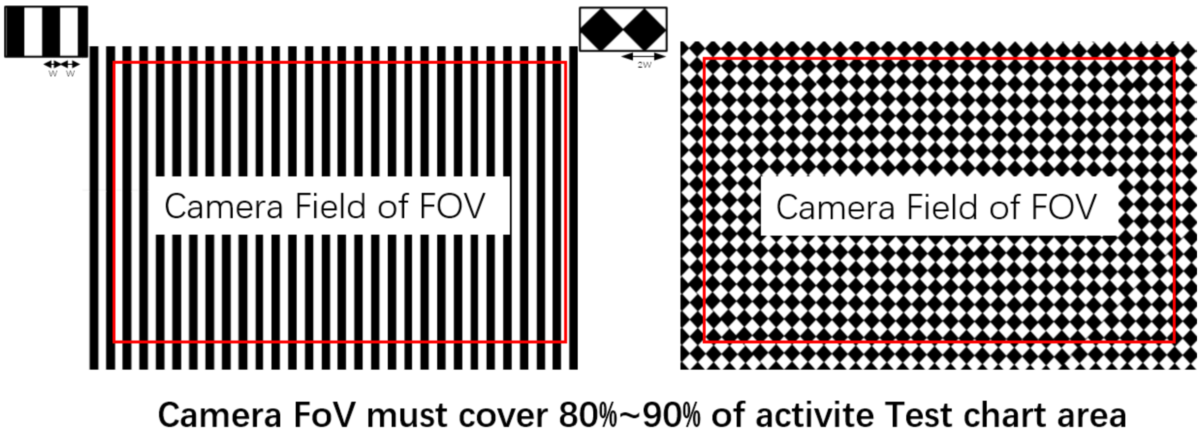


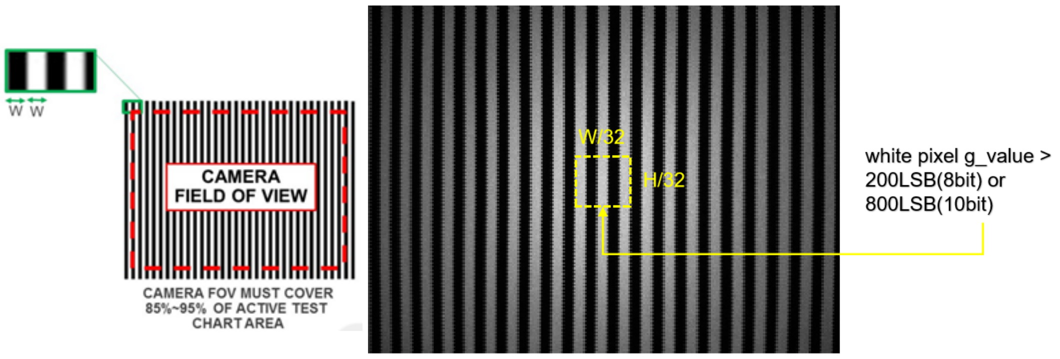
1、Light source environment and equipment

D50 (5000K±100K) to avoid stray light interference from other environments.

Homogeneous sheet or DNP light box that meets the above light source, integrating sphere, black and white standard plate;

Standard plate: The standard plate is made of customized translucent film sheet, and the pattern is evenly spaced black and white stripes or diamond-shaped grids.





The calibration distance is determined in the 2.1.1 AF base correction step

The width of the standard depends on the horizontal FOV of the lens and the calibration distance a, which is recommended

$$S >= a * tan(FOV * pi/360) * 2/0.9$$

The calibration distance is determined in the 2.1.1 AF base correction step

The width of the standard depends on the horizontal FOV of the lens and the calibration distance a, which is recommended

$$\lambda[mm] = \frac{u[um]}{1000} * \frac{raw\_width}{6 * dccmap\_width}$$

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

For IMX258, the coefficient λ is recommended to be 0.11, and for S5KJN1, the coefficient λ is recommended to be 0.088~0.099.

DCC calibration requires correct customization and placement of the standard board, and the inappropriate size of the stripes, rotation or deformation during placement, and over-dimming or overexposure of the brightness will lead to the failure of DCC calibration, as shown in the figure below for some error cases.

图片5

2、How to shoot

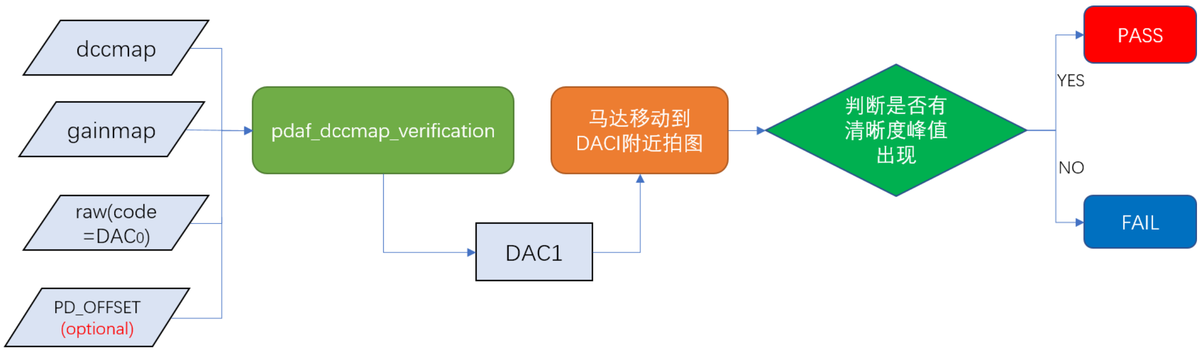
```
  1. Place the striped film in front of the integrating sphere, and place the
module in front of the standard plate about the middle of the focus, and the
distance value is determined by the basic correction in step 2.1.1AF. The end
face is kept parallel to the standard plate to avoid rotation, tilt and twisting
of the standard plate, and to ensure that the standard plate is full of images
when shooting;
  2. sensor gain: for metal-shield and 2x1OCL sensors, set again and dgain to 1x,
dual-pd sensor, again set to 2x, and dgain set to 1x;
  3. Turn off the settings such as mirro/flip/OB in the sensor;
  4. Exposure: Ensure that the average value of the white block g channel in the
ROI area of 1/32w*1/32h in the center of the standard plate screen is 800LSB
(raw10) and 200LSB (raw8), and the average value of the dual pd G channel is
700LSB (raw10) or 175LSB (raw8), so as to avoid insufficient brightness or
overexposure affecting the DCC calibration accuracy; For metal-shield sensors,
ensure that the non-shielded pixels are not overexposed.
  5. Move the motor from DCC_LENS_BEGIN to DCC_LENS_END at DCC_LENS_INTERVAL
sampling intervals, and take three RAW images after the motor stabilizes each
time to obtain the average image data of each position (eg. During the
experiment, the code was moved from 0 to 64 at 8 intervals, and a total of 9
positions were taken, and 27 images were taken
```

3、Demarcate

input the gainmap, motor position parameters and average image data of each position calibrated in section 2.1 into the function pdaf_dccmap_calibration (...) to obtain the dccmap data and obtain the size of the dccmap, the size is dccmap _width*dccmap _height *2 bytes; DCC_RSQ_THRESHOLD If the number of blocks that do not meet the linearity threshold is greater than a certain number, the calibration fails, and the input data needs to be checked for error, and the calibration environment needs to be adjusted to reshoot.

4、Verify

The purpose of this step is to verify that the current calibrated dccmap is correct The verification process is shown in the following figure:



1. The lighting environment of the verification data shooting is the same as that of the DCCmap calibration environment, and the lens surface is parallel to the standard plate to ensure that the image is filled with the standard plate pattern during shooting.
2. Turn on the camera to warm up for 3 minutes, and adjust the motor to a **image blur** position DAC0，

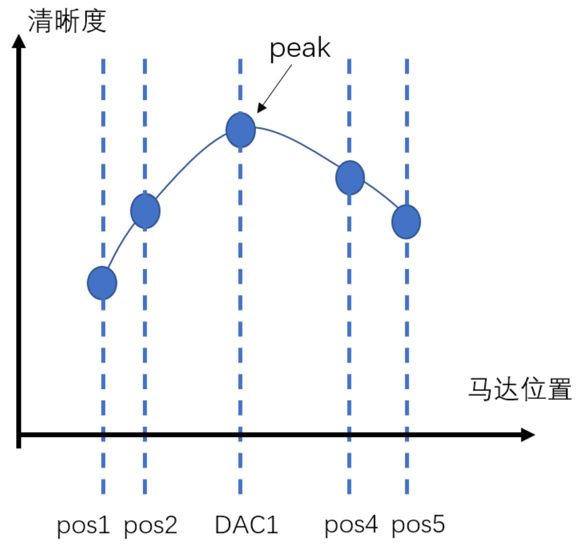
   $$DAC_0 \in (DACinf + 0.2 * abs(DACinf - DACmacro), DACinf + 0.8 * abs(DACinf - DACmacro)]$$

   After the motor is stabilized, three Raw images are captured, the average image data is obtained, and the current position information of the motor DAC0, the average image data, and the calibrated gainmap and dccmap are input into the pdaf_dccmap_vertification() function, and the ideal focusing position DAC1 of the image at the focus blur is calculated；
3. Then move the motor to DAC1, and capture five images in **tolerance_factor***abs(DACinf-DACmacro)+Δ (e.g.Δ=2) before and after DAC1, tolerance_factor the tolerance range during verification, and the error range is different for different types of sensors (e.g.5%, 10%, 20%).

   Move the motor from position 1 to position 5 during verification, and the five positions are (e.g.tolerance_factor=10%, Δ=2)
4.

1	**DAC1-10% *abs(DACinf-DACmacro)-Δ**
2	**DAC1-10% *abs(DACinf-DACmacro)**
3	**DAC1**
4	**DAC1+10% *abs(DACinf-DACmacro)**
5	**DAC1+10% *abs(DACinf-DACmacro)+Δ**

5. Pass five graphs into the function pdaf_calc_sharpness (...) Calculate the sharpness of the image center area DCC_PERCENT_ROI_W* DCC_PERCENT_ROI_H (e.g.30%*30%), and judge whether there is a peak, and if so, the verification is successful; If the verification fails, zoom in on the range of tolerance_factor and Δ (e.g. Δ from 2 to 5), re-acquire 5 images, and repeat step 4. It is best to choose a channel with fewer PD pixels to calculate the definition.


清晰度
peak
马达位置
pos1
pos2
DAC1
pos4
pos5

3. 3 Calibration scheme

Calibrated modules include AWB (Auto White Balance), LSC (Lens Shading Correction), and PDAF (Phase Detection Auto-focus).

3.1 3.1 AWB calibration

【Function Description】

This function is used to calculate the R/G,B/G Gr/Gb of ROI in a raw image.

【Function Prototype】

```
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)
```

【Enter information】

Parameter Name	Description
RawAddr	The raw image data captured by the current module in the production line
width	The width of the raw image is
height	The height of the raw image
bits	The number of bits per pixel. 10bit=10
roiw	The reciprocal of the ratio of the width of the ROI region to the width of the image
roih	The reciprocal of the ratio of the height of the ROI area to the height of the image
bayer	Bayer mode for raw images. BGGR=0， GBRG=1，GRBG=2， RGGB=3
blc_r	R channel BLC
blc_gr	Gr channel BLC
blc_gb	Gb channel BLC
blc_b	B-channel BLC

【Output Information】

Parameter Name	Description
R	ROI region R channel mean
Gr	ROI region Gr channel mean
Gb	ROI Region Gb Channel Mean
B	ROI region B-channel mean
R_G	The ratio of the mean value of the R channel to the mean value of the Gb channel in the ROI region *1024
B_G	The ratio of the mean value of the B channel to the mean value of the Gb channel in the ROI region *1024
Gr_Gb	The ratio of the mean value of the Gr channel to the mean value of the Gb channel in the ROI region *1024

【Return value】

- =0：The parameter is incorrect
- =1：The calibration was successful

3.2 3.2 LSC calibration

3.2.1 3.2.1 LSC calibration

【Feature description】

This function is used to calculate the gain of each channel of the raw image.

【Function prototypes】

```
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)
```

【Enter your information】

Parameter Name	Description
vig	LSC correction force, default vig=70 (%)
RawAddr	Raw image data captured by the current module on the production line
width	The width of the raw image
height	The height of the raw image
bits	The number of bits per pixel. 10bit=10
bayer	Bayer mode for raw images. BGGR=0， GBRG=1，GRBG=2， RGGB=3
blc_r	R channel BLC
blc_gr	Gr channel BLC
blc_gb	Gb channel BLC
blc_b	B channel BLC
ratio	**Threshold to measure whether the surrounding vignetting angle is symmetrical, recommended value ratio=1.5**

【Output Information】

Parameter Name	Description
lsc_otp	LSC Gain： ● R channel gain（17 * 17） ● Gr channel gain（17 * 17） ● Gb channel gain（17 * 17） ● B channel gain（17 * 17）

【Return Value】

- =0：The LSC calibration data is abnormal and the vignetting is abnormal
- =1：calibration succeeded

3.2.2 3.2.2 LSC Validation

【Feature description】

This function is used to verify the results of the LSC calibration.

【Function prototypes】

```
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 *fRGCorner,float *fBGCorner)
```

【Enter your information】

The name of the parameter	Description
RawAddr	Raw image data captured by the current module on the production line
width	The width of the raw image
height	The height of the raw image
bits	The number of bits per pixel. 10bit=10
bayer	Bayer mode for raw images. BGGR=0， GBRG=1，GRBG=2， RGGB=3
blc_r	R channel BLC
blc_gr	Gr channel BLC
blc_gb	Gb channel BLC
blc_b	B channel BLC
lsc_otp	The lsc_otp output is calculated by the lsc_otp_Calibrate
Ydiffer_down	Calibrated control indicator YShading_Corner_Max - the lower limit of the YShading_Corner_Min, in percentage form
Ydiffer_up	Calibrated control indicator YShading_Corner_Max - The upper limit of the YShading_Corner_Min, in percentage form
ColorShading_down	The lower limit of calibrated control indicator |(R/G_Corner)/(R/G_Center)-1|、|(B/G_Corner)/(B/G_Center)-1|, in percentage form
ColorShading_up	The upper limit of calibrated control indicator |(R/G_Corner)/(R/G_Center)-1|、|(B/G_Corner)/(B/G_Center)-1|, in percentage form

【Output information】

The name of the parameter	**Description**
fYdiffer	The value of YShading_Corner_Max - YShading_Corner_Min
fRGCorner	The value of each block |(R/G_Corner)/(R/G_Center)-1|, a total of 5 x 5 blocks
fBGCorner	The value of each block|(B/G_Corner)/(B/G_Center)-1|, a total of 5 x 5 blocks

【Return value】

- =0：If the calibrated control indicators are not met, the verification fails
- =1：Verification successful

3.3 3.3 PDAF calibration

3.3.1 3.3.1 Get sensor config

【Feature description】

Configure the sensor.ini file and get information about the PD sensor from the sensor.ini file.

【Configure the content】

[sensor config]

Keywords	Description
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 1:2x2OCL 2:dual PD
PD_DUAL_MODE	**0:non-dual** 1:dual PD
PD_BINNING_TYPE	calibration using binning PD 0:ON 1:OFF
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]

Keywords	Description
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(5%*1024)
CROSS_VER	using other imgs verify gainmap

[DCCmap_Calib]

Keywords	Description
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_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]

Keywords	Description
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 critieria

【Example】

Take the IMX258 pd pixel distribution as an example, as shown in the following figure:

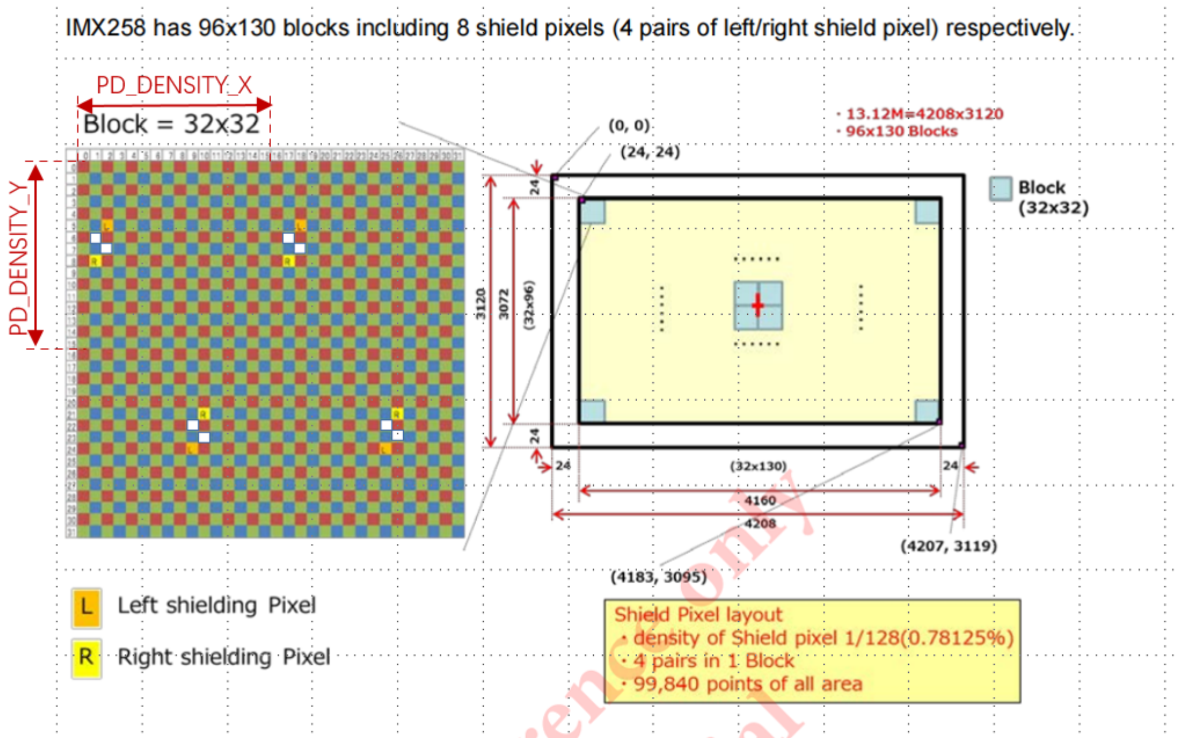


Figure 4-16 Shield Pixel Block Arrangement

According to the information on the figure, the contents of the sensor.ini file are as follows:

```
[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
```

【Function prototypes】

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

【Iutput information】

Parameter name	Description
filename	The full path of the sensor.ini file

【Output information】

Parameter name	Description
psensor_cfg	A struct pointer that stores the contents of the sensor.ini file

【Return value】

- =0：Failed to get sensor.ini
- =1：Get sensor.ini successfully

3.3.2 3.3.2 Gain Map calibration

【Feature description】

This function is used for the indexing of PDAF gain maps.

【Function prototypes】

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

【Iutput information】

Parameter name	Description
RawAddr	Raw image data captured by the current module on the production line
psensor_cfg	A struct pointer that stores the contents of the sensor.ini file

【Output information】

Palamet then	Description
gainmap_lut	Calibrated gain map
gainmap_width	The width of the indexed gain map
gainmap_height	The high of the gain map obtained by calibration

【Return value】

- =0：The parameter is incorrect
- =1：Calibration successful

3.3.3 3.3.3 Gain Map Validation

【Feature description】

This function is used to verify whether the calibrated gain map meets the requirements.

【Function prototypes】

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

【Iutput information】

Parameter name	Description	
RawAddr	Raw image data captured by the current module on the production line	
psensor_cfg	A struct pointer that stores the contents of the sensor.ini file	
gainmap_lut	The gain map obtained by the pdaf_gainmap_calibration index	

【Return value】

- =-1：The number of dead pixels in the captured RAW image data exceeds the allowable range.
- =0：The validation failed, and the gainmap used for validation did not meet the requirements.
- =1：Verification successful

3.3.4 3.3.4 DCC Map calibration

【Feature description】

This function is used for the calibration of PDAF DCC MAP.

【Function prototypes】

```
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)
```

【Iutput information】

Parameter name	Description
RawAddrAll	The average sequence of image data captured at each motor position
psensor_cfg	A struct pointer that stores the contents of the sensor.ini file
gainmap_lut	The gain map obtained by the pdaf_gainmap_calibration index

【Output information】

Parameter name	Description
dccmap_lut	The resulting DCC MAP
dccmap_width	The width of the indexed DCC map
dccmap_height	The height of the indexed DCC MAP
dccSign	Calibrate the orientation of the resulting DCC MAP

【Return value】

- =0：Calibration failed
- =1：Calibration successful

3.3.5 3.3.5 DCC Map Verification

Used to verify whether the calibrated dcc map meets the requirements, including two function interfaces pdaf_dccmap_vertification() and pdaf_calc_sharpness().

【Function description】

Use the calibrated dcc map to obtain the ideal focus clear position DAC1 of the blurred image;

【function prototype】

```
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)
```

【Iutput information】

Parameter name	Description
RawAddr	Average image data of captured blurry images
psensor_cfg	Structure pointer that stores the contents of the sensor.ini file
cur_pos_id	Blurred image taken of current motor position
gainmap_lut	gain map calibrated by pdaf_gainmap_calibration
dccmap_lut	dcc map calibrated by pdaf_dccmap_calibration
dccSign	dccSign calibrated by pdaf_dccmap_calibration

【Return value】

None

Motor position DAC1 when blurred image is ideally in focus.

【Function description】

Calculate the sharpness of the image center area within 5%*(DACinf-DACmacro)+1 before and after DAC1 to determine whether a peak occurs.

【function prototype】

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

【Iutput information】

Parameter name	Description
RawAddrAll	Average image data sequence within 5% (DACinf-DACmacro) + 1 before and after DAC1
img_num	Number of images contained in RawAddrAll
psensor_cfg	Structure pointer that stores the contents of the sensor.ini file

【Return value】

- =0：No sharpness peak found, verification failed
- =1：Verification successful

4. 4 Appendix

4.1 4.1 OTP application sample code

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

int main()
{
    int ret = 0;
    ret =lsc_awbtest();

    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;
    int curpos = 864;

    sensor_cfg sensor_param;
    if (pdaf_initial("D:\\testdll\\testdll\\sensor_ov50c.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))
            {
                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]={16,16,16,16};
    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);
    float ratio=1.5;
    ret =lsc_otp_Calibrate(vig,Rawdata, width, height, bits, bayer, blc[0],
blc[1],blc[2], blc[3], ratio,lsctable);
    ret = awb_otp_Calibrate(Rawdata, width, height,roiw,roih, bits, bayer,
blc[0], blc[1],blc[2], blc[3],&Rave,&Grave,&Gbave,&Bave, &awb[0], &awb[1],
&awb[2]);
    ret = lsc_otp_verify(Rawdata, width, height, bits, bayer, blc[0], blc[1],
blc[2], blc[3], lsctable, Ydiffer_down, Ydiffer_up, ColorShading_down,
ColorShading_up, &Ydiffer, RGconer, BGconer);
    if(!ret)
    {
        return false;   #The calibrated data does not meet the control standards
and the verification fails.
    }
    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;
    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)
{

    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;

    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);

    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_vertification(ave_pixelbuf, psensor_cfg, curpos,
gainmap_lut, dccmap_lut, *dcc_Sign);
    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;
    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 4.2 Burning packaged data format

The OTP burning format is uniformly arranged in block format, as shown in the table below:

Length(Byte)	data item	Remark
8	Mark	ROCKCHIP
1	ID	ID=0，Sensor Info
4	Size	Block data size
	block data	
1	ID	ID=1，AWB Calibration
4	Size	Block data size
	block data	
1	ID	ID=2，LSC Calibration
4	Size	Block data size
	block data	
1	ID	ID=3，PDAF Calibration
4	Size	Block data size
	block data	
1	ID	ID=4，AF Code
4	Size	Block data size
	block data	
...	...	...
1	0xFF	end sign

The specific content of the block is as follows. Multi-byte data is unified in big-endian mode. The high byte is stored in the low address and the low byte is stored in the high address:

Offset address	Length(Byte)	data item	Remark
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	User specified
0x0010	1	Date：Year	For example, in 2021, write 21
0x0011	1	Date：Month	For example, in July, write 7
0x0012	1	Date：Day	For example, on the 12th, write 12
0x0013	1	Sensor ID	User specified
0x0014	1	Lens ID	User specified
0x0015	1	VCM ID	User specified
0x0016	1	Driver ID	User specified
0x0017	4	Module ID	User specified
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
0x002F	1	Checksum	Sensor Info Checksum Sum(0x0009~0x002E) % 255+1

Offset address	Length(Byte)	Data item	Remark
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
0x002F	1	Checksum	AWB Calibration Checksum Sum(0x0001~0x002E) % 255+1

Offset address	Length(Byte)	Data item	Remark
0x0000	1	ID	ID=2，LSC Calibration
0x0001	4	Size	Size=2347
0x0005	2	Version	LSC Version： **v1.0.a=0x010a**
0x0007	2312	LSC Calibation Data	17x17x4matrix fixed to 1024, unpackaged, big endian, the stored channel order is R, Gr, Gb, B
0x090F	32	Reserved	Reserved
0x092F	1	Checksum	LSC CalibrationChecksum Sum(0x0001~0x092E) % 255+1

Offset address	**Length(Byte)**	**Data item**	**Remark**
0x0000	1	ID	ID=3，PDAF Calibration
0x0001	4	Size	Size=2603
0x0005	2	Version	PDAF Version： **v1.1.7=0x0117**
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
0x0809	1	Checksum	Gainmap Checksum Sum(0x0007~0x00808) % 255+1
0x080A	1	mode value	DCC Map Fit mode (**this version defaults to 1**), consistent with DCC_CALIBRATE_MODE in pd ini configuration
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
0x0A0E	1	Checksum	DCCmap Checksum Sum(0x080A~0x0A0D) % 255+1
0x0A0F	32	Reserved	Reserved
0x0A2F	1	Checksum	PDAF Calibration Checksum Sum(0x0001~0x0A2E) % 255+1

Offset address	Length(Byte)	Data item	Remark
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 far focus Code value high bit
0x0008	1	AF Infinite L	AF far focus Code value low
0x0009	1	AF Macro H	AF close focus Code value high
0x000A	1	AF Macro L	AF close focus Code value low
0x000B	1	AF_Medium H	AF medium focus Code value high
0x000C	1	AF_Medium L	AF medium focus Code value low
0x000D	18	Reserved	Reserved
0x001F	1	Checksum	AF Code Checksum Sum(0x0001~0x001E) % 255+1