UNISOC Camera AWB&OC&LSC Software Guide

Release Date	2019.07.02
Document No.	
Version	1.1
Document Type	User Guide
Platform	SC9832E/SC9863A/UMS312/UIS8581E/UDS710+UDX710/UIS7862/SL8541E
OS Version	Android8.1/Android9.0/Android10

声明 Statement

本文件所含数据和信息都属于紫光展锐所有的机密信息，紫光展锐保留所有相关权利。本文件仅为信息参考之目的提供，不包含任何明示或默示的知识产权许可，也不表示有任何明示或默示的保证，包括但不限于满足任何特殊目的、不侵权或性能。当您接受这份文件时，即表示您同意本文件中内容和信息属于紫光展锐机密信息，且同意在未获得紫光展锐书面同意前，不使用或复制本文件的整体或部分，也不向任何其他方披露本文件内容。紫光展锐有权在未经事先通知的情况下，在任何时候对本文件做任何修改。紫光展锐对本文件所含数据和信息不做任何保证，在任何情况下，紫光展锐均不负责任何与本文件相关的直接或间接的、任何伤害或损失。

All data and information contained in or disclosed by this document is confidential and proprietary information of UNISOC and all rights therein are expressly reserved. This document is provided for reference purpose, no license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document, and no express and implied warranties, including butwithout limitation, the implied warranties of fitness for any particular purpose, and non-infringement, as well as any performance. By accepting this material, the recipient agrees that the material and the information contained therein is to be held in confidence and in trust and will not be used, copied, reproduced in whole or in part, nor its contents revealed in any manner to others without the express written permission of UNISOC. UNISOC may make any changes at any time without prior notice. Although every reasonable effort is made to present current and accurate information, UNISOC makes no guarantees of any kind with respect to the matters addressed in this document. In no event shall UNISOC be responsible or liable, directly or indirectly, for any damage or loss caused or alleged to be caused by or in connection with the use of or reliance on any such content.

关键字 Keywords

AWB：Auto White Balance

AE：Auto Exposure

LSC：Lens Shading Correction

版本历史 Revision history

版本 Version	日期 Date	作者 Author	描述 Description
1.0.0	2017-04-17	Unisoc	草案
1.0.6	2017-10-16	Unisoc	补充 OTP 烧录标准要求
1.0.7	2017-12-15	Unisoc	1.图像 ROI 区域中 G 分量平均值(Gr 和 Gb 的平均值）由 V1.0.6 的 800±50 改为 V1.0.7 的 720±40 2.增加对 Shading 的光源要求，请避免出现 flicker 3.AE 计算时 改为计算 ROI 的 G 值，相关补偿对应的 Gain 为 720±10。并增加了计算公式 4.AF 计算要求更改：测试结果不能低于解析力标准的 90%。
1.0.8	2018-03-09	Unisoc	1. 增加 Dual PD 标定 2. 更新 AF 验证说明
1.1	2019-07-02	Unisoc	1. 更新版本名称 2. 更新文档名称 3. 更新文档模板

前　言 Foreword

一 范围 Scope

此文档适用于需要采用紫光展锐摄像头模组烧录标准，指导其人员对模组内容进行正确烧录。

二 内容定义 Details Definitions

1. 定义 Definitions

   NA

2. 缩略语 Abbreviations

   NA

三 参考文献　References

暂无。

目　　录 Contents

1. Camera Module Calibration

Camera Module Calibration 包括 AWB(auto white balance) 和 OC(optical center)、LSC(lens shading correction)的 calibration

(1) 拍摄当前模组在工厂生产线的 16bits raw 数据，拍图请参考《Unisoc Camera AWB&LSC&AF&PDAF 烧录规范 V1.1》

(2) 基于以上 raw 数据和对应 setting，产生 AWB/OC/LSC 等数据；

(3) 将 AWB/OC/LSC 等数据按照 OTP MAP 的设定写入到对应 OTP binary(例如 8K bytes)中；

1.1 AWB calibration

The function is used to calculate the R/G/B average value of ROI in a raw image(16bits raw, only 10-bits valid).

```
int32_t cal_otp_awb(uint16_t* raw_image_random_factory,
                    uint32_t raw_width, uint32_t raw_height, uint32_t bayer_pattern,
                    uint16_t blc_r, uint16_t blc_gr, uint16_t blc_gb, uint16_t blc_b,
                    uint32_t awb_roi_x, uint32_t awb_roi_y, uint32_t awb_roi_w, uint32_t awb_roi_h,
                    uint16_t* awb_r, uint16_t* awb_g, uint16_t* awb_b)
```


Input information

raw_image_random_factory	当前模组在产线拍摄的 raw 数据
raw_width	width of raw image
raw_height	height of raw image
bayer_pattern	raw image bayer pattern. /gr=0;r=1;b=2;gb=3/
blc_r	blc of r channel
blc_gr	blc of gr channel
blc_gb	blc of gb channel
blc_b	blc of b channel
awb_roi_x	X-coordinate of AWB ROI top- left
awb_roi_y	Y-coordinate of AWB ROI top- left
awb_roi_w	width of AWB ROI
awb_roi_h	height of AWB ROI

Output information

awb_r	Mean value of AWB R channel
awb_g	Mean value of AWB G channel
awb_b	Mean value of AWB B channel

Return information
<0: parameter error
==0: OK

1.2 OC Calibration

The Optical Center should be calculated in each color channel (4 channels: r/gr/gb/b).

```
int32_t cal_otp_oc(uint16_t* raw_image_random_factory,
                   uint32_t raw_width, uint32_t raw_height, uint32_t bayer_pattern,
                                       uint16_t blc_r, uint16_t blc_gr, uint16_t blc_gb, uint16_t
        blc_b,
                                       uint16_t* oc_x_r, uint16_t* oc_y_r,
                   uint16_t* oc_x_gr, uint16_t* oc_y_gr,
                   uint16_t* oc_x_gb, uint16_t* oc_y_gb,
                   uint16_t* oc_x_b, uint16_t* oc_y_b)
```


Input information

raw_image_random_factory	当前模组在产线拍摄的 raw 数据
raw_width	width of raw image
raw_height	height of raw image
bayer_pattern	raw image bayer pattern. /gr=0;r=1;b=2;gb=3/
blc_r	blc of r channel
blc_gr	blc of gr channel
blc_gb	blc of gb channel
blc_b	blc of b channel

Output information

oc_x_r	R channel optical center position of X-Coordinate
oc_y_r	R channel optical center position of Y-Coordinate
oc_x_gr	Gr channel optical center position of X-Coordinate
oc_y_gr	Gr channel optical center position of Y-Coordinate
oc_x_gb	Gb channel optical center position of X-Coordinate

oc_y_gb	Gb channel optical center position of Y-Coordinate
oc_x_b	B channel optical center position of X-Coordinate
oc_y_b	B channel optical center position of Y-Coordinate

Return information
<0: parameter error
==0: OK

1.3 LSC Calibration

Calculate the gain of each channel.

```
int32_t cal_otp_lsc(uint16_t* raw_image_random_factory,
                    uint32_t raw_width, uint32_t raw_height, uint32_t bayer_pattern,
                                        uint16_t blc_r, uint16_t blc_gr, uint16_t blc_gb,
    uint16_t blc_b,
                                        uint32_t lsc_grid, uint16_t* lsc_otp, uint32_t* lsc_size,
    uint32_t is_lsc_compression)
```

Input information

raw_image_random_factory	当前模组在产线拍摄的 raw 数据
raw_width	width of raw image
raw_height	height of raw image
bayer_pattern	raw image bayer pattern. /gr=0;r=1;b=2;gb=3/
blc_r	blc of r channel
blc_gr	blc of gr channel
blc_gb	blc of gb channel
blc_b	blc of b channel
lsc_grid	grid size, grid=16,32,64,96,128
is_lsc_compression	LSC table compression, 0 is recommmed, this feature is not ready

Output information

lsc_otp	lsc gain buffer: r gain(lsc_size/4) gr gain(lsc_size/4) gb gain(lsc_size/4) b gain(lsc_size/4)
lsc_size	lsc gain buffer size, lsc gain channel size is lsc_size/4

Return information
<0: parameter error
==0: OK

1.4 AWB & LSC Verification

Verify the AWB&LSC Calibration result.

```
int32_t verify_otp_lsc_awb(uint16_t* raw_image, uint32_t raw_width, uint32_t raw_height,
                uint32_t bayer_pattern, uint16_t* raw_image_output,
                uint16_t blc_r, uint16_t blc_gr, uint16_t blc_gb, uint16_t blc_b,
                uint32_t lsc_grid, uint16_t* lsc_otp, uint32_t lsc_size,
               uint32_t is_lsc_compression,
               uint16_t awb_r, uint16_t awb_g, uint16_t awb_b);
```

Input information

raw_image	当前模组在产线拍摄的 raw 数据
raw_width	width of raw image
raw_height	height of raw image
bayer_pattern	raw image bayer pattern. /gr=0;r=1;b=2;gb=3/
blc_r	blc of r channel
blc_gr	blc of gr channel
blc_gb	blc of gb channel
blc_b	blc of b channel
lsc_grid	grid size, grid=16,32,64,96,128
lsc_otp	Lsc calibration buffer
Lsc_size	Lsc calibration data size in bytes
is_lsc_compression	LSC table compression, now only support 0

Output information

raw_image_output	Verify 后输出的 raw 数据

Return information
==0: OK
!=0: failed

2. OTP Apply Sample Code

请参考 otp_cal_dll_test.cpp

请基于这个 apply LSC&AWB 之后输出的 raw 图上进行 verify

```
int main(int argc, char* argv[])
{
    // input: raw, now only support 16bits raw
    const char* raw_filename = "input.raw";

    /******************** please check below parameters *********************/
#if 1
    uint32_t raw_width = 3264;
    uint32_t raw_height = 2448;
    uint32_t bayer_pattern = 2;    /* 0 - Gr, 1 - R, 2 - B, 3 - Gb */

    uint16_t blc_r = 16; /* OB value */
    uint16_t blc_gr = 16; /* OB value */
    uint16_t blc_gb = 16; /* OB value */
    uint16_t blc_b = 16; /* OB value */

    // input: ROI of AWB
    uint32_t awb_roi_w = raw_width/8 / 2 * 2; // raw_width/8, center 1/8 is recommended, must be even number
    uint32_t awb_roi_h = raw_height/8 / 2 * 2; // raw_height/8, center 1/8 is recommended, must be even number
    uint32 awb_roi_x = (raw_width-awb_roi_w)/2 / 2 * 2; // must be even number
    uint32 awb_roi_y = (raw_height-awb_roi_h)/2 / 2 * 2; // must be even number

    // input: LSC grid
    uint32_t lsc_grid = 96;

    // input: otp
    PTR_FUNC_SAVE_OTP_BIN save_otp_bin = save_otp_8KB_8m;
    char* filename_otp = "otp_8KB_8m.bin";
#endif
    /****************** please check above parameters ******************************/


    const char* raw_filename_output = "output.raw";
```

```
// output
// awb output, AWB@OTP
uint16_t awb_r = 0;
uint16_t awb_g = 0;
uint16_t awb_b = 0;

// optical center output, OC@OTP
uint16_t oc_x_r = (uint16_t)raw_width/2;
uint16_t oc_y_r = (uint16_t)raw_height/2;
uint16_t oc_x_gr = (uint16_t)raw_width/2;
uint16_t oc_y_gr = (uint16_t)raw_height/2;
uint16_t oc_x_gb = (uint16_t)raw_width/2;
uint16_t oc_y_gb = (uint16_t)raw_height/2;
uint16_t oc_x_b = (uint16_t)raw_width/2;
uint16_t oc_y_b = (uint16_t)raw_height/2;

// lsc shading table, lsc@OTP
uint16_t lsc_gain[4096] = {0};
uint32_t lsc_size = 0;


HINSTANCE hDll = NULL;
FNUC_CAL_OTP_AWB cal_otp_awb = NULL;
FUNC_CAL_OTP_OC cal_otp_oc = NULL;
FUNC_CAL_OTP_LSC cal_otp_lsc = NULL;
FUNC_VERIFY_OTP_LSC_AWB verify_otp_lsc_awb = NULL;
unsigned short* raw_image_input = NULL;
unsigned short* raw_image_output = NULL;
int ret = 0;



/*get dll handle*/
hDll = LoadLibrary("OTPshTool.dll");
if (hDll == NULL)
{
    printf("Failed to load OTPshTool.dll\n");
    goto EXIT;
}

/*get function pointer*/
```

```
    cal_otp_awb = (FNUC_CAL_OTP_AWB)GetProcAddress(hDll, "cal_otp_awb");
    if (cal_otp_awb == NULL)
    {
        printf("Failed to GetProcAddress cal_otp_awb\n");
        goto EXIT;
    }

    cal_otp_oc = (FUNC_CAL_OTP_OC)GetProcAddress(hDll, "cal_otp_oc");
    if (cal_otp_oc == NULL)
    {
        printf("Failed to GetProcAddress cal_otp_oc\n");
        goto EXIT;
    }

    cal_otp_lsc = (FUNC_CAL_OTP_LSC)GetProcAddress(hDll, "cal_otp_lsc");
    if (cal_otp_lsc == NULL)
    {
        printf("Failed to GetProcAddress cal_otp_lsc\n");
        goto EXIT;
    }

    verify_otp_lsc_awb = (FUNC_VERIFY_OTP_LSC_AWB)GetProcAddress(hDll,
"verify_otp_lsc_awb");

    // read raw image
    raw_image_input = (unsigned short*)malloc(raw_width * raw_height * sizeof(uint16_t));
    if (raw_image_input == NULL)
    {
        printf("Failed to malloc memory\n");
        goto EXIT;
    }
    if (read_unpacked_raw(raw_image_input, raw_width, raw_height, raw_filename) != 0)
    {
        printf("Failed to read file %s\n", raw_filename);
        goto EXIT;
    }

    raw_image_output = (unsigned short*)malloc(raw_width * raw_height * sizeof(uint16_t));


    ret = cal_otp_awb(raw_image_input, raw_width, raw_height, bayer_pattern, blc_r, blc_gr, blc_gb,
blc_b, awb_roi_x, awb_roi_y, awb_roi_w, awb_roi_h, &awb_r, &awb_g, &awb_b);
```

```
    if (ret < 0)
    {
        printf("error parameter\n");
        goto EXIT;
    }
    printf("AWB@OTP: (%d, %d, %d) or (0x%x, 0x%x, 0x%x)\n", awb_r, awb_g, awb_b, awb_r, awb_g, awb_b);



    ret = cal_otp_oc(raw_image_input, raw_width, raw_height, bayer_pattern, blc_r, blc_gr, blc_gb, blc_b, &oc_x_r, &oc_y_r, &oc_x_gr, &oc_y_gr, &oc_x_gb, &oc_y_gb, &oc_x_b, &oc_y_b);
    if (ret < 0)
    {
        printf("error parameter\n");
        goto EXIT;
    }
    printf("OC@OTP: r(%d,%d) gr(%d,%d) gb(%d,%d) b(%d,%d) or r(0x%x,0x%x) gr(0x%x,0x%x) gb(0x%x,0x%x) b(0x%x,0x%x)\n", oc_x_r, oc_y_r, oc_x_gr, oc_y_gr, oc_x_gb, oc_y_gb, oc_x_b, oc_y_b, oc_x_r, oc_y_r, oc_x_gr, oc_y_gr, oc_x_gb, oc_y_gb, oc_x_b, oc_y_b);



    ret = cal_otp_lsc(raw_image_input, raw_width, raw_height, bayer_pattern, blc_r, blc_gr, blc_gb, blc_b, lsc_grid, lsc_gain, &lsc_size, is_lsc_compression);
    if (ret < 0)
    {
        printf("error parameter\n");
        goto EXIT;
    }
    printf("LSC@OTP size = %d Bytes\n", lsc_size);



    // verify
    if ((verify_otp_lsc_awb != NULL) && (raw_image_output != NULL))
    {
        verify_otp_lsc_awb(raw_image_input, raw_width, raw_height, bayer_pattern, raw_image_output, blc_r, blc_gr, blc_gb, blc_b, lsc_grid, lsc_gain, lsc_size, is_lsc_compression, awb_r, awb_g, awb_b);

        write_unpacked_raw(raw_image_output, raw_width, raw_height, raw_filename_output);
    }
```

```
    // save awb/oc/lsc@OTP to otp.bin
    save_otp_bin(awb_r, awb_g, awb_b, oc_x_r, oc_y_r, oc_x_gr, oc_y_gr, oc_x_gb, oc_y_gb, oc_x_b, oc_y_b, (uint16_t)raw_width, (uint16_t)raw_height, (uint8_t)lsc_grid, lsc_gain, lsc_size, filename_otp);



EXIT:
    if (raw_image_output != NULL)
    {
        free(raw_image_output);
    }

    if (raw_image_input != NULL)
    {
        free(raw_image_input);
    }

    if (hDll != NULL)
    {
        FreeLibrary(hDll);
    }


    return 0;
}
```