



Unisoc Confidential For hiar

摄像头模组 PDAF 标定工具使用说明

文档版本
发布日期

V1.6
2021-03-03

版权所有 © 紫光展锐（上海）科技有限公司。保留一切权利。

本文件所含数据和信息都属于紫光展锐（上海）科技有限公司（以下简称紫光展锐）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，紫光展锐均不负任何与本文件相关的直接或间接的、任何伤害或损失。

请参照交付物中说明文档对紫光展锐交付物进行使用，任何人对紫光展锐交付物的修改、定制化或违反说明文档的指引对紫光展锐交付物进行使用造成的任何损失由其自行承担。紫光展锐交付物中的性能指标、测试结果和参数等，均为在紫光展锐内部研发和测试系统中获得的，仅供参考，若任何人需要对交付物进行商用或量产，需要结合自身的软硬件测试环境进行全面的测试和调试。

Unisoc Confidential For hiar

紫光展锐（上海）科技有限公司



前言

概述

本文档对摄像头模组 PDAF 标定过程（主要包括 SPC 和 DCC 两部分）、相关的参数配置文件以及标定过程中的常见问题及解决办法进行说明，并提供了应用示例代码供参考。

读者对象


本文适用于与展锐合作的模组厂，以及想要了解 PDAF 标定的展锐内部人员。



缩略语

缩略语	英文全名	中文解释
AF	Auto Focus	自动对焦
AWB	Auto White Balance	自动白平衡
BLC	Black Level Correction	黑电平校正
CDAF	Contrast Detection Auto Focus	反差对焦
DCC	Defocus Conversion Coefficient	离焦转换系数
LSC	Lens shading correction	镜头阴影校正
OTP	One-Time-Programmable Memory	一次性可编程存储器
PD	Phase Detection	相位检测
PDAF	Phase Detection Auto Focus	相位对焦
SPC	Shield Pixel Correction	掩蔽像素校正
VCM	Voice Coil Motor	音圈马达

符号约定

在本文中可能出现下列标志，它所代表的含义如下。

符号	说明
 说明	用于突出重要/关键信息、补充信息和小窍门等。

符号	说明
	“说明”不是安全警示信息，不涉及人身、设备及环境伤害。
	用于突出容易出错的操作。 “注意”不是安全警示信息，不涉及人身、设备及环境伤害。
	用于可能无法恢复的失误操作。 “警告”不是危险警示信息，不涉及人身及环境伤害。

变更信息

文档版本	发布日期	修改说明
V1.0.0	2017-01-06	第一次正式发布。
V1.0.1	2017-01-10	在 DCC 输入参数文件中添加 BAYER。
V1.1.0	2017-03-08	增加验证过程。
V1.1.1	2017-03-21	添加 DCC 输入参数
V1.2	2017-07-10	修改 SPC OTP 大小。
V1.3	2018-02-28	增加全像素双核对焦内容。
V1.3.1	2018-09-25	调整 DCC raw 图数量。
V1.4	2019-07-2	1. PDAF 文档合并 2. 更新文档模板
V1.5	2020-11-16	增加对 ERR_CODE_DCC_NEGATIVE 的说明。
V1.6	2021-03-03	1. 统一文档语言，将文档中的英文描述统一为中文。 2. 调整文档格式，优化文档结构。

关键字

PDAF 标定、PDAF/相位对焦、SPC/掩蔽像素校正、SPC 验证、DCC/离焦转换系数、DCC 验证

目 录

1 摄像头模组 PDAF 标定	1
1.1 SPC 标定	1
1.1.1 输入信息	2
1.1.2 参数设置	2
1.1.3 输出信息	6
1.1.4 返回值	6
1.2 SPC 验证	6
1.2.1 输入信息	7
1.2.2 参数设置	7
1.2.3 返回值	7
1.3 DCC 标定	8
1.3.1 输入信息	8
1.3.2 参数设置	9
1.3.3 输出信息	11
1.3.4 返回值	11
1.4 DCC 验证	11
1.4.1 输入信息	12
1.4.2 参数设置	12
1.4.3 输出信息	12
1.4.4 返回值	12
2 OTP 应用示例代码	14
3 PDAF 标定常见问题	25
4 参考文档	28

图目录

图 1-1 shield PD 像素分布示意图及相关参数设置.....	4
图 1-2 dual PD 像素分布示意图及相关参数设置	5

Unisoc Confidential For hiar

表目录

表 1-1 SPC 标定返回值说明	6
表 1-2 SPC 验证返回值说明	8
表 1-3 DCC 标定返回值说明	11
表 1-4 DCC 验证返回值说明	13

Unisoc Confidential For hiar

1 摄像头模组 PDAF 标定

摄像头模组 PDAF（Phase Detection Auto Focus，相位对焦）标定流程包括 SPC（Shield Pixel Correction，掩蔽像素校正）和 DCC（Defocus Conversion Coefficient，离焦转换系数）的标定。

大致的标定流程如下：

1. 拍摄当前模组在工厂生产线的 16bit raw 图。
2. 基于以上 raw 数据和相应的参数配置文件，生成 SPC 和 DCC 等数据。
3. 将 SPC 和 DCC 数据按照 OTP MAP 的要求写入到 OTP（例如 8K bytes）对应地址中。

说明

- 具体的 raw 图拍摄要求参考《Camera AWB 和 LSC 和 AF 和 PDAF 烧录规范》。
- 展锐提供名为 _spc_param_XXX.txt 和 _dcc_param_XXX.txt 的参数配置文件。文件名中的“XXX”部分代表模组厂使用的具体 sensor。

1.1 SPC 标定

以下函数用于计算 SPC 增益图（gain map）。该增益图由两组 SPC_MAP_W * SPC_MAP_H 大小的数据组成，分别对应左 PD 点（left pd_pixel）和右 PD 点（right pd_pixel）。

说明

参数 SPC_MAP_W 和 SPC_MAP_H 在参数文件 _spc_param_XXX.txt 中定义。

```
int cal_otp_pdaf_spc_init(const char *spc_param_setting)

int cal_otp_pdaf_spc_getraw(const char raw_image_path[], int ***raw_image)

int cal_otp_pdaf_spc(int **raw_image, unsigned char *otp_spc_out, unsigned char *otp_ppp_out)

int cal_otp_pdaf_spc_releaseraw(int **raw_image)

int cal_otp_pdaf_spc_deinit()
```

SPC 标定流程如下：

1. 调用函数 cal_otp_pdaf_spc_init(...)，初始化 SPC 过程。
2. 调用函数 cal_otp_pdaf_spc_getraw(...)，读取 raw 图。
3. 调用函数 cal_otp_pdaf_spc(...)，计算 SPC 数据。
4. 调用函数 cal_otp_pdaf_spc_releaseraw(...)，释放 raw 图。

5. 调用函数 `cal_otp_pdaf_spc_deinit()`，SPC 去初始化。

1.1.1 输入信息

1. 拍摄 2 张 SPC raw 图用于 SPC 标定。raw 图数据格式：尺寸 `IMAGE_W*IMAGE_H`，Bayer Pattern，存储格式为 16bit 按像素小端对齐。
2. 相应的参数配置文件（`_spc_param_xxx.txt`）。

说明

- 具体的 raw 图拍摄要求参考《Camera AWB 和 LSC 和 AF 和 PDAF 烧录规范》。
- `IMAGE_W*IMAGE_H`，Bayer Pattern 的设置与 `_spc_param_xxx.txt` 中保持一致。

1.1.2 参数设置

展锐提供配置好的 SPC 参数文件 `_spc_param_xxx.txt`，供模组厂使用。配置文件中的参数说明如下。

- 图像相关参数

参数	说明
<code>IMAGE_W</code>	raw 图宽度
<code>IMAGE_H</code>	raw 图高度
<code>BAYER</code>	raw 图 Bayer Pattern 设置：Gr=0，R=1，B=2，Gb=3
<code>blc_r</code>	R 通道 BLC（Black Level Correction，黑电平校正）
<code>blc_gr</code>	Gr 通道 BLC
<code>blc_gb</code>	Gb 通道 BLC
<code>blc_b</code>	B 通道 BLC
<code>PD_CHANNEL</code>	PD 通道：Gr=0，R=1，B=2，Gb=3

- PD 坐标参数

参数	说明
<code>PD_BEGIN_X</code>	PD 区域的起始水平坐标 X
<code>PD_BEGIN_Y</code>	PD 区域的起始垂直坐标 Y
<code>PD_AREA_W</code>	PD 区域宽度
<code>PD_AREA_H</code>	PD 区域高度
<code>PD_UNIT_W</code>	单个 PD unit 的宽度
<code>PD_UNIT_H</code>	单个 PD unit 的高度
<code>PD_PAIR_W</code>	PD 对的宽度

参数	说明
PD_PAIR_H	PD 对的高度
PD_PAIR_NUM_UNIT	每个 unit 中所包含的 PD 对数。
LEFT_PD_OFFSET_X	PD unit 中左 PD 点的相对水平位置。其格式如下： LEFT_PD_OFFSET_X[0], LEFT_PD_OFFSET_X[1], ..., LEFT_PD_OFFSET_X[PD_PAIR_NUM_UNIT-1]
LEFT_PD_OFFSET_Y	PD unit 中左 PD 点的相对垂直位置。其格式如下： LEFT_PD_OFFSET_Y[0], LEFT_PD_OFFSET_Y[1], ..., LEFT_PD_OFFSET_Y[PD_PAIR_NUM_UNIT-1]
RIGHT_PD_OFFSET_X	PD unit 中右 PD 点的相对水平位置。其格式如下： RIGHT_PD_OFFSET_X[0], RIGHT_PD_OFFSET_X[1], ..., RIGHT_PD_OFFSET_X[PD_PAIR_NUM_UNIT-1]
RIGHT_PD_OFFSET_Y	PD unit 中右 PD 点的相对垂直位置。其格式如下： RIGHT_PD_OFFSET_Y[0], RIGHT_PD_OFFSET_Y[1], ..., RIGHT_PD_OFFSET_Y[PD_PAIR_NUM_UNIT-1]

图 1-1 是 shield PD 像素分布示意图。每个 PD unit 都由 16x16 个像素点构成，包含 4 个 PD 对，每个 PD 对包含一个左 PD 点和一个右 PD 点。对应的参数设置如下：

- PD_UNIT_W 16
- PD_UNIT_H 16
- PD_PAIR_NUM_UNIT 4
- LEFT_PD_OFFSET_X[0] 3
- LEFT_PD_OFFSET_X[1] 10
- LEFT_PD_OFFSET_X[2] 1
- LEFT_PD_OFFSET_X[3] 12
- LEFT_PD_OFFSET_Y[0] 2
- LEFT_PD_OFFSET_Y[1] 3
- LEFT_PD_OFFSET_Y[2] 12
- LEFT_PD_OFFSET_Y[3] 13
- RIGHT_PD_OFFSET_X[0] 4
- RIGHT_PD_OFFSET_X[1] 11
- RIGHT_PD_OFFSET_X[2] 2
- RIGHT_PD_OFFSET_X[3] 13
- RIGHT_PD_OFFSET_Y[0] 5
- RIGHT_PD_OFFSET_Y[1] 6
- RIGHT_PD_OFFSET_Y[2] 9
- RIGHT_PD_OFFSET_Y[3] 10
- PD_PAIR_W 8
- PD_PAIR_H 8

图1-1 shield PD 像素分布示意图及相关参数设置

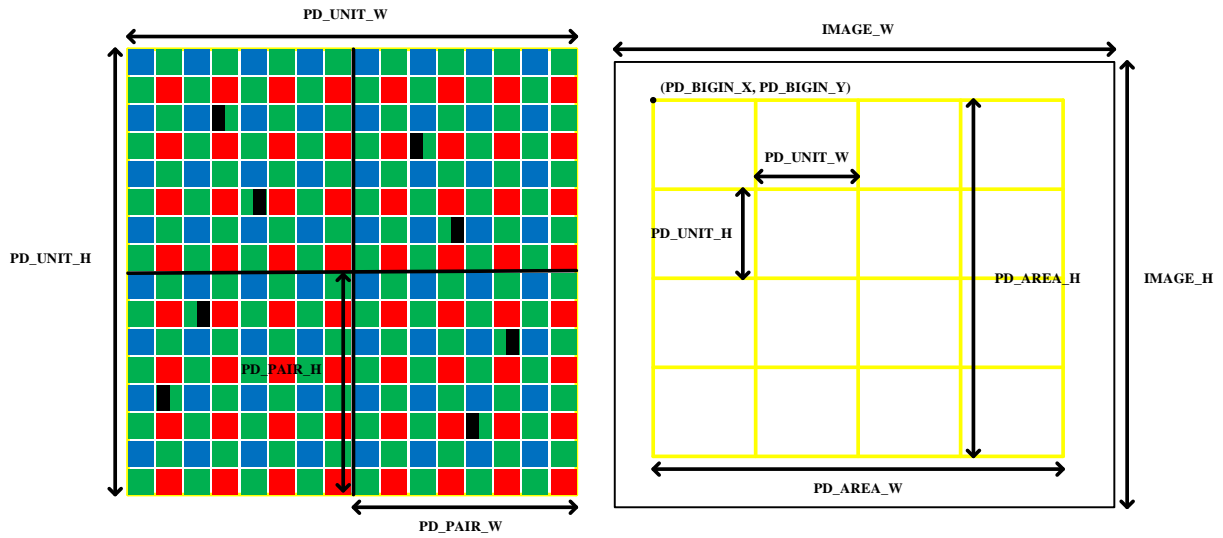
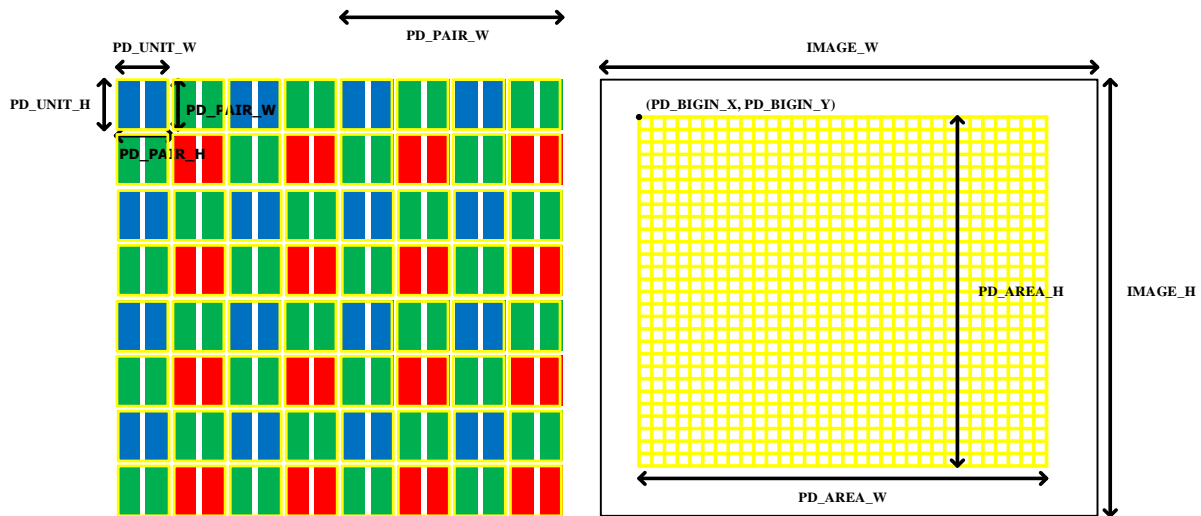


图 1-2 是 dual PD 像素分布示意图。每个像素由半个右像素和半个左像素组成，因此每个 PD unit 都是一个 1x1 的区域。对应的参数设置如下：

- PD_UNIT_W 1
- PD_UNIT_H 1
- PD_PAIR_NUM_UNIT 1
- LEFT_PD_OFFSET_X[0] 0
- LEFT_PD_OFFSET_Y[0] 0
- RIGHT_PD_OFFSET_X[0] 0
- RIGHT_PD_OFFSET_Y[0] 0
- PD_PAIR_W 1
- PD_PAIR_H 1

图1-2 dual PD 像素分布示意图及相关参数设置



• SPC 增益图参数

参数	说明
SPC_MAP_W	SPC 增益图宽度
SPC_MAP_H	SPC 增益图高度
SPC_VERIFY_MARGIN	SPC 验证误差阈值

• 亮度检查参数

参数	说明
CHECKER_W_SIZE_BY_IMAGE	所检查区域的窗口宽度
CHECKER_H_SIZE_BY_IMAGE	所检查区域的窗口高度
CHECKER_TARGET_LUMA	raw 图目标亮度
CHECKER_TARGET_MARGIN	允许的图像亮度的波动范围
CHECKER_LUMA_THRESHOLD	PD 点的亮度验证阈值
CHECKER_TARGET_LUMA_VERIFY	用于 SPC 验证的 raw 图目标亮度
CHECKER_TARGET_MARGIN_VERIFY	SPC 验证过程中允许的图像亮度的波动范围

• 其他参数

参数	说明
DEBUG_MODE	debug 类型，可控制生成 debug 文件。

参数	说明
	默认设置为 0，需检查 spc_data.csv 时，可设置为 1。

1.1.3 输出信息

输出项	说明
spc_map	SPC 数据，数据大小为 2*SPC_MAP_W* SPC_MAP_H

1.1.4 返回值

如果返回值为 0，说明标定通过。SPC 标定过程可能出现的返回值含义如表 1-1 所示。

说明

该流程中每个主要函数的返回值都被定义为 rtn，用于指示该函数的运行状态。若正常运行通过，返回值为 0。若返回其他值，则说明有异常退出的问题，具体可参考下表及 3 PDAF 标定常见问题中的说明。

表1-1 SPC 标定返回值说明

返回值	说明
0	SPC 标定通过
1	读取 raw 图失败
2	检查 raw 图亮度失败
3	PD 坐标验证失败

1.2 SPC 验证

下面的函数用于验证 SPC 增益图。该增益图由两组 SPC_MAP_W * SPC_MAP_H 大小的数据组成。

说明

参数 SPC_MAP_W 和 SPC_MAP_H 在输入的参数文件_spc_param_xxx.txt 中定义。

```
int cal_otp_pdaf_spc_init(const char *spc_param_setting)
```

```
int cal_otp_pdaf_spc_getraw_v(const char raw_image_path[], int ***raw_image)
```

```
int cal_otp_pdaf_spc_verify(int **raw_image, unsigned char *otp_spc_out, unsigned char *otp_ppp_out)
```

```
int cal_otp_pdaf_spc_releaseraw_v (int **raw_image)
```

```
int cal_otp_pdaf_spc_deinit ()
```

SPC 验证流程如下：

1. 调用函数 `cal_otp_pdaf_spc_init (...)`，初始化 SPC 过程。
2. 调用函数 `cal_otp_pdaf_spc_getraw_v (...)`，读取 raw 图。
3. 调用函数 `cal_otp_pdaf_spc_verify (...)`，验证 SPC 增益图。
4. 调用函数 `cal_otp_pdaf_spc_releaseraw_v (...)`，释放 raw 图。
5. 调用函数 `cal_otp_pdaf_spc_deinit ()`，去初始化。

1.2.1 输入信息

1. 1 张 raw 图：尺寸 `IMAGE_W*IMAGE_H`，Bayer Pattern，存储格式为 16bit 按像素小端对齐。
2. 相应的参数配置文件（`_spc_param_xxx.txt`）。
3. `spc_map`：SPC 标定阶段得到的 SPC 数据。

说明

- 具体的 raw 图拍摄要求参考《Camera AWB 和 LSC 和 AF 和 PDAF 烧录规范》。
- `IMAGE_W*IMAGE_H`，Bayer Pattern 的设置与 `_spc_param_xxx.txt` 中保持一致。

1.2.2 参数设置

SPC 验证过程中的参数设置与 SPC 标定过程相同，详细信息参见 [1.1.1 输入信息](#)

3. 拍摄 2 张 SPC raw 图用于 SPC 标定。raw 图数据格式：尺寸 `IMAGE_W*IMAGE_H`，Bayer Pattern，存储格式为 16bit 按像素小端对齐。
4. 相应的参数配置文件（`_spc_param_xxx.txt`）。

说明

- 具体的 raw 图拍摄要求参考《Camera AWB 和 LSC 和 AF 和 PDAF 烧录规范》。
- `IMAGE_W*IMAGE_H`，Bayer Pattern 的设置与 `_spc_param_xxx.txt` 中保持一致。

参数设置。

1.2.3 返回值

如果返回值为 0，说明验证通过。SPC 验证过程可能出现的返回值含义如[表 1-2](#) 所示。

说明

该流程中每个主要函数的返回值都被定义为 `rtn`，用于指示该函数的运行状态。若正常运行通过，返回值为 0。若返回其他值，则说明有异常退出的问题，具体可参考下表及 [3 PDAF 标定常见问题](#) 中的说明。

表1-2 SPC 验证返回值说明

返回值	说明
0	SPC 验证通过
8	SPC 验证失败

1.3 DCC 标定

下面的函数用于计算离焦转换系数（DCC）表，共有 $DCC_XKNOT_NUM * DCC_YKNOT_NUM$ 个数据。

说明

参数 DCC_XKNOT_NUM 和 DCC_YKNOT_NUM 在输入的参数文件 `_dcc_param_xxx.txt` 中定义。

```
int cal_otp_pdaf_dcc_init(const char *dcc_param_setting)
```

```
int cal_otp_pdaf_dcc_getraw(const char *raw_image_path[], int ***raw_image, int raw_num)
```

```
int cal_otp_pdaf_dcc(int **raw_image, unsigned char *otp_spd_in, int VCM_LOW_BND, int VCM_UPP_BND, unsigned char *otp_dcc_out, unsigned char *otp_ppp_out)
```

```
int cal_otp_pdaf_dcc_releaseraw(int **raw_image)
```

```
int cal_otp_pdaf_dcc_deinit()
```

DCC 标定流程如下：

1. 调用函数 `cal_otp_pdaf_dcc_init(...)`，初始化 DCC 过程。
2. 调用函数 `cal_otp_pdaf_dcc_getraw(...)`，读取 raw 图。
3. 调用函数 `cal_otp_pdaf_dcc(...)`，计算 DCC 数据。
4. 调用函数 `cal_otp_pdaf_dcc_releaseraw(...)`，释放 raw 图。
5. 调用函数 `cal_otp_pdaf_dcc_deinit()`，去初始化。

1.3.1 输入信息

1. 6 张 raw 图：尺寸 $IMAGE_W * IMAGE_H$ ，Bayer Pattern，存储格式为 16bit 按像素小端对齐。每张 raw 图都对应 VCM_LOW_BND 和 VCM_UPP_BND 之间的一个特定 VCM（Voice Coil Motor，音圈马达）位置。
2. 相应的参数配置文件（`_dcc_param_xxx.txt`）。
3. VCM_LOW_BND 和 VCM_UPP_BND 的数值。

4. spc_map: SPC 标定阶段生成的 SPC 数据。

说明

- 具体的 raw 图拍摄要求参考《Camera AWB 和 LSC 和 AF 和 PDAF 烧录规范》。
- IMAGE_W*IMAGE_H, Bayer Pattern 的设置与_dcc_param_xxx.txt 中保持一致。

1.3.2 参数设置

展锐提供配置好的 DCC 参数文件_dcc_param_xxx.txt, 给模组厂使用。配置文件中的参数说明如下。

- 图像相关参数

参数	说明
IMAGE_W	raw 图宽度
IMAGE_H	raw 图高度
BAYER	raw 图 Bayer Pattern 设置: Gr=0, R=1, B=2, Gb=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

- PD 坐标参数

参数	说明
PD_BEGIN_X	PD 区域的起始水平坐标 X
PD_BEGIN_Y	PD 区域的起始垂直坐标 Y
PD_AREA_W	PD 区域宽度
PD_AREA_H	PD 区域高度
PD_UNIT_W	单个 PD unit 宽度
PD_UNIT_H	单个 PD unit 高度
PD_PAIR_W	PD 对之间的水平距离
PD_PAIR_H	PD 对之间的垂直距离
PD_PAIR_NUM_UNIT	每个 PD unit 中所包含的 PD 对数
LEFT_PD_OFFSET_X	PD unit 中左 PD 点的相对水平位置。其格式如下: LEFT_PD_OFFSET_X[0], LEFT_PD_OFFSET_X[1], ..., LEFT_PD_OFFSET_X[PD_PAIR_NUM_UNIT-1]
LEFT_PD_OFFSET_Y	PD unit 中左 PD 点的相对垂直位置。其格式如下:

参数	说明
	LEFT_PD_OFFSET_Y[0], LEFT_PD_OFFSET_Y[1], ..., LEFT_PD_OFFSET_Y[PD_PAIR_NUM_UNIT-1]
RIGHT_PD_OFFSET_X	PD unit 中右 PD 点的相对水平位置。其格式如下： RIGHT_PD_OFFSET_X[0], RIGHT_PD_OFFSET_X[1], ..., RIGHT_PD_OFFSET_X[PD_PAIR_NUM_UNIT-1]
RIGHT_PD_OFFSET_Y	PD pattern 中右 PD 点的相对垂直位置。其格式如下： RIGHT_PD_OFFSET_Y[0], RIGHT_PD_OFFSET_Y[1], ..., RIGHT_PD_OFFSET_Y[PD_PAIR_NUM_UNIT-1]

- SPC 增益图参数

参数	说明
SPC_MAP_W	SPC 增益图宽度
SPC_MAP_H	SPC 增益图高度

- 亮度检查参数

参数	说明
CHECKER_W_SIZE_BY_IMAGE	所检查区域的窗口宽度
CHECKER_H_SIZE_BY_IMAGE	所检查区域的窗口高度
CHECKER_TARGET_LUMA	raw 图目标亮度
CHECKER_TARGET_MARGIN	允许的图像亮度的波动范围

- DCC 图参数

参数	说明
DCC_XKNOT_NUM	水平方向的 DCC 系数个数
DCC_YKNOT_NUM	垂直方向的 DCC 系数个数

- 其他参数

参数	说明
DEBUG_MODE	debug 类型，可控制生成 debug 文件。 默认设置为 0，需检查 dcc_data.csv 时，可设置为 1。
REVERSAL_MODULE	模组类型。0：常规模组；1：反转模组。

1.3.3 输出信息

输出项	说明
dcc_map	DCC 数据，数据大小为 DCC_XKNOT_NUM * DCC_YKNOT_NUM

1.3.4 返回值

如果返回值为 0，说明标定通过。DCC 标定过程中可能出现的返回值含义如表 1-3 所示。

说明

该流程中每个主要函数的返回值都被定义为 rtn，用于指示该函数的运行状态。若正常运行通过，返回值为 0。若返回其他值，则说明有异常退出的问题，具体可参考下表及 3 PDAF 标定常见问题中的说明。

表1-3 DCC 标定返回值说明

返回值	说明
0	DCC 标定通过
1	读取 raw 图失败
2	检查 raw 图亮度失败
3	PD 坐标验证失败
5	DCC 验证失败
6	获取 SPC 增益图失败

1.4 DCC 验证

下面的函数用于验证离焦转换系数表。

```
int cal_otp_pdaf_dcc_init(std::string dcc_param_setting)

int cal_otp_pdaf_dcc_getraw_v(std::string raw_image_path[], int ***raw_image)

int cal_otp_pdaf_get_infocus_vcm(int **raw_image, unsigned char *otp_spc_in, unsigned char *otp_dcc_in, int vcm_offset)

int cal_otp_pdaf_dcc_releaseraw_v(int **raw_image)

int cal_otp_pdaf_dcc_deinit()
```

DCC 验证流程如下：

1. 调用函数 `cal_otp_pdaf_dcc_init(...)`，初始化 DCC 过程。
2. 调用函数 `cal_otp_pdaf_dcc_getraw_v(...)`，读取 raw 图。
3. 调用函数 `cal_otp_pdaf_get_infocus_vcm(...)`，获取偏离准焦位的 code 值。
4. 调用函数 `cal_otp_pdaf_dcc_releaseraw_v(...)`，释放 raw 图。
5. 调用函数 `cal_otp_pdaf_dcc_deinit()`，去初始化。
6. 按照 `vcm_offset` 计算 PDAF 对焦位置。
7. 手动输入 CDAF（Contrast Detection Auto Focus，反差对焦）准焦位，评估 CDAF 和 PDAF 之间的偏差。如果差值的绝对值小于 30，则说明模组符合 PDAF 要求，否则模组不符合 PDAF 要求。

1.4.1 输入信息

1. 1 张 raw 图：尺寸 `IMAGE_W*IMAGE_H`，16bit 按像素小端对齐。
2. `current_vcm_pos` 的值，raw 图 VCM 位置，对应 `VCM_LOW_BND` 与 `VCM_UPP_BND` 之间的一个 VCM 位置。
3. 相应的参数配置文件（`_dcc_param_xxx.txt`）。
4. `spc_map`：SPC 标定阶段生成的 SPC 数据。
5. `dcc_map`：DCC 标定阶段生成的 DCC 数据。
6. `cdaf_infocus_pos`：CDAF 预测的准焦位。

说明

- 具体的 raw 图拍摄要求参考《Camera AWB 和 LSC 和 AF 和 PDAF 烧录规范》。
- `IMAGE_W*IMAGE_H` 的设置与 `_dcc_param_xxx.txt` 中保持一致。

1.4.2 参数设置

DCC 验证过程中的参数设置与 DCC 标定过程相同，详细信息参见 [1.3.2 参数设置](#)。

1.4.3 输出信息

输出项	说明
<code>vcm_offset</code>	VCM 偏移值，由准焦 VCM 位置减当前 VCM 位置得到。
<code>infocus_pos_error</code>	PDAF 预测的准焦位与 CDAF 预测结果的偏差。

1.4.4 返回值

如果返回值为 0，说明验证通过。DCC 验证过程中可能出现的返回值含义如 [表 1-4](#) 所示。

说明

该流程中每个主要函数的返回值都被定义为 `rtn`，用于指示该函数的运行状态。若正常运行通过，返回值为 0。若返回其他值，则说明有异常退出的问题，具体可参考下表及 **3 PDAF 标定常见问题** 中的说明。

表1-4 DCC 验证返回值说明

返回值	说明
0	DCC 验证通过
1	读取 raw 图失败
2	检查 raw 图亮度失败
3	PD 坐标验证失败
4	DCC 过程中获取 PD 失败
5	DCC 验证失败
6	获取 SPC 增益图失败
7	检查 PD 位置失败
8	SPC 验证失败
9	读取参数失败
10	DCC 是负值
11	CDAF 与 PDAF 之间的偏差大于 30

2 OTP 应用示例代码

参考 sample_code/main.cpp 中的示例代码。需要输入的信息都以红色加粗字体标注在示例代码中，分别对应 SPC 和 DCC 过程中的输入信息（[1.1.1](#)、[1.2.1](#)、[1.3.1](#)、[1.4.1](#)）。

下面具体罗列出需要输入的信息：

- SPC 参数配置文件的相对路径。
- 2 张 SPC 标定 raw 图文件的相对路径。
- 1 张 SPC 验证 raw 图文件的相对路径。
- DCC 参数配置文件的相对路径。
- 6 张 DCC 标定 raw 图文件的相对路径。
- VCM_LOW_BND 值，即 VCM 位置的下限值，这里对应为第一张 DCC 标定 raw 图的 VCM 位置。
- VCM_UPP_BND 值，即 VCM 位置的上限值，这里对应为最后一张 DCC 标定 raw 图的 VCM 位置。
- 1 张 DCC 验证 raw 图文件的相对路径。
- current_vcm_pos 值，即 DCC 验证 raw 图文件对应的 VCM 位置，位于 VCM_LOW_BND 与 VCM_UPP_BND 之间。
- cdaf_infocus_pos 值，即 CDAF 准焦位置。

说明

上述提到的 raw 图和参数配置文件都放置于与 sample_code.sln 相同的路径下。

```
int main(int argc, char* argv[])
{
    int rtn = 0;

    HINSTANCE hDll = NULL;

    FNUC_CAL_OTP_PDAF_SPC_INIT                cal_otp_pdaf_spc_init                = NULL;
    FNUC_CAL_OTP_PDAF_SPC_GETRAW              cal_otp_pdaf_spc_getraw              = NULL;
    FNUC_CAL_OTP_PDAF_SPC_GETRAW_V           cal_otp_pdaf_spc_getraw_v           = NULL;
    FNUC_CAL_OTP_PDAF_SPC                    cal_otp_pdaf_spc                    = NULL;
    FNUC_CAL_OTP_PDAF_SPC_VERIFY              cal_otp_pdaf_spc_verify              = NULL;
    FNUC_CAL_OTP_PDAF_SPC_RELEASERAW          cal_otp_pdaf_spc_releaseraw          = NULL;
    FNUC_CAL_OTP_PDAF_SPC_RELEASERAW_V       cal_otp_pdaf_spc_releaseraw_v       = NULL;
    FNUC_CAL_OTP_PDAF_SPC_DEINIT              cal_otp_pdaf_spc_deinit              = NULL;

    FNUC_CAL_OTP_PDAF_DCC_INIT                cal_otp_pdaf_dcc_init                = NULL;
```

```

FNUC_CAL_OTP_PDAF_DCC_GETRAW          cal_otp_pdaf_dcc_getraw          = NULL;
FNUC_CAL_OTP_PDAF_DCC_GETRAW_V        cal_otp_pdaf_dcc_getraw_v        = NULL;
FNUC_CAL_OTP_PDAF_DCC                 cal_otp_pdaf_dcc                 = NULL;
FNUC_CAL_OTP_PDAF_DCC_GET_INFOCUS_VCM  cal_otp_pdaf_dcc_get_infocus_vcm  = NULL;
FNUC_CAL_OTP_PDAF_DCC_RELEASERAW       cal_otp_pdaf_dcc_releaseraw       = NULL;
FNUC_CAL_OTP_PDAF_DCC_RELEASERAW_V     cal_otp_pdaf_dcc_releaseraw_v     = NULL;
FNUC_CAL_OTP_PDAF_DCC_DEINIT           cal_otp_pdaf_dcc_deinit           = NULL;

// pd pixel profile parameter
unsigned char otp_ppp_out[PD_PROFILE_TABLE_NUM];
unsigned char infocus_pos_error = 0;

// spc parameter
std::string spc_raw_path[SPC_RAW_SEQUENCE_NUMBER] = {"spc_01.raw" /*"dnp_0.raw"*/,
"spc_0_2.raw" /*"dnp_1.raw"*/,          }; // input raw file path, NOT use absolute path but use relative path
std::string spc_parameter = "_spc_param_sensorname_normal.txt"; // input spc parameter
path, NOT use absolute path but use relative path
unsigned char spc_gain_map[SPC_MAP_W*SPC_MAP_H*2*2];
// spc verify parameter
std::string spc_verify_raw_path[SPC_RAW_SEQUENCE_NUMBER_V] = {"spc_verify.raw"/*"dnp_2.raw"*/ };
// input raw file path, NOT use absolute path but use relative path
std::string spc_verify_parameter = "_spc_param_sensorname_normal.txt"; // input spc parameter path,
NOT use absolute path but use relative path
unsigned char otp_spc_in[SPC_MAP_W*SPC_MAP_H*2*2];
// dcc parameter
std::string dcc_raw_path[DCC_RAW_SEQUENCE_NUMBER] = { "dcc_01.raw",
"dcc_02.raw",
"dcc_03.raw",
"dcc_04.raw",
"dcc_05.raw",
"dcc_06.raw",}; // input raw file path, NOT use
absolute path but use relative path
std::string dcc_parameter = "_dcc_param_sensorname_normal.txt"; // input dcc parameter path, NOT
use absolute path but use relative path
unsigned char dcc_coef_map[DCC_XKNOT_NUM*DCC_YKNOT_NUM*2];
int VCM_LOW_BND = 270/*400300*/;
int VCM_UPP_BND = 515/*650550*/;
// dcc verify parameter
std::string dcc_verify_raw_path[DCC_RAW_SEQUENCE_NUMBER_V] = { "dcc_verify.raw" }; // input raw file

```

path, NOT use absolute path but use relative path

```
std::string dcc_verify_parameter = "_dcc_param_sensorname_normal.txt"; // input dcc parameter path,
NOT use absolute path but use relative path

unsigned char otp_dcc_in[DCC_XKNOT_NUM*DCC_YKNOT_NUM*2];

int current_vcm_pos = 466; //the vcm position of dcc_verify.raw

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

/*get spc function pointer*/
cal_otp_pdaf_spc_init = (FNUC_CAL_OTP_PDAF_SPC_INIT)GetProcAddress(hDll, "cal_otp_pdaf_spc_init");
if (cal_otp_pdaf_spc_init == NULL){
    printf("Failed to GetProcAddress cal_otp_pdaf_spc_init\n");
    goto EXIT;
}

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

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

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

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

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

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

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

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

/*get dcc function pointer*/
cal_otp_pdaf_dcc_init = (FNUC_CAL_OTP_PDAF_DCC_INIT)GetProcAddress(hDll, "cal_otp_pdaf_dcc_init");
if (cal_otp_pdaf_dcc_init == NULL){
    printf("Failed to GetProcAddress cal_otp_pdaf_dcc_init\n");
    goto EXIT;
}
```



```
}

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

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

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

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

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

cal_otp_pdaf_dcc_releaseraw_v = (FNUC_CAL_OTP_PDAF_DCC_RELEASERAW_V)GetProcAddress(hDll,
"cal_otp_pdaf_dcc_releaseraw_v");
if (cal_otp_pdaf_dcc_releaseraw_v == NULL){
    printf("Failed to GetProcAddress cal_otp_pdaf_dcc_releaseraw_v\n");
}
```

```
    goto EXIT;
}

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

// 1. spc calibration

// 1.1 spc initialization

```
if (SPC_DCC_SWITCH < 2){
    rtn = cal_otp_pdaf_spc_init(spc_parameter.c_str());
    if (rtn != ERR_CODE_PASS){
        return rtn;
    }
}
```

// 1.2 get raw to image buffer

```
int **raw_image_spc_calc = NULL;
const char *spc_calc_raw_path[SPC_RAW_SEQUENCE_NUMBER] = {
    spc_raw_path[0].c_str(),
    spc_raw_path[1].c_str()
};
rtn = cal_otp_pdaf_spc_getraw(spc_calc_raw_path, &raw_image_spc_calc);
if (rtn != ERR_CODE_PASS){
    goto RELEASE_SPC;
}
```

// 1.3 spc calculation

```
rtn = cal_otp_pdaf_spc(raw_image_spc_calc, spc_gain_map, otp_ppp_out);
```

RELEASE_SPC:

// 1.4 release image buffer

```
cal_otp_pdaf_spc_releaseraw (raw_image_spc_calc);
```

// 1.5 spc deinitialization

```
cal_otp_pdaf_spc_deinit ();
```

```
if(rtn!=ERR_CODE_PASS){
    return rtn;
}
```

```
}

// output SPC calibration .bin data (i.e., write spc data to OTP)
FILE *fp_otp_spc = fopen ("otp_spc.bin","wb");
unsigned char spc_w = SPC_MAP_W;
unsigned char spc_h = SPC_MAP_H;
fwrite (&spc_w, 1, 1, fp_otp_spc);
fwrite (&spc_h, 1, 1, fp_otp_spc);
fwrite (spc_gain_map, 1, spc_w*spc_h*2*2, fp_otp_spc);
fclose (fp_otp_spc);

// 2. spc verification
// 2.1 spc initialization
rtn = cal_otp_pdaf_spc_init(spc_verify_parameter.c_str());
if(rtn!=ERR_CODE_PASS){
    return rtn;
}

// 2.2 get raw to image buffer
int **raw_image_spc_verify = NULL;
const char *spc_veri_raw_path[SPC_RAW_SEQUENCE_NUMBER_V] = {spc_verify_raw_path[0].c_str()};
rtn = cal_otp_pdaf_spc_getraw_v(spc_veri_raw_path, &raw_image_spc_verify);
if(rtn!=ERR_CODE_PASS){
    goto RELEASE_SPC_VERIFY;
}

// 2.3 copy spc data from the OTP to a buffer
for (int i = 0; i < SPC_MAP_W*SPC_MAP_H*2*2; i++)
    otp_spc_in[i] = spc_gain_map[i];

// 2.4 verification
rtn = cal_otp_pdaf_spc_verify(raw_image_spc_verify, otp_spc_in, otp_ppp_out); // CY: spc_gain_map need modify to 9x7x2
bytes

RELEASE_SPC_VERIFY:

// 2.5 release image buffer
cal_otp_pdaf_spc_releaseraw_v (raw_image_spc_verify);

// 2.6 spc deinitialization
cal_otp_pdaf_spc_deinit ();
```

```
        if (rtn != ERR_CODE_PASS){
            return rtn;
        }
    }
}

// 4. dcc calibration

// 4.1 dcc initialization
if (SPC_DCC_SWITCH == 0 || SPC_DCC_SWITCH == 2){
    rtn = cal_otp_pdaf_dcc_init(dcc_parameter.c_str());
    if (rtn != ERR_CODE_PASS){
        return rtn;
    }

    // 4.2 get raw to image buffer
    int **raw_image_dcc_calc = NULL;
    const char *dcc_calc_raw_path[DCC_RAW_SEQUENCE_NUMBER] = {
        dcc_raw_path[0].c_str(),
        dcc_raw_path[1].c_str(),
        dcc_raw_path[2].c_str(),
        dcc_raw_path[3].c_str(),
        dcc_raw_path[4].c_str(),
        dcc_raw_path[5].c_str(),
    };
    rtn = cal_otp_pdaf_dcc_getraw(dcc_calc_raw_path, &raw_image_dcc_calc, DCC_RAW_SEQUENCE_NUMBER);
    if (rtn != ERR_CODE_PASS){
        goto RELEASE_DCC;
    }

    // 4.3 copy spc data from the OTP to a buffer
    if (SPC_DCC_SWITCH == 2){
        FILE *fp_open_spc_forDCC = fopen("otp_spc.bin", "rb");
        fseek(fp_open_spc_forDCC, 2L, 0);
        fread(otp_spc_in, 1, (SPC_MAP_W*SPC_MAP_H * 2 * 2), fp_open_spc_forDCC);
        fclose(fp_open_spc_forDCC);
    }
    else
    {
        for (int i = 0; i < SPC_MAP_W*SPC_MAP_H * 2 * 2; i++)
            otp_spc_in[i] = spc_gain_map[i];
    }
}
```

```
// 4.4 dcc calculation
if(SPC_DCC_SWITCH == 2){
    FILE *fp_ppp_copy = fopen("otp_ppp.bin", "rb");
    fread(otp_ppp_out, 1, (PD_PROFILE_TABLE_NUM - 4), fp_ppp_copy);
    fclose(fp_ppp_copy);
}

rtn = cal_otp_pdaf_dcc(raw_image_dcc_calc, otp_spc_in, VCM_LOW_BND, VCM_UPP_BND, dcc_coef_map,
otp_ppp_out);

RELEASE_DCC:

// 4.5 release image buffer
cal_otp_pdaf_dcc_releaseraw (raw_image_dcc_calc);

// 4.6 dcc deinitialization
cal_otp_pdaf_dcc_deinit ();

if(rtn!=ERR_CODE_PASS){
    return rtn;
}

// output DCC calibration .bin data (i.e., write dcc data to OTP)
FILE *fp_otp_dcc = fopen ("otp_dcc.bin","wb");
unsigned char dcc_w = DCC_XKNOT_NUM;
unsigned char dcc_h = DCC_YKNOT_NUM;
fwrite (&dcc_w, 1, 1, fp_otp_dcc);
fwrite (&dcc_h, 1, 1, fp_otp_dcc);
fwrite (dcc_coef_map, 1, dcc_w*dcc_h*2, fp_otp_dcc);
fclose (fp_otp_dcc);

// 5. dcc calibration verify
// 5.1 dcc initialization
rtn = cal_otp_pdaf_dcc_init (dcc_verify_parameter.c_str());
if(rtn!=ERR_CODE_PASS){
    return rtn;
}

// 5.2 get raw to image buffer
int **raw_image_dcc_verify = NULL;

const char *dcc_veri_raw_path[DCC_RAW_SEQUENCE_NUMBER_V] = {dcc_verify_raw_path[0].c_str()};
```

```

rtn = cal_otp_pdaf_dcc_getraw_v(dcc_veri_raw_path, &raw_image_dcc_verify);
if(rtn!=ERR_CODE_PASS){
    goto RELEASE_DCC_VERIFY;
}
// 5.3 Copy dcc data from the OTP to a buffer
for (int i = 0; i < DCC_XKNOT_NUM*DCC_YKNOT_NUM*2; i++)
    otp_dcc_in[i] = dcc_coef_map[i];
// 5.4 Copy spc data from the OTP to a buffer
if (SPC_DCC_SWITCH == 0){
    for (int i = 0; i < SPC_MAP_W*SPC_MAP_H*2*2; i++)
        otp_spc_in[i] = spc_gain_map[i];
}
// 5.5 dcc verification
int vcm_offset;
rtn = cal_otp_pdaf_dcc_get_infocus_vcm (raw_image_dcc_verify, otp_spc_in, otp_dcc_in, vcm_offset);
int pdaf_infocus_pos = current_vcm_pos + vcm_offset;

RELEASE_DCC_VERIFY:
// 5.6 release image buffer
cal_otp_pdaf_dcc_releaseraw_v(raw_image_dcc_verify);
// 5.7 dcc deinitialization
cal_otp_pdaf_dcc_deinit();
// 5.8 do CDAF, get the vcm position with the peakest contrast value to "cdaf_infocus_pos"
int cdaf_infocus_pos = 366;
// 5.9 joint PDAF and CDAF infocus verification,
infocus_pos_error = (abs(pdaf_infocus_pos - cdaf_infocus_pos) > 0xff) ? 0xff : abs(pdaf_infocus_pos -
cdaf_infocus_pos);
printf("the diff between pdaf and cdaf is [%d]\n", infocus_pos_error);
if (infocus_pos_error >= error_scope){
    rtn = ERR_CODE_OUT_ERROR_SCOPE;
    return rtn;
}
}

// output Pd Pixel Profile calibration .bin data (i.e., write ppp data to OTP)
if (SPC_DCC_SWITCH == 1)
{

```

```
FILE *fp_otp_ppp = fopen("otp_ppp.bin", "wb");
fwrite(otp_ppp_out, 1, PD_PROFILE_TABLE_NUM - 4, fp_otp_ppp);
fclose(fp_otp_ppp);
}
else
{
    FILE *fp_otp_ppp = fopen("otp_ppp.bin", "wb");
    fwrite(otp_ppp_out, 1, PD_PROFILE_TABLE_NUM, fp_otp_ppp);
    fwrite(&infocus_pos_error, 1, 1, fp_otp_ppp);
    fclose(fp_otp_ppp);
}
printf("=====\n");
printf("=====calibration done!!!=====\n");
printf("=====\n");
EXIT:
if (hDll != NULL)
{
    FreeLibrary(hDll);
}

return rtn;
}
```

Unisoc Confidential For hiar

3

PDAF 标定常见问题

PDAF 标定过程中每个主要函数的返回值都被定义为 `rtn`，用于指示该函数的运行状态。若正常运行通过，返回值为 0。若返回其他值，则说明有异常退出的问题。下面对 PDAF 标定过程中返回值提示的常见问题及解决办法进行说明。

说明

有关各标定和验证流程中可能出现的返回值，参见前面对应的返回值章节。

1. `ERR_CODE_PASS = 0`

说明：表示通过标定，可以烧入 PDAF 的 OTP 数据。

2. `ERR_CODE_RAW_READ_FAIL = 1`

产生原因：在读取 raw 文件时，无法打开 SPC 的 raw 图文件或无法打开 DCC 的 raw 图文件。

解决办法：检查文件夹下是否包含 SPC 的 raw 图文件或 DCC 的 raw 图文件。

3. `ERR_CODE_RAW_LUMA_CHECK_FAIL = 2`

产生原因：在检查 raw 图的亮度时，所求亮度均值与阈值的 SAD（绝对误差和）超过了限定值。

解决办法：

- 检查 raw 图中心区域的像素值是否达到要求。若不达标，可适当调整拍摄环境的亮度。
- 检查配置文件中（`_spc_param_xxx.txt` 和 `_dcc_param_xxx.txt`）的参数设置是否合理：
 - `CHECKER_TARGET_LUMA` 和 `CHECKER_TARGET_MARGIN`：若在 SPC 标定阶段出错，需调整 `_spc_param_xxx.txt` 中的这两个参数；若在 DCC 标定阶段出错，需调整 `_dcc_param_xxx.txt` 中的这两个参数。
 - `CHECKER_TARGET_LUMA_VERIFY` 和 `CHECKER_TARGET_MARGIN_VERIFY`：在 SPC 验证阶段出错，需调整 `_spc_param_xxx.txt` 中的这两个参数。

dll 内部检查亮度是否足条件的方法如下：

`ABS(Average - CHECKER_TARGET_LUMA) > CHECKER_TARGET_MARGIN`

`ABS(Average - CHECKER_TARGET_LUMA_VERIFY) > CHECKER_TARGET_MARGIN_VERIFY`

- 一般 `CHECKER_TARGET_LUMA` 和 `CHECKER_TARGET_LUMA_VERIFY` 都是 800，不做修改。
- 如果一批模组的偏差较大，则可以适当地加大 `CHECKER_TARGET_MARGIN` 或者 `CHECKER_TARGET_MARGIN_VERIFY`。

4. `ERR_CODE_DCC_GET_PD_FAIL = 4`

产生原因：在从 raw 图中获取 PD 数据时，left PD、right PD 个数之和与整个 raw 图所包含的 PD 总数不符。

解决办法：根据 sensor 厂的 spec 文件，检查配置文件 `_spc_param_xxx.txt` 和 `_dcc_param_xxx.txt` 中设置 PD 位置的相关参数是否配置正确。需要检查的参数如下：

- `PD_AREA_W`
- `PD_AREA_H`

- PD_UNIT_W
- PD_UNIT_H
- PD_PAIR_NUM_UNIT
- PD_BEGIN_X
- PD_BEGIN_Y
- LEFT_PD_OFFSET_X 数组
- LEFT_PD_OFFSET_Y 数组
- RIGHT_PD_OFFSET_X 数组
- RIGHT_PD_OFFSET_Y 数组

5. ERR_CODE_DCC_VERIFICATION_FAIL = 5

产生原因 1: 检查中心区域和非中心区域的 DCC 线性度时，中心区域或非中心区域 DCC 线性度没有满足门限要求。

解决办法 1: 可尝试调整配置文件_dcc_param_xxx.txt 中的 DCC_VERIFICATION_MARGIN（中心区域的门限）和 DCC_VERIFICATION_MARGIN_B（非中心区域的门限）。非必要情况，不建议更改。若需要更改，请务必与展锐反馈。

这个是 DCC 线性度的检测，dll 会计算 DCC 每个不同 VCM 位置上 raw 图的相位差，然后使用 VCM 位置和相位差数据计算出 DCC 线性度。DCC 线性度值在 0~100 之间，值越大表示线性度越好。

在出现这个错误时，可以查看 dcc_data.csv 文件（dll 生成的文件），这个文件中可以看到如下数据（数值都在 0~100 之内）：

- 当下图中红色框内的数据小于 DCC_VERIFICATION_MARGIN 的个数达到算法库中约定的数目时会报 ERR_CODE_DCC_VERIFICATION_FAIL。
- 当下图中红色框外的数据小于 DCC_VERIFICATION_MARGIN_B 的个数达到算法库中约定的数目时也会报 ERR_CODE_DCC_VERIFICATION_FAIL。

所以把这两个门限调小可以提高良率。

表征 8x8 block 中 DCC 线性度的数值如下（来自 dcc_data.csv 文件）。

96.873505	95.04543	92.26898	94.74739	95.34676	89.47464	95.33508	95.19769
95.778091	93.51872	93.03234	95.15693	94.40894	97.72997	90.93629	95.3402
96.533524	96.31112	95.77654	94.76031	94.14622	93.14113	83.77274	89.28692
95.88588	93.2185	94.28064	96.04914	96.47371	91.13162	87.56189	97.60342
97.529045	95.29804	93.38566	95.47263	97.30237	92.00259	88.3937	92.16055
96.856186	95.62111	94.46411	94.08854	92.45606	89.57727	93.53636	94.02836
96.783676	96.76254	95.11193	94.37767	95.31429	93.88998	93.4943	86.46933
94.473198	94.07777	95.2402	95.0219	94.79042	94.30471	90.19744	92.36655

产生原因 2: 也有可能是 DCC 测试中在两边端点(infi,macro)的几个连续 VCM 位置不在马达的物理移动线性区域内导致的。

解决办法 2: 需要模组厂对模组做一些措施，尽量使 DCC 测试的几个 VCM 位置落在 infi~macro 之间的线性区域内，比如可以把 DCC 的 VCM 限制在 infi~macro 中间 60%至 90%的区间之内。

6. ERR_CODE_PD_POSITION_ERR = 7

产生原因: PD 像素的 (x,y) 坐标存在小于 0 或大于 raw 图的宽或高等情形的非法数值。

解决办法: 根据 sensor 厂的 spec 文件，检查配置文件_spc_param_xxx.txt 和_dcc_param_xxx.txt 中 raw 图大小的参数和设置 PD 位置的相关参数是否配置正确。需要检查的参数如下：

- IMAGE_W

- IMAGE_H
- PD_AREA_W
- PD_AREA_H
- PD_UNIT_W
- PD_UNIT_H
- PD_PAIR_NUM_UNIT
- PD_BEGIN_X
- PD_BEGIN_Y
- LEFT_PD_OFFSET_X 数组
- LEFT_PD_OFFSET_Y 数组
- RIGHT_PD_OFFSET_X 数组
- RIGHT_PD_OFFSET_Y 数组

7. ERR_CODE_SPC_VERIFICATION_FAIL = 8

产生原因：在 SPC 验证时，需要将 SPC 计算得到的 SPC 增益和 SPC 验证得到的 SPC 增益进行对比。如果两者之间差异较大，就会报错。

解决办法：

- a) 确定拍摄环境的光源是否稳定，AE 是否锁定。
- b) 尝试调整配置文件_spc_param_xxx.txt 中的 SPC_VERIFY_MARGIN 参数，调大 SPC_VERIFY_MARGIN 的数值，可以降低这个错误出现的概率。非必要情况，不建议对此进行更改；若需要更改，请务必与展锐反馈。

8. ERR_CODE_PARAM_READ_FAIL = 9

产生原因：读取配置文件_spc_param_xxx.txt 或_dcc_param_xxx.txt 失败。

解决办法：查看项目文件夹内是否包含该配置文件。

9. ERR_CODE_DCC_NEGATIVE = 10

产生原因：DCC 为负值。

解决办法：

- 若为常规模组 ($\text{dac}(\text{infinity}) < \text{dac}(\text{macro})$)，需检查 left_pd 和 right_pd 的坐标是否颠倒。
- 若为反转模组 ($\text{dac}(\text{infinity}) > \text{dac}(\text{macro})$)，可在_dcc_param.txt 中添加参数：
“REVERSAL_MODULE 1”

10. ERR_CODE_OUT_ERROR_SCOPE = 11

产生原因：CDAF 预测的准焦位与 PDAF 的偏差超过管控值 30。

解决办法：在保证 CDAF 预测准确的前提下，请检查 main.c 中所有的 DAC 值填写是否正确。确认一切无误后，若偏差依然超出管控，请与展锐反馈沟通。

4

参考文档

《Camera AWB 和 LSC 和 AF 和 PDAF 烧录规范》

Unisoc Confidential For hiar