MEDIATEK

PDAF Driver and Buf_mgr Porting Guide

Outline

- 1. PDAF Introduction
  - Phase Diffenece Princlple
  - PD Module and MTK INI File
- 2. PDAF Bring Up
  - Prepare + Driver + EEPROM + PD Buffer Manager + BPCI + Para
  - PDAF Porting Example of Case
- 3. PDAF Verification
- 4. PDAF Debug and Trouble Shooting

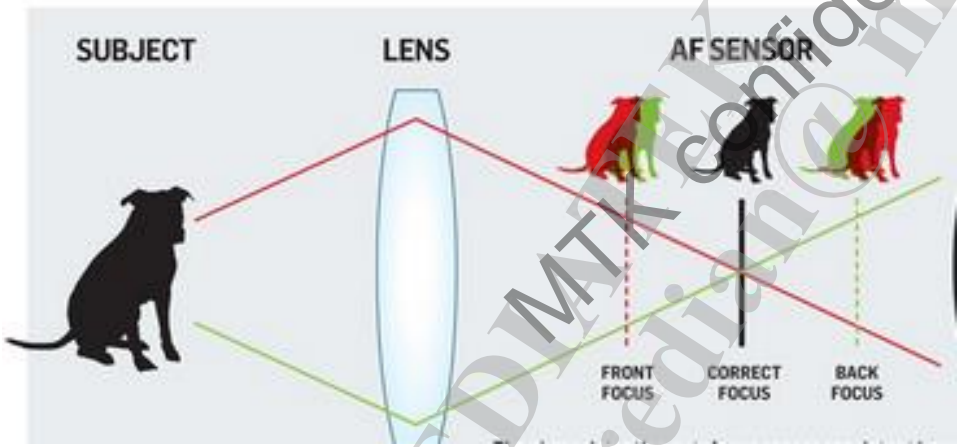
MEDIATEK

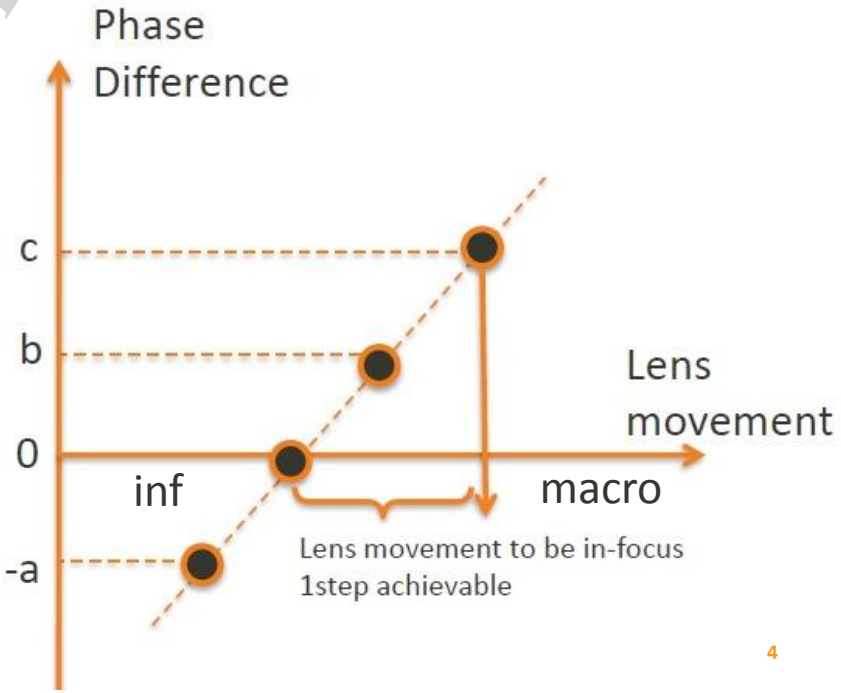
PDAF Introduction

PDAF 简介

- PDAF: Phase Detection Auto Focus
  - 通过比较L/R PD pixel构成的两幅图像，PD算法会计算出当前**相位差**
  - 根据相位差和模组的PD calibration data，估算出像距，从而移动lens快速对焦.

    （推lens的方向和幅度，由该模组的PD calibration data和当前相位差来决定）





PD相位差 L/R图

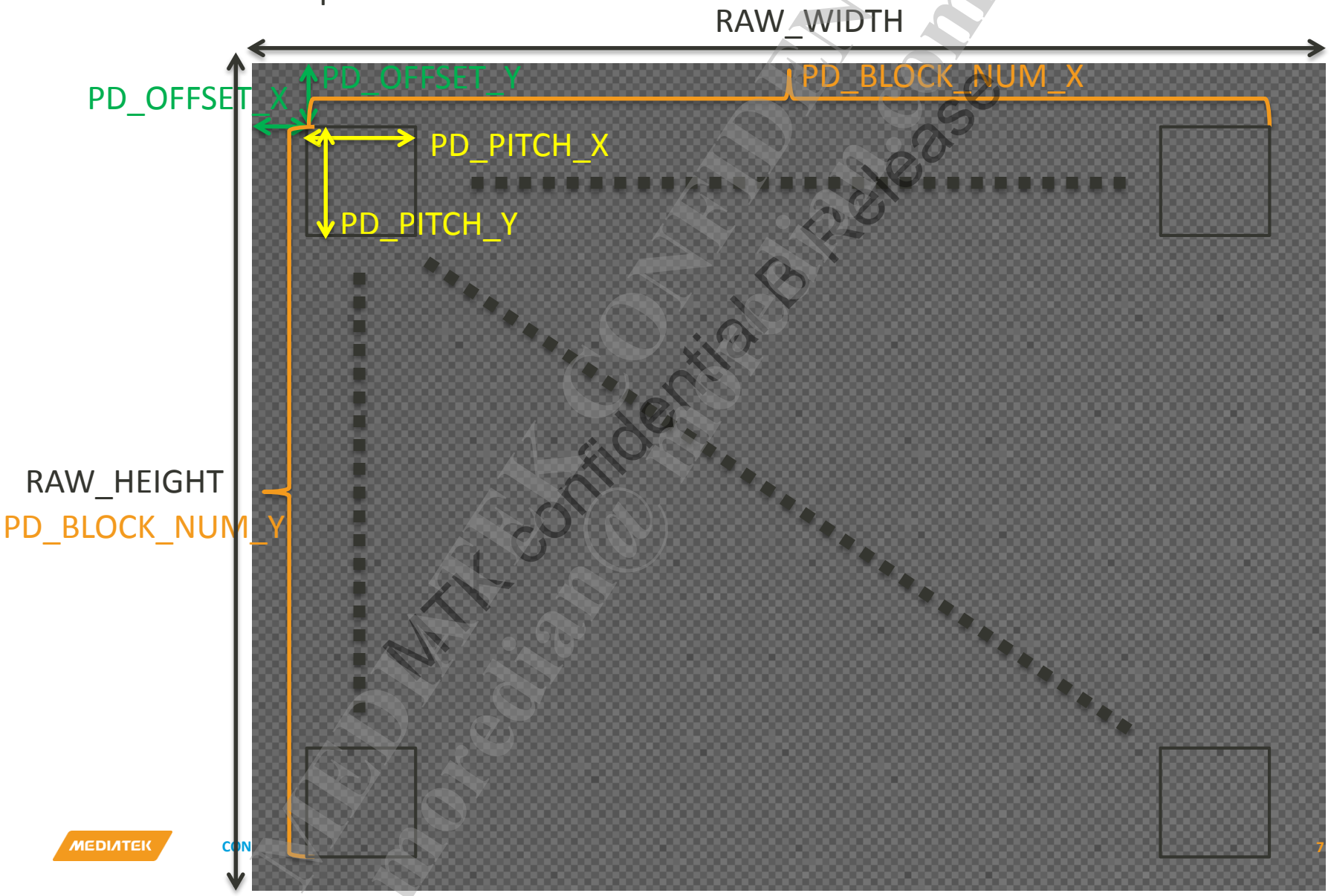
L R

通过两张图像计算出相位差，将相位差与PD calibration data做比较，来确定lens需要移动到哪个位置。

一类sensor PD点分布

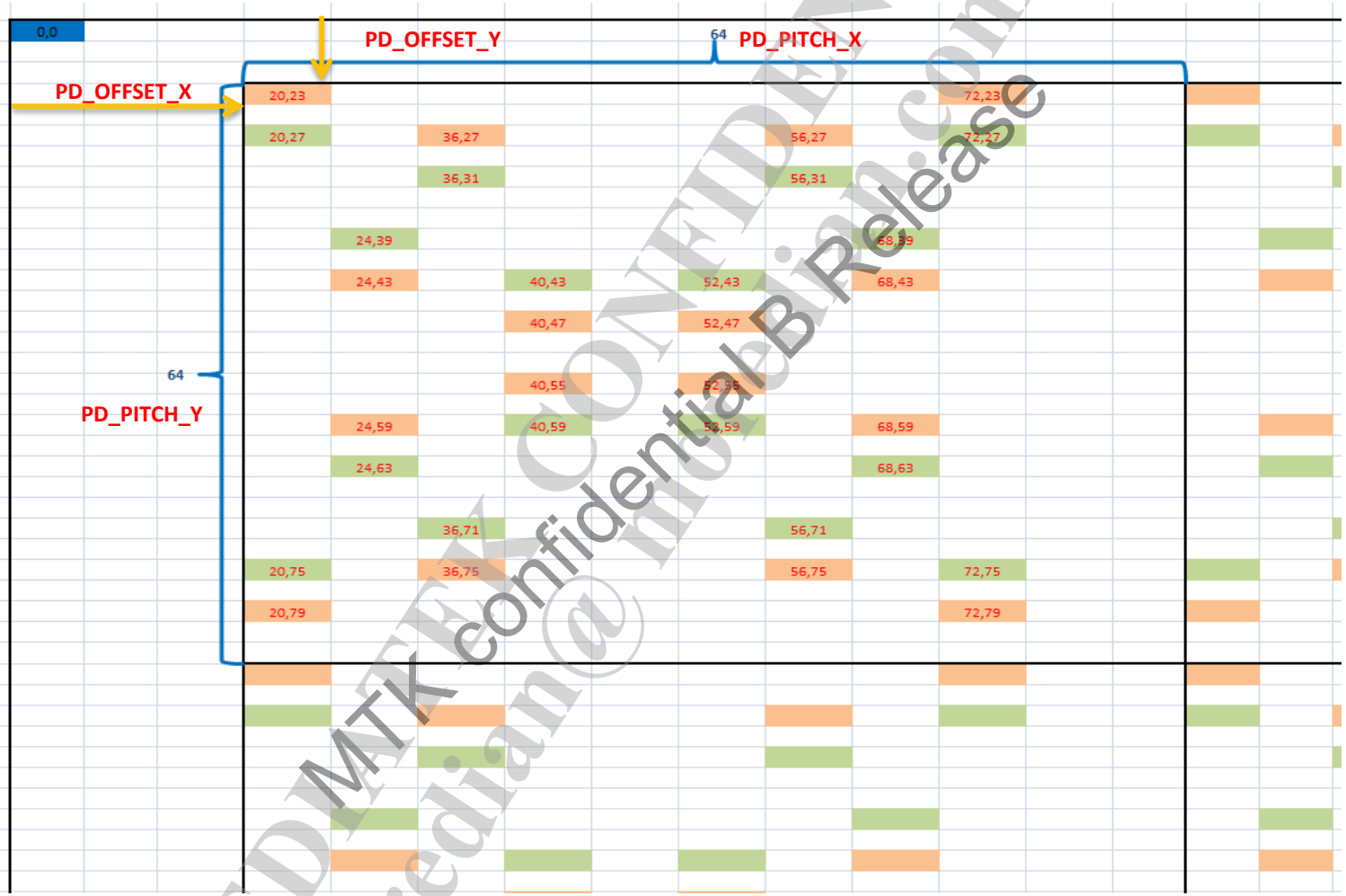
PD sensor Description - INI

Please follow sensor spec



PD sensor Description - INI

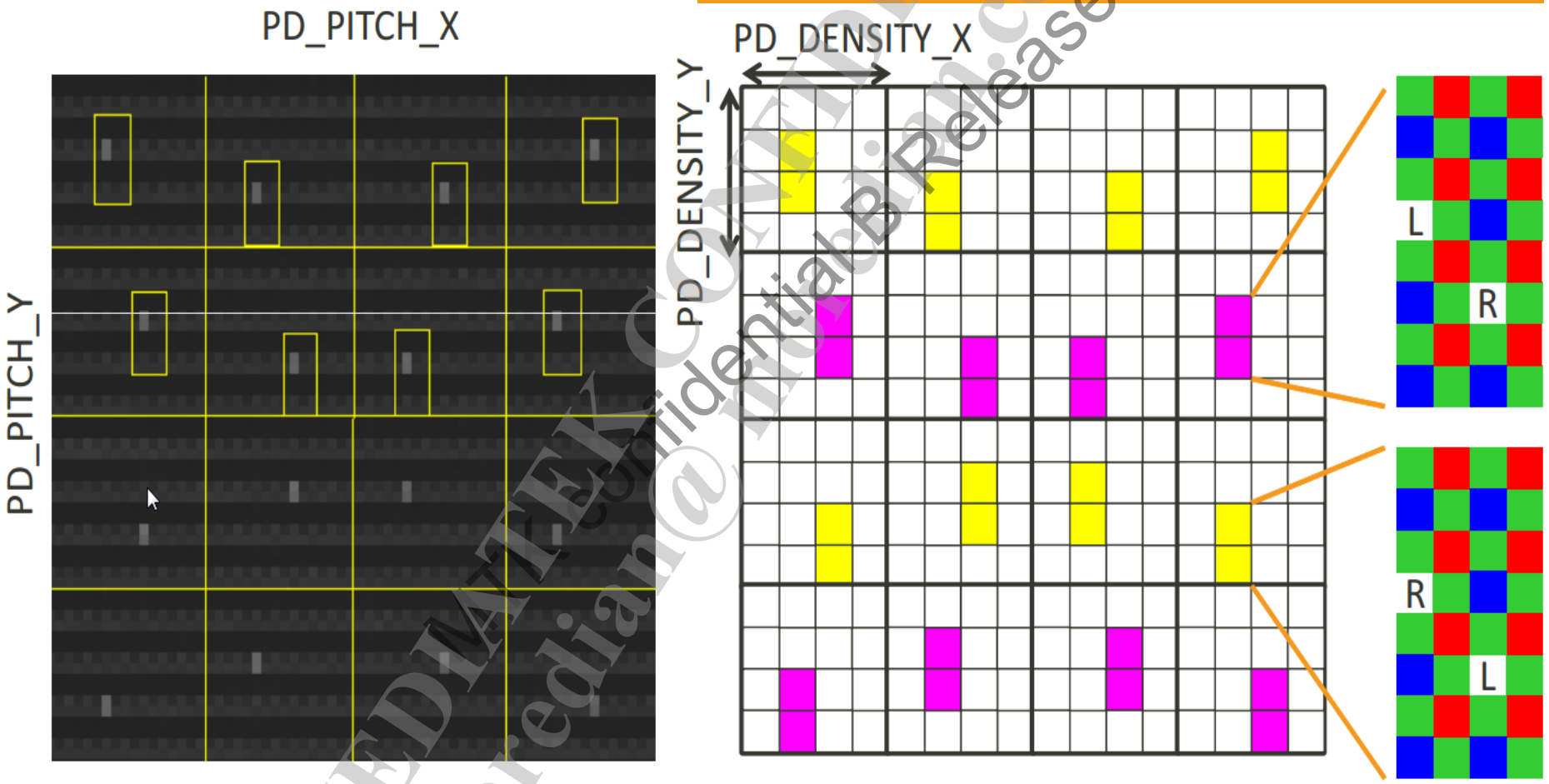
PD area offset



必须满足Offset <= PD坐标 <= Offset+Pitch-1

PD sensor Description - INI

一个Density_X*Y的sub block里面有一对PD pixel



PD INI 如何正确填写

INI档介绍：

INI文件是sensor vendor给module house做PD calibration时的配置文件，描述PD sensor的output信息及PD calibration参数，其内容会被烧录进eeprom

Driver中需要参考INI档的相关信息，INI档的内容一般如右图：其中右图53行以下calibration的相关内容不允许修改。

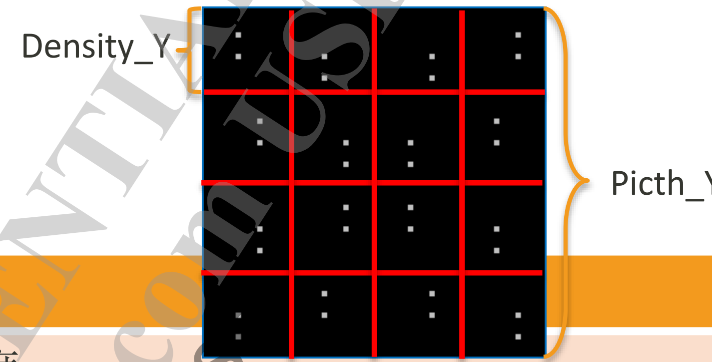
那么senor vendor应该如何根据sensor setting正确填写INI呢？

```
1   RAW_WIDTH=4192;
2   RAW_HEIGHT=3104;
3   RAW_BITS=10;
4   RAW_BYTE_ORDER=0;
5   PD_OFFSET_X=20;
6   PD_OFFSET_Y=23;
7   PD_PITCH_X=64;
8   PD_PITCH_Y=64;
9   PD_DENSITY_X=16;
10  PD_DENSITY_Y=16;
11  PD_BLOCK_NUM_X=65;
12  PD_BLOCK_NUM_Y=48;
13  CALI_PARAM1=20;
14  CALI_PARAM2=4;
15  CALI_PARAM3=8;
16  CALI_PARAM4=8;
17  CALI_PARAM5=10;
18  PD_POS_R=
19  [20 27]
20  [72 27]
21  [36 31]
22  [56 31]
23  [24 39]
24  [68 39]
25  [40 43]
26  [52 43]
27  [40 59]
28  [52 59]
29  [24 63]
30  [68 63]
31  [36 71]
32  [56 71]
33  [20 75]
34  [72 75];
35
```

```
36  PD_POS_L=
37  [20 23]
38  [72 23]
39  [36 27]
40  [56 27]
41  [24 43]
42  [68 43]
43  [40 47]
44  [52 47]
45  [40 55]
46  [52 55]
47  [24 59]
48  [68 59]
49  [36 75]
50  [56 75]
51  [20 79]
52  [72 79];
53  STEP1_VERIFY_INPUT_WIN_W=3;
54  STEP1_VERIFY_INPUT_WIN_H=3;
55  STEP1_VERIFY_INPUT_MIN=60;
56  STEP1_VERIFY_INPUT_MAX=230;
57  STEP1_VERIFY_INPUT_CORNER_MIN=40;
58  STEP1_VERIFY_INPUT_CORNER_MAX=230;
59  STEP1_VERIFY_OUTPUT_DIFF_MIN=40;
60  STEP1_VERIFY_OUTPUT_DIFF_MAX=220;
61  STEP2_VERIFY_INPUT_WIN_W=2;
62  STEP2_VERIFY_INPUT_WIN_H=2;
63  STEP2_VERIFY_INPUT_MIN=120;
64  STEP2_VERIFY_INPUT_MAX=230;
65  STEP2_VERIFY_INPUT_IMAGE_TILT_MAX=50;
```

PD INI L/R之间留一个Enter(空行)

PD INI



INI档参数介绍：

参数	含义
RAW_WIDTH	sensor输出数据的宽度
RAW_HEIGHT	sensor输出数据的高度
RAW_BITS	Pixel Value的位宽 (10: Raw10)
RAW_BYTE_ORDER	Pixel传输字节序: LSB or MSB, 目前都是(0: MSB)
RAW_BAYER_PATTERN	0: Gr 1: R 2: B 3: Gb
PD_OFFSET_X	Sensor PD Area 起始X坐标 （与PD点坐标无绝对关系）
PD_OFFSET_Y	Sensor PD Area 起始X坐标 （与PD点坐标无绝对关系）
PD_PITCH_X	PD block的宽度
PD_PITCH_Y	PD block的高度
PD_DENSITY_X	PD pixel的密度（一个X*Y的size里面有一对PD pixel）
PD_DENSITY_Y	PD pixel的密度（一个X*Y的size里面有一对PD pixel）
PD_BINNING_TYPE	默认设0；若有sensor PD pixel binning时设1 设1时再配置PD_POS_R_BINN、PD_POS_L_BINN

PD INI

INI档参数介绍：

参数	含义
PD_BLOCK_NUM_X	Width方向有多少个PD block
PD_BLOCK_NUM_Y	Height 方向有多少个PD block
CALI_PARAM1	calibration参数，保持默认值20，step1 grid sizeW
CALI_PARAM2	calibration参数，保持默认值4，step1 grid sizeH
CALI_PARAM3	calibration参数，保持默认值8，step2 grid sizeW
CALI_PARAM4	calibration参数，保持默认值8，step2 grid sizeH
CALI_PARAM5	calibration参数，保持默认值10，step2 step number
PD_POS_R	PD BLOCK0中R-Masked PD pixel的绝对坐标(Start from (0,0))
PD_POS_L	PD BLOCK0中L-Masked PD pixel的绝对坐标(Start from (0,0))
QUALITY_BAYER_PATTERN	Output bayer pattern的模式（0: Gr 1: R 2: B 3: Gb）
QUALITY_PD_L_LOCATION	PD pixel所在channel [0]B channel, [1]G channel
QUALITY_PD_R_LOCATION	PD pixel所在channel [0]B channel, [1]G channel

MEDIATEK

PDAF Bring Up

Prepare + Driver + EEPROM + PD Buffer Manager + BPCI + Para

PDAF Porting

一、准备工作

- *CAF work*　　*模组INI文档*

二、Sensor driver porting

- *XXXXmipiraw_Sensor.c*

三、读取PD calibration data

- *camera_calibration_cam_cal.cpp*
- *XXXX_pdaf_cal.c*、*XXXX_pdafotp.c*

四、PD buffer manager porting

- *pd_buf_list.cpp*
- *pd_buf_mgr.cpp*
- *pd_XXXXmipiraw.cpp*、*pd_XXXXmipiraw.h*

五、lens参数配置——PD部分

- *lens_para_XXXXAF.cpp*

一、准备工作

1. 确认Contrast AF能清晰对焦

2. 确认当前sensor的PD信息用RAW或VC传输

   VC – Virtual Chanel

   RAW – raw type (6735 6737T 6738 6739 6750 6753 6755等平台)

   PDO (6757 6763 6797 6799及之后平台)

3. 请sensor厂或模组厂提供如下文档和信息：

   (1) 模组PDAF calibration使用的**.ini文档**

   (2) 模组 pdaf calibration对应eeprom的layout文档

   (3) PDAF calibration时sensor的**安装角度**(建议正向安装: 0°)

   和**出图方向**（建议与手机实际方向设定一致, 例如**不做**MirrorFlip）

一、准备工作

4.确认sensor setting中设置的出图方向

```
/*Need Mirror/Flip or not*/
set_mirror_flip(3);
```

```
enum {
    IMAGE_NORMAL = 0,
    IMAGE_H_MIRROR,
    IMAGE_V_MIRROR,
    IMAGE_HV_MIRROR
};
```

注意：有些sensor并不调用set_mirror_flip()函数做mirrorflip，而是直接写setting

如果方案设计中需要camera sensor driver做mirror或flip，
那么模组PD calibration时也要做mirror或flip，保持一致

二、Sensor driver porting

1. 增加pd pixel的相关信息

```
static SET_PD_BLOCK_INFO_T imgsensor_pd_info =
{
    .i4OffsetX = 0,
    .i4OffsetY = 0,
    .i4PitchX  = 32,
    .i4PitchY  = 32,
    .i4PairNum  =8,
    .i4SubBlkW  =16,
    .i4SubBlkH  =8,
    .i4BlockNumX = 132,
    .i4BlockNumY = 98,
    .i4PosL = {{14, 6},{30, 6},{6, 10},{22, 10},{14, 22},{30, 22},{6, 26},{22, 26}},
    .i4PosR = {{14, 2},{30, 2},{6, 14},{22, 14},{14, 18},{30, 18},{6, 30},{22, 30}},
    .iMirrorFlip = 0,
};
```

(1) 前4个变量和L/R的坐标直接从PD INI文档中获取
(2) .i4PairNum指一个block中有几对L/R pixel
(3) .i4SubBlkW和. i4SubBlkH分别对应PD INI文档中的PD_DENSITY_X/Y
(4) .i4BlockNumX和.i4BlockNumY对应PD INI文档中的PD_BLOCK_NUM_X/Y （**必填**）
(5) .iMirrorFlip指手机sensor出图方向相对于模组厂calibration出图方向的**相对**MirrorFlip

二、 Sensor driver porting

2. get_info()函数增加PDAF的支持

```
get_info
control
```

```
/*0: NO PDAF, 1: PDAF Raw Data mode, 2:PDAF VC mode*/
sensor_info->PDAF_Support = 1;
```

As is

```
typedef enum
{
   SensorType_NO_PDAF = 0,
   SensorType_PDAF_Raw = 1,
   SensorType_PDAF_VC_HDR = 2,
   SensorType_PDAF_VC_Binning,
   SensorType_DualPD_Raw,
   SensorType_DualPD_VC
} SensorType_t;
```

To be

```
typedef enum {
        PDAF_SUPPORT_NA = 0,
        PDAF_SUPPORT_RAW = 1,
        PDAF_SUPPORT_CAMSV = 2,
        PDAF_SUPPORT_CAMSV_LEGACY = 3,
        PDAF_SUPPORT_RAW_DUALPD = 4,
        PDAF_SUPPORT_CAMSV_DUALPD = 5,
        PDAF_SUPPORT_RAW_LEGACY = 6,
} IMGSENSOR_PDAF_SUPPORT_TYPE_ENUM;
```

The 6 is used for MT6739 **RAW**-type PD ONLY

二、 Sensor driver porting

3. feature_control()函数增加对PDAF的支持

(1) case SENSOR_FEATURE_**GET_PDAF_INFO**

(2) case SENSOR_FEATURE_**GET_SENSOR_PDAF_CAPACITY**

~~(3) case SENSOR_FEATURE_GET_PDAF_DATA~~

*读取PD calibration data不再使用sensor driver API

*使用**camera_calibration_cam_cal.cpp**中**DoCamCalPDAF**

(4) case SENSOR_FEATURE_**SET_PDAF**

(5) case SENSOR_FEATURE_**GET_VC_INFO** → VC only

二、 Sensor driver porting

3. feature_control()函数增加对PDAF的支持

(1) case SENSOR_FEATURE_**GET_PDAF_INFO**

```
case SENSOR_FEATURE_GET_PDAF_INFO:
    LOG_INF("SENSOR_FEATURE_GET_PDAF_INFO scenarioId:%lld\n", *feature_data);
    PDAFinfo= (SET_PD_BLOCK_INFO_T *)(uintptr_t)(*(feature_data+1));

    switch (*feature_data) {
        case MSDK_SCENARIO_ID_CAMERA_CAPTURE_JPEG:
            memcpy((void *)PDAFinfo,(void *)&imgsensor_pd_info,sizeof(SET_PD_BLOCK_INFO_T));
            break;
        case MSDK_SCENARIO_ID_VIDEO_PREVIEW:
        case MSDK_SCENARIO_ID_HIGH_SPEED_VIDEO:
        case MSDK_SCENARIO_ID_SLIM_VIDEO:
        case MSDK_SCENARIO_ID_CAMERA_PREVIEW:
        default:
            break;
    }
    break;
```

二、 Sensor driver porting

3. feature_control()函数增加对PDAF的支持

(2) case SENSOR_FEATURE_**GET_SENSOR_PDAF_CAPACITY**

```
case SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY:
    LOG_INF("SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY scenarioId:%lld\n", *feature_data);
    //PDAF capacity enable or not, 2p8 only full size support PDAF
    switch (*feature_data) {
        case MSDK_SCENARIO_ID_CAMERA_CAPTURE_JPEG:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 1;
            break;
        case MSDK_SCENARIO_ID_VIDEO_PREVIEW:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
        case MSDK_SCENARIO_ID_HIGH_SPEED_VIDEO:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
        case MSDK_SCENARIO_ID_SLIM_VIDEO:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
        case MSDK_SCENARIO_ID_CAMERA_PREVIEW:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
        default:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
    } ? end switch *feature_data ?
    break;
```

二、 Sensor driver porting

3. feature_control()函数增加对PDAF的支持

~~(3) case SENSOR_FEATURE_GET_PDAF_DATA~~

*读取PD calibration data不再使用sensor driver API（在第三章介绍）

i. 注意：如果log中显示 NVRAM Have PDAF calib data，则不会再从eeprom读取

```
D af_mgr   : [AF][        Start] NVRAM Have PDAF calib data 2048
pd_mgr   : setPDCaliData There is PDAF calibration data in NVRAM.
```

ii. 若仍希望重新从eeprom读取calibration data，则可以：

(1) 可以format all全擦除再下载Bin

(2) 可以手动删除NVRAM

adb shell rm -rf /nvcfg/camera

adb shell rm /data/nvram/media/CAMERA_VERSION [MT6755]

(4) case SENSOR_FEATURE_SET_PDAF

```
case SENSOR_FEATURE_SET_PDAF:
    LOG_INF("PDAF mode :%d\n", *feature_data_16);
    imgsensor.pdaf_mode= *feature_data_16;
    break;
```

二、 Sensor driver porting

3. feature_control()函数增加对PDAF的支持

(5) case SENSOR_FEATURE_GET_VC_INFO

```
case SENSOR_FEATURE_GET_VC_INFO:
        LOG_INF("SENSOR_FEATURE_GET_VC_INFO %d\n", (UINT16)*feature_data);
        pvcinfo = (SENSOR_VC_INFO_STRUCT *)(uintptr_t)(*(feature_data+1));
        switch (*feature_data_32) {
        case MSDK_SCENARIO_ID_CAMERA_CAPTURE_JPEG:
            memcpy((void *)pvcinfo,(void *)&SENSOR_VC_INFO[1],sizeof(SENSOR_VC_INFO_STRUCT));
            break;
        case MSDK_SCENARIO_ID_VIDEO_PREVIEW:
            memcpy((void *)pvcinfo,(void *)&SENSOR_VC_INFO[2],sizeof(SENSOR_VC_INFO_STRUCT));
            break;
        case MSDK_SCENARIO_ID_CAMERA_PREVIEW:
        default:
            memcpy((void *)pvcinfo,(void *)&SENSOR_VC_INFO[0],sizeof(SENSOR_VC_INFO_STRUCT));
            break;
        }
        break;
```

VC only

二、 Sensor driver porting

SENSOR_VC_INFO（ Raw data mode不需要）

```
static SENSOR_VC_INFO_STRUCT SENSOR_VC_INFO[3]=
{// Preview mode setting
   {0x02, 0x0A,   0x00,   0x08, 0x40, 0x00,
   0x00, 0x2B, 0x0910, 0x06D0, 0x01, 0x00, 0x0000, 0x0000,
   0x01, 0x30, 0x00B4, 0x0360, 0x03, 0x00, 0x0000, 0x0000},
// Capture mode setting
 {0x02, 0x0A,   0x00,   0x08, 0x40, 0x00,
 0x00, 0x2B, 0x1220, 0x0DA0, 0x01, 0x00, 0x0000, 0x0000,
 0x01, 0x30, 0x00B4, 0x0360, 0x03, 0x00, 0x0000, 0x0000},
// Video mode setting
 {0x02, 0x0A,   0x00,   0x08, 0x40, 0x00,
 0x00, 0x2B, 0x1220, 0x0DA0, 0x01, 0x00, 0x0000, 0x0000,
 0x01, 0x30, 0x00B4, 0x0280, 0x03, 0x00, 0x0000, 0x0000}};
```

```
typedef struct
{
  MUINT16 VC_Num;
  MUINT16 VC_PixelNum;
  MUINT16 ModeSelect;   /* 0: HDR auto mode, 1:HDR direct mode */
  MUINT16 EXPO_Ratio;   /* 1/1, 1/2, 1/4, 1/9 */
  MUINT16 ODValue;      /* OD Vaule */
  MUINT16 RG_STATSMODE; /* STATS division mdoe 0: 16x16, 1:8x8, 2:4x4,
  MUINT16 VC0_ID;           /*VC0 : Channel ID */      → Raw data channel
  MUINT16 VC0_DataType;     /*VC0  Data type */
  MUINT16 VC0_SIZEH;        /*data size width,  unit : byte*/
  MUINT16 VC0_SIZEV;        /*data size height ,  unit : byte */
  MUINT16 VC1_ID; ;         /*VC1: Channel ID */    → HDR channel
  MUINT16 VC1_DataType; ;   /*VC1 Data type */
  MUINT16 VC1_SIZEH;
  MUINT16 VC1_SIZEV;
  MUINT16 VC2_ID;           /*VC2: Channel ID */    → PDAF channel
  MUINT16 VC2_DataType; ;   /*VC2: Data Type */
  MUINT16 VC2_SIZEH;        /*Data size per line*/
  MUINT16 VC2_SIZEV;        /*Line number of all VC_PD data*/
  MUINT16 VC3_ID;
  MUINT16 VC3_DataType;
  MUINT16 VC3_SIZEH;
  MUINT16 VC3_SIZEV;
}SENSOR_VC_INFO_STRUCT, *pSENSOR_VC_INFO_STRUCT;
```

这四个值的填法，请参考sensor datasheet
或咨询sensor vendor

VC2_SIZEH填写规则

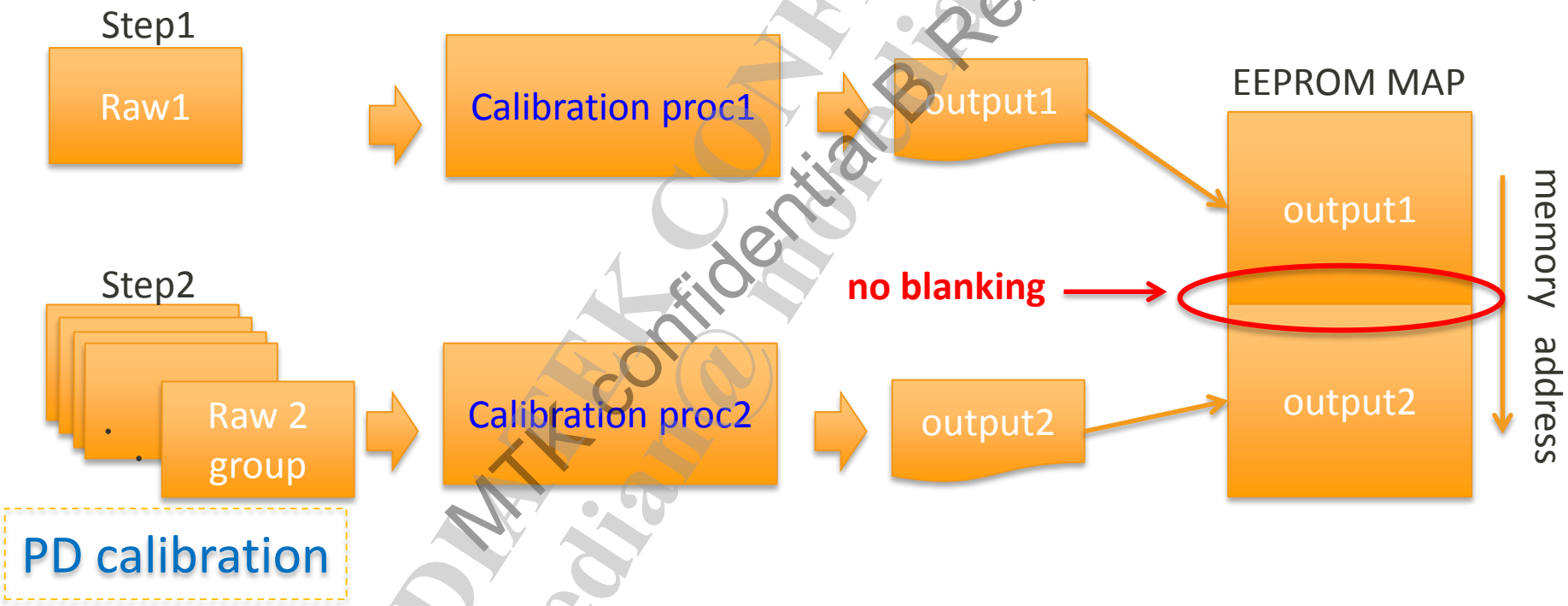
只有当VC2_DataType 等于 0x2B时：
(6735 6737T 6738 6750 6753 6755 6795等平台) 单位为**Pixel个数**
(6757 6763 6797 6799及之后平台) 单位为**Byte数**

当VC2_DataType 不等于0x2B时：VC2_SIZEH填写数字的**单位统一是Byte数**

VC only

三、读取PD calibration data

PD calibration后的数据保存在eeprom的layout示意



三、读取PD calibration data

MT6757

文件	执行函数
AfMgr::Start()	**readOTP**(CAMERA_CAM_CAL_DATA_PDAF);
AfMgr::readOTP	pCamCalDrvObj->**GetCamCalCalData**(i4SensorDevID, enCamCalEnum, (void *)&GetCamCalData);
camera_custom_msdk.cpp ::GetCameraCalData	pstSensorInitFunc[i].**getCameraCalData**(pGetSensorCalData);
sensorlist.cpp ::SensorList[]	RAW_INFO(IMX386SUNNY_SENSOR_ID, SENSOR_DRVNAME_IMX386SUNNY_MIPI_RAW, **CAM_CALGetCalData**),
camera_calibration_cam_cal.cpp ::CAM_CALGetCalData()	CalLayoutTbl[LayoutType].CalItemTbl[IsCommand].**GetCalDataProcess**() {0x00000001, 0x00000763, 0x00000800, **DoCamCalPDAF**}
camera_calibration_cam_cal.cpp ::**DoCamCalPDAF**()	ioctl(CamcamFID, **CAM_CALIOC_G_READ**, &cam_calCfg);
Cam_cal_drv.c	cam_cal_get_cmd_info_ex cam_cal_get_cmd_info cam_cal_check_mtk_cid
{$eeprom}.c	{$eeprom}_selective_read_region

三、读取PD calibration data

MT6763

文件	执行函数
PDMgr::start	**setPDCaliData**(ptrInTuningData, i4OutInfoSz, ptrOutInfo)
PDMgr::setPDCaliData	CamCalDrvBase::createInstance()->**GetCamCalCalData**(i4SensorDevID, CAMERA_CAM_CAL_DATA_PDAF, (void *)&CamCalData);
cam_cal_drv.cpp ::GetCamCalCalData	GetCameraCalData(i4CurrSensorId,(MUINT32*)pCamcalData);
camera_custom_msdk.cpp ::GetCameraCalData	pstSensorInitFunc[i].**getCameraCalData**(pGetSensorCalData);
sensorlist.cpp ::SensorList[]	RAW_INFO(IMX386SUNNY_SENSOR_ID, SENSOR_DRVNAME_IMX386SUNNY_MIPI_RAW, **CAM_CALGetCalData**),
camera_calibration_cam_cal.cpp ::CAM_CALGetCalData()	CalLayoutTbl[LayoutType].CalItemTbl[IsCommand].**GetCalDataProcess**() {0x00000001, 0x00000763, 0x00000800, **DoCamCalPDAF**}
camera_calibration_cam_cal.cpp ::**DoCamCalPDAF**()	ioctl(CamcamFID, **CAM_CALIOC_G_READ**, &cam_calCfg);
eeprom_driver.c	EEPROM_get_cmd_info_ex EEPROM_get_cmd_info
{$eeprom}.c	{$eeprom}_selective_read_region

三、读取PD calibration data

读取PD calibration data的实现需要参考vendor提供的eeprom的layout。

0x0800	PD Calibration information	Bit enable=0x03 Bit[0]: step1 Bit[1]: step2	
0x0801	PD Output-1 data	The First Byte	496 Bytes
…		…	
0x09F0		The Last Byte	
0x09F1	PD Output-2 data	The First Byte	908 Bytes
…		…	
0x0D16		…	
0x0D17		…	
…		…	
0x0D7C		The Last Byte	

我们需要把eeprom中从0x0801到0x0D7C的1404(0x57c)个byte全部读取出来

camera_calibration_cam_cal.cpp ::CAM_CALGetCalData()	CalLayoutTbl[LayoutType].CalItemTbl[IsCommand].GetCalDataProcess() {0x00000001, 0x00000801, 0x0000057c, **DoCamCalPDAF**}

三、读取PD calibration data

如果PD信息每个proc之间有一些flag byte和check sum byte,那么这些信息不要返回给上层，可自行在读取eeprom的函数里面做客制化即可。同时后一个proc的first byte要紧挨着前一个proc的最后一个byte（地址必须要连续）

```
UINT32 DoCamCalPDAF(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize,
    UINT32* pGetSensorCalData)
{
    stCAM_CAL_INFO_STRUCT  cam_calCfg;
    PCAM_CAL_DATA_STRUCT pCamCalData = (PCAM_CAL_DATA_STRUCT)pGetSensorCalData;
    MUINT32 idx;
    UINT32 ioctlerr;
    UINT32 err =  CamCalReturnErr[pCamCalData->Command];

        pCamCalData->PDAF.Size_of_PDAF = BlockSize;
        CAM_CAL_LOG_IF(dumpEnable,"PDAF start_addr =%x table_size=%d\n",start_addr, BlockSize);

        cam_calCfg.u4Offset = 0x1400;
        cam_calCfg.u4Length = 496;
        cam_calCfg.pu1Params= (u8 *)&pCamCalData->PDAF.Data[0];//PDAF.Data[0];
        cam_calCfg.sensorID = pCamCalData->sensorID;
        cam_calCfg.deviceID = pCamCalData->deviceID;
        CAM_CAL_LOG_IF(dumpEnable,"u4Offset1=%d u4Length1=%d", cam_calCfg.u4Offset, cam_calCfg.u4Length);
        ioctlerr= ioctl(CamcamFID, CAM_CALIOC_G_READ, &cam_calCfg);
        if (ioctlerr > 0)
        {
            err = CAM_CAL_ERR_NO_ERR;
            CAM_CAL_LOG_IF(dumpEnable,"Poc1 = 0x%x \n", err);
        }

        cam_calCfg.u4Offset = 0x1600;
        cam_calCfg.u4Length = 876;
        cam_calCfg.pu1Params= (u8 *)&pCamCalData->PDAF.Data[496];////PDAF.Data[496]; 496=proc1 size
        cam_calCfg.sensorID = pCamCalData->sensorID;
        cam_calCfg.deviceID = pCamCalData->deviceID;
        CAM_CAL_LOG_IF(dumpEnable,"u4Offset2=%d u4Length2=%d", cam_calCfg.u4Offset, cam_calCfg.u4Length);
        ioctlerr= ioctl(CamcamFID, CAM_CALIOC_G_READ, &cam_calCfg);
        if (ioctlerr > 0)
        {
            err = CAM_CAL_ERR_NO_ERR;
            CAM_CAL_LOG_IF(dumpEnable,"Poc2 = 0x%x \n", err);
        }
```

客制化Proc1 start_size和block_size

客制化Proc2 start_size和block_size

三、读取PD calibration data

MT6755等其他的平台则是在Sensor driver中feature_control()函数会通过case SENSOR_FEATURE_**GET_PDAF_DATA**这个分支来获取模组的PDAF calibration信息。

0x0800	PD Calibration information	Bit enable=0x03 Bit[0]: step1 Bit[1]: step2	
0x0801	PD Output-1 data	The First Byte	496 Bytes
…		…	
0x09F0		The Last Byte	
0x09F1	PD Output-2 data	The First Byte	908 Bytes
…		…	
0x0D16		…	
0x0D17		…	
…		…	
0x0D7C		The Last Byte	

```
bool read_XXX_eeprom(kal_uint16 addr, BYTE* data, kal_uint32 size){
    addr = 0x801;   //need modified
    size = 1404;    //need modified

    LOG_INF("read_XXX_eeprom, size = %d\n", size);

    if(!get_done || last_size != size || last_offset != addr) {
        if(!read_3P3_eeprom(addr, s5k3P3_eeprom_data, size)){
            get_done = 0;
            last_size = 0;
            last_offset = 0;
            return false;
        }
    }
    memcpy(data, XXxxxx_eeprom_data, size);
    return true;
}
```

三、读取PD calibration data

读取成功的log

```
D af_mgr  : [AF][        Start] NVRAM NO PDAF calib data, read from EEPROM !!
D pd_mgr  : [Core] set calibration data flow
D PdAlgo  : [parseCaliData]
D pd_mgr  : [Core] configure PD algo done 512 128
```

若**NVRAM**中无Calibration Data，则读取eeprom，并把Data存进NVRAM

```
D pd_mgr  : setPDCaliData There is PDAF calibration data in NVRAM.
D pd_mgr  : pd calibration data:
D pd_mgr  : 50 44 30 31 0f 5c e5 00 e4 01 00 00 04 00 14 00
D PdAlgo  : [parseCaliData]
D pd_mgr  : [Core] configure PD algo done 576 224
```

若**NVRAM**中有Calibration Data，则使用NVRAM中的Calibration Data

读取失败的log

```
D af_mgr : NVRAM NO PDAF calib data, read from EEPROM !!
E PdAlgo : [parseCaliData()] Err: 1489:, ParseCaliData error tag in PD01:
D pd_mgr : Load PDAF calib data error!!
D pd_mgr : [Core] PD init data error, close PDAF
D pd_mgr : close PD mgr thread
```

四、 PD buffer manager porting

1. Make sure PD data buffer type

pd_buf_common.h

\vendor\mediatek\proprietary\custom\$platform\hal\inc\aaa

2. Modify PD buffer type mapping table by using sensor ID

pd_buf_list.cpp

\vendor\mediatek\proprietary\custom\$platform\hal\pd_buf_mgr\src

3. Implement PD buffer manager for ported sensor.

pd_XXXXmipiraw.cpp *pd_XXXXmipiraw.h*

\vendor\mediatek\proprietary\custom\$platform\hal\pd_buf_mgr

四、 PD buffer manager porting

1. Make sure PD data buffer type

-pd_buf_common.h

6757 6763等新平台新增**PDO**支持RAW type PD sensor，
原EPDBuf_Raw /EPDBuf_Raw_Open 在新平台不支持

As is

```
typedef enum
{
    EPDBuf_NotDef    = 0,
    EPDBuf_VC        = 1,
    EPDBuf_VC_Open   = 2,
    EPDBuf_Raw       = 3,
    EPDBuf_Raw_Open  = 4,
    EPDBuf_PDO       = 5,
    EPDBuf_PDO_Open  = 6

} EPDBuf_Type_t;
```

To Be

```
typedef enum
{
    EPDBUF_NOTDEF            = 0x00,
    EPDBUF_VC                = 0x01,
    EPDBUF_VC_OPEN           = 0x11,
    EPDBUF_RAW_LEGACY        = 0x02,
    EPDBUF_RAW_LEGACY_OPEN   = 0x12,
    EPDBUF_PDO               = 0x04,
    EPDBUF_PDO_OPEN          = 0x14,
    EPDBUF_DUALPD_VC         = 0x21,
    EPDBUF_DUALPD_RAW        = 0x24

} EPDBUF_TYPE_t;
```

Sensor Type与Buffer Type对应关系

(6735 6737T 6738 6739 6750 6753 6755 等平台)

Name	Sensor Type	Buffer Type
S5K3M2	Raw	Raw
IMX258	VC	VC
IMX230	VC	VC_Open
OV16880	Raw	Raw
	VC	VC
IMX362/S5K2L7	DualPD_Raw/VC	DualPD_RAW/VC

SensorType
PDAF_SUPPORT_NA = 0,
PDAF_SUPPORT_RAW = 1,
PDAF_SUPPORT_CAMSV = 2,
~~PDAF_SUPPORT_CAMSV_LEGACY = 3,~~
PDAF_SUPPORT_RAW_DUALPD = 4,
PDAF_SUPPORT_CAMSV_DUALPD = 5,
PDAF_SUPPORT_RAW_LEGACY = 6,

Buffer_Type
EPDBUF_NOTDEF = 0x00,
EPDBUF_VC = 0x01,
EPDBUF_VC_OPEN = 0x11,
EPDBUF_RAW_LEGACY = 0x02,
~~EPDBUF_RAW_LEGACY_OPEN = 0x12,~~
~~EPDBUF_PDO = 0x0004,~~
~~EPDBUF_PDO_OPEN = 0x14,~~
EPDBUF_DUALPD_VC = 0x21,
EPDBUF_DUALPD_RAW = 0x22

The 6 is used for MT6739 **RAW**-type PD ONLY

Sensor Type与Buffer Type对应关系

(6757 6763 6797 6799及新一代的平台)

Name	Sensor Type	Buffer Type
S5K3M2	Raw	PDO
IMX258	VC	VC
IMX230	VC	VC_Open
OV16880	Raw	PDO
	VC	VC
IMX362/S5K2L7	DualPD_Raw/VC	DualPD_RAW/VC

SensorType
PDAF_SUPPORT_NA = 0,
PDAF_SUPPORT_RAW = 1,
PDAF_SUPPORT_CAMSV = 2,
PDAF_SUPPORT_CAMSV_LEGACY = 3,
PDAF_SUPPORT_RAW_DUALPD = 4,
PDAF_SUPPORT_CAMSV_DUALPD = 5,
PDAF_SUPPORT_RAW_LEGACY = 6,

Buffer_Type
EPDBUF_NOTDEF = 0x00,
EPDBUF_VC = 0x01,
EPDBUF_VC_OPEN = 0x11,
EPDBUF_RAW_LEGACY = 0x02,
EPDBUF_RAW_LEGACY_OPEN = 0x12,
EPDBUF_PDO = 0x0004,
EPDBUF_PDO_OPEN = 0x14,
EPDBUF_DUALPD_VC = 0x21,
EPDBUF_DUALPD_RAW = 0x22

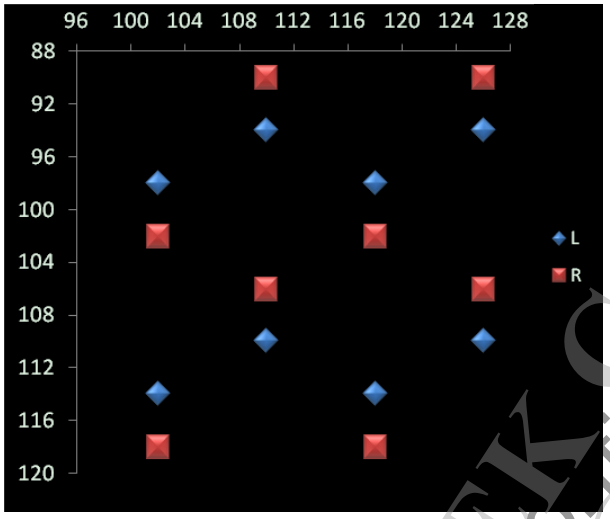
四、 PD buffer manager porting

2. Modify **PD buffer type** mapping table by using **sensor ID**

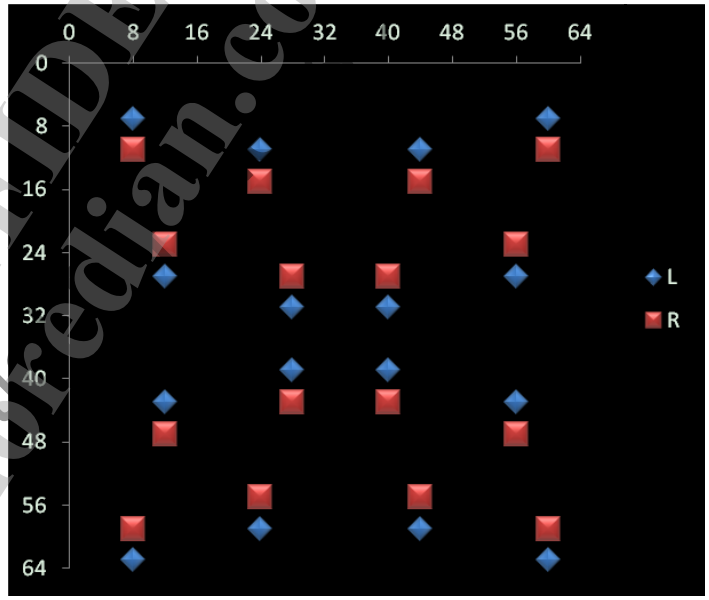
   - *pd_buf_list.cpp*

```
PDBuf_List_t PDList_main[MAX_NUM_OF_SUPPORT_SENSOR] =
{
    {XXxxxxx1_SENSOR_ID, EPDBuf_VC},
    {XXxxxxx2_SENSOR_ID, EPDBuf_PDO},
};

PDBuf_List_t PDList_sub[MAX_NUM_OF_SUPPORT_SENSOR] =
{
    {{XXxxxxx1_SENSOR_ID, EPDBuf_VC},
    {{XXxxxxx3_SENSOR_ID, EPDBuf_Raw},
};
```

四、 PD buffer manager porting



ov sensor PD pattern
democode参考ov16880



Samsung sensor PD pattern
democode参考s5k3m3(VC) s5k2p8(PDO)

根据L/R的坐标可以画出sensor的PD pattern，不同厂家pattern可能不同
PDO Porting的时候要根据不同的pattern参考不同的democode

四、 PD buffer manager porting

3. Implement PD buffer manager for ported sensor.

- *pd_buf_mgr.cpp* *pd_XXXXmipiraw.cpp* *pd_XXXXmipiraw.h*

Algorithm	Buffer Type	Buffer mgr
MTK	VC - ov16880 s5k3m3 imx258 PDO - ov16880 s5k2p8 s5k2x8 Raw - s5k3m2	pd_buf_mgr.cpp
3rd party	VC - imx338 imx386 imx230	pd_buf_mgr_open.cpp

pd_XXXXmipiraw.cpp和***pd_XXXXmipiraw.h***中，不同的PD buffer type需要继承不同的C++基类以及需要实现不同的函数，建议按照同类型的其他sensor的头文件进行拷贝修改。

四、 PD buffer manager porting

3. Implement PD buffer manager for ported sensor.

pd_buf_mgr.cpp
pd_XXXXmipiraw.cpp
pd_XXXXmipiraw.h

MTK PD algorithm

- Add get instance function and include header file in pd_buf_mgr.cpp

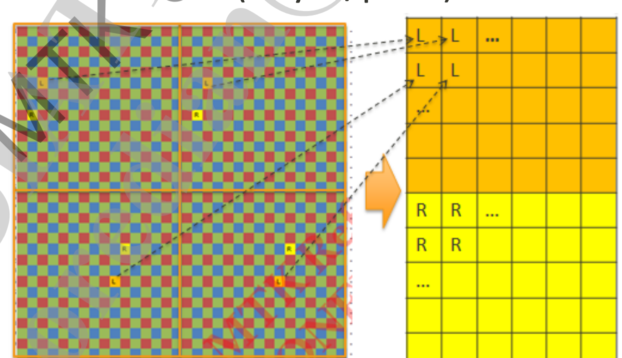
3rd party PD algorithm

- in pd_buf_mgr_open.cpp

```
#include <pd_buf_mgr.h>
#include <pd_ov23850mipiraw.h>
#include <pd_s5k2p8mipiraw.h>
#include <pd_imx258mipiraw.h>
#include <pd_s5k3m2mipiraw.h>
#include <pd_s5k2x8mipiraw.h>
#include <pd_ov16880mipiraw.h>

PDBufMgr::PDBufMgr()
{
    memset( &m_PDBlockInfo, 0, sizeof(SET_PD_BLOCK_INFO_T));
}

PDBufMgr::~PDBufMgr()
{
}

PDBufMgr*
PDBufMgr::createInstance(SPDProfile_t &iPdProfile)
{
    PDBufMgr *instance = NULL;
    PDBufMgr *ret      = NULL;

    switch( iPdProfile.i4CurrSensorId)
    {
#if defined(OV23850_MIPI_RAW)
    case OV23850_SENSOR_ID :
        instance = PD_OV23850MIPIRAW::getInstance();
        break;
#endif
#if defined(OV16880_MIPI_RAW)
    case OV16880_SENSOR_ID :
        instance = PD_OV16880MIPIRAW::getInstance();
        break;
#endif
#if defined(S5K2P8_MIPI_RAW)
    case S5K2P8_SENSOR_ID :
        instance = PD_S5K2P8MIPIRAW::getInstance();
        break;
```

四、 PD buffer manager porting
-- *pd_XXXXmipiraw.cpp*

- **MTK PD algorithm.**
  - 必须by sensor porting以下代码：
    - 移植**IsSupport()**函数
      - 支持PDAF的条件，如full size或1/4 size mode等
    - 正确设定以下**变量**：
      - **(1)m_PDXSz (2)m_PDYSz (3) m_PDBufSz**
    - 移植**ConvertPDBufFormat() (seprate)**函数：
      - PD点统一转换为RAW16 (2byte/pixel)，并重新排列buffer: L靠前R靠后



ConvertPDBufFormat()函数作用

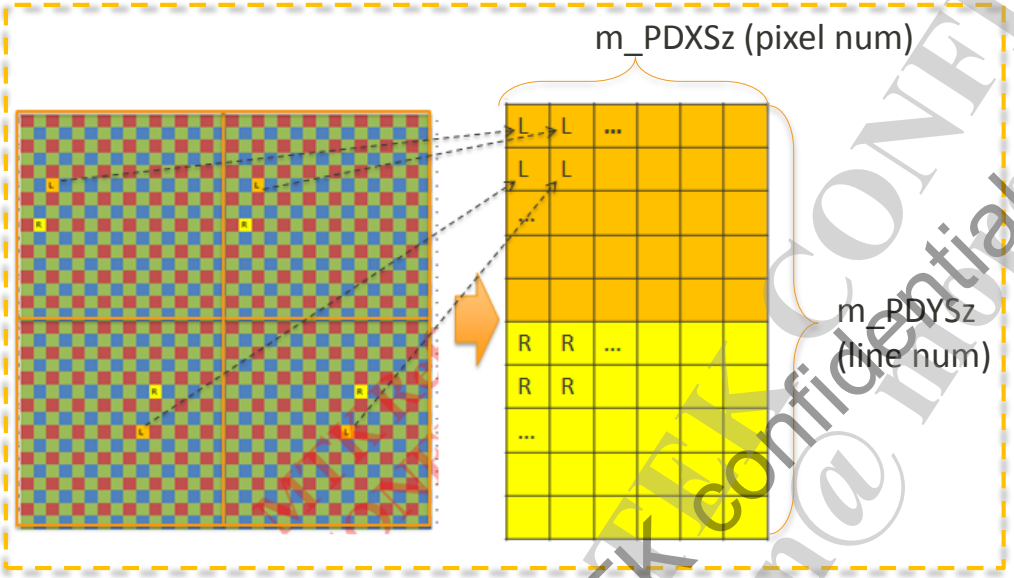
四、 PD buffer manager porting
-- *pd_XXXXmipiraw.cpp*

- **IsSupport()** –支持PDAF的条件

```
MBOOL PD_OV13855MIPIRAW::IsSupport( SPDProfile_t &iPdProfile)
{
    MBOOL ret = MFALSE;

    if(iPdProfile.uImgYsz == 3136)  //4224x3136 4:3 full mode
    {
        if( m_PDBuf) { delete m_PDBuf;m_PDBufSz = 0; m_PDBuf = NULL;}

        m_PDXSz    = 264;
        m_PDYSz    = 784;
        m_PDBufSz = m_PDXSz*m_PDYSz;
        m_PDBuf = new MUINT16 [m_PDBufSz];

        ret = MTRUE;
        AAA_LOGD("PDAF Mode is Supported with (%d, %d)\n", iPdProfile.uImgXsz, iPdProfile.uImgYsz);
    }
    else if(iPdProfile.uImgYsz == 2376) //4224x2376 16:9 mode
    {
        if( m_PDBuf) { delete m_PDBuf;m_PDBufSz = 0; m_PDBuf = NULL;}

        m_PDXSz    = 264;
        m_PDYSz    = 592;
        m_PDBufSz = m_PDXSz*m_PDYSz;
        m_PDBuf = new MUINT16 [m_PDBufSz];

        ret = MTRUE;
        AAA_LOGD("PDAF Mode is Supported with (%d, %d)\n", iPdProfile.uImgXsz, iPdProfile.uImgYsz);
    }
    else
    {
        AAA_LOGD("PDAF Mode is not Supported (%d, %d)\n", iPdProfile.uImgXsz, iPdProfile.uImgYsz);
    }
    m_eBufType = iPdProfile.BufType;
    return ret;
} ? end IsSupport ?
```

四、 PD buffer manager porting
-- *pd_XXXXmipiraw.cpp*

- 设定(1)m_PDXSz (2)m_PDYSz (3)m_PDBufSz




```
RAW_WIDTH=4224;
RAW_HEIGHT=3136;
RAW_BITS=10;
RAW_BYTE_ORDER=0;
PD_OFFSET_X=0;
PD_OFFSET_Y=0;
PD_PITCH_X=32;
PD_PITCH_Y=32;
PD_DENSITY_X=16;
PD_DENSITY_Y=8;
PD_BLOCK_NUM_X=132;
PD_BLOCK_NUM_Y=98;
```

```
PD_POS_R=
[14 2]
[30 2]
[6 14]
[22 14]
[14 18]
[30 18]
[6 30]
[22 30];
```

```
PD_POS_L=
[14 6]
[30 6]
[6 10]
[22 10]
[14 22]
[30 22]
[6 26]
[22 26];
```

(注意INI L/R之间要有空行)

Example

By sensor 根据PD点的Position和Data spec分析计算。上面Example INI表明L/R PD pixel都不在同一行，是一种较为简单的传输模式，其相应size计算为：

每一行传送pixel num = PitchX / DensityX * BlockNumX = 32/16*132 = 264 (m_PDXSz)
传送的行数 line num = PitchY / DensityY * 2 * BlockNumY = 32/8*2*98 = 784 (m_PDYSz)

```
m_PDXSz    = 264;
m_PDYSz    = 784;
m_PDBufSz  = m_PDXSz*m_PDYSz;
m_PDBuf = new MUINT16 [m_PDBufSz];
```

四、 PD buffer manager porting
-- *pd_XXXXmipiraw.cpp*

- ConvertPDBufFormat() – PD点统一转换为RAW16 (2byte/pixel)，并重新排列buffer: L靠前R靠后

Buffer Type	Reference Code
VC	pd_imx258mipiraw.cpp pd_ov16880mipiraw.cpp pd_s5k3m3mipiraw.cpp
PDO	pd_ov16880mipiraw.cpp pd_s5k2p8mipiraw.cpp
Raw	ConvertPDBufFormat() 直接return NULL (6735 6737T 6738 6739 6750 6753 6755 6795等平台)

四、 PD buffer manager porting
-- *pd_XXXXmipiraw.cpp*

- **MTK PD algorithm - PDO.**

PDO需要额外生成**BPCI table**，并将ISP中的BPC_EN和PDC_EN置1，实现获取和补偿PD点

(1) 参考文档 <***PDAF BPCI_table Generation_Guide***>，
用**CCT tool**和**PD INI档**，生成BPCI table (注意INI L/R之间要有空行)

(2) 生成正确的BPCI table后，参考不同平台的BPCI version分别进行porting

(3) 将ISP中的BPC_EN和PDC_EN置1

四、 PD buffer manager porting
-- *pd_XXXXmipiraw.cpp*

- **MTK PD algorithm - PDO.**

BPCI Version	Platform	Characteristic
BPCI 1.0	MT6797	BPCI table include in *pd_{$sensor}mipiraw.cpp*
BPCI 2.0	MT6757	BPCI table include in *camera_info_{$sensor}mipiraw.h*
BPCI 3.0	MT6763	Multi BPCI table include in *camera_info_{$sensor}mipiraw.h*

PD buffer manager Porting guide – BPCI table 1.0 (1)

- The BPCI table format is shown as below :

  Bpci_tbl_[sensor name].h

```
const unsigned int bpci_xsize_s5k2x8=10239;
const unsigned int bpci_ysize_s5k2x8=0;
const unsigned int pdo_xsize_s5k2x8=687;
const unsigned int pdo_ysize_s5k2x8=767;
const unsigned char bpci_array_s5k2x8[]={
0x41,0xC0,0x40,0x00,0x75,0x15,0x01,0x00,0x34,0x00,0x0C,0x00,0x45,0xC0,0x40,0x00,
0x75,0x15,0x03,0x00,0x10,0x40,0x14,0x00,0x40,0x00,0x0C,0x40,0x49,0xC0,0x50,0x00,
0x55,0x15,0x01,0x00,0x14,0x40,0x2C,0x40,0x61,0xC0,0x44,0x00,0x6D,0x15,0x01,0x00,
0x2C,0x40,0x14,0x40,0x55,0xC0,0x44,0x00,0x6D,0x15,0x03,0x00,0x10,0x00,0x0C,0x40,
0x10,0x40,0x14,0x00,0x59,0xC0,0x54,0x00,0x4D,0x15,0x01,0x00,0x0C,0x00,0x34,0x00,
0x61,0xC0,0x54,0x00,0x4D,0x15,0x01,0x00,0x0C,0x00,0x34,0x00,0x65,0xC0,0x44,0x00,
0x6D,0x15,0x03,0x00,0x10,0x00,0x0C,0x40,0x10,0x40,0x14,0x00,0x69,0xC0,0x44,0x00,
0x6D,0x15,0x01,0x00,0x2C,0x40,0x14,0x40,0x71,0xC0,0x50,0x00,0x55,0x15,0x01,0x00,
0x14,0x40,0x2C,0x40,0x75,0xC0,0x40,0x00,0x75,0x15,0x03,0x00,0x10,0x40,0x14,0x00,
0x10,0x00,0x0C,0x40,0x79,0xC0,0x40,0x00,0x75,0x15,0x01,0x00,0x34,0x00,0x0C,0x00,
```

- Add **bpci table (.h)**

  vendor\mediatek\proprietary\custom\mt6797\hal\pd_buf_mgr\[Sensor name]\ **Bpci_tbl_[sensor name].h**

PD buffer manager Porting guide – BPCI table 1.0 (2)

- BPCI table should be ported during using ***PDBuf_PDO***.
  - After BPCI table is generated, the table should be included in
    - pd_buf_mgr. (**pd_[sensor name]mipiraw.cpp**)

```
#include <utils/Log.h>
#include <fcntl.h>
#include <math.h>

#include <pd_s5k2x8mipiraw.h>
#include <aaa_log.h>
#include <cutils/properties.h>
#include <stdlib.h>
#include "bpci_tbl_s5k2x8.h"

#define LOG_TAG "pd_buf_mgr_s5k2x8mipiraw"

PDBufMgr*
PD_S5K2X8MIPIRAW::getInstance()
{
    static PD_S5K2X8MIPIRAW singleton;
    return &singleton;
}
```

  - ***virtual MBOOL GetPDOHWInfo(MINT32 i4CurSensorMode, SPDOHWINFO_T &oPDOhwInfo);*** should be inherited from **PDBufMgr**. Assign values in ***Bpci_tbl_[sensor name].h*** to struct SPDOHWINFO_t directly.

```
MBOOL PD_S5K2X8MIPIRAW::GetPDOHWInfo( MINT32 i4CurSensorMode, SPDOHWINFO_T &oPDOhwInfo)
{
    oPDOhwInfo.u4Bpci_xsz  = bpci_xsize_s5k2x8;
    oPDOhwInfo.u4Bpci_ysz  = bpci_ysize_s5k2x8;
    oPDOhwInfo.pu1Bpci_tbl = bpci_array_s5k2x8;
    oPDOhwInfo.u4Pdo_xsz   = pdo_xsize_s5k2x8;
    oPDOhwInfo.u4Pdo_ysz   = pdo_ysize_s5k2x8;

    return MTRUE;
}
```

PD buffer manager Porting guide – BPCI table 2.0 (1)

- The BPCI table format is shown as below :

  Camera_bpci_tbl_[sensor name].h

```
const unsigned int bpci_xsize=9983;
const unsigned int bpci_ysize=0;
const unsigned int pdo_xsize=2239;
const unsigned int pdo_ysize=415;
const unsigned char bpci_array[]={
0x49,0xC0,0x5A,0x00,0x7A,0x11,0x07,0x00,0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,
0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,0x51,0xC0,0x5E,0x00,0x7A,0x11,0x07,0x00,
0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,
0x59,0xC0,0x5A,0x00,0x7A,0x11,0x07,0x00,0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,
0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,0x61,0xC0,0x5E,0x00,0x7A,0x11,0x07,0x00,
0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,
0x69,0xC0,0x5A,0x00,0x7A,0x11,0x07,0x00,0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,
0x01,0x00,0x07,0x40,0x01,0x00,0x07,0x40,0x71,0xC0,0x5E,0x00,0x7A,0x11,0x07,0x00
```

- Add **bpci table (.h)**

  vendor\mediatek\proprietary\custom\[platform]\hal\imgsensor\[sensor name]\
  Camera_bpci_tbl_[sensor name].h

PD buffer manager Porting guide – BPCI table 2.0 (2)

- BPCI table should be ported during using ***PDBuf_PDO***.
  - After BPCI table is generated, the table should be included in
    - **custom\[platform]\hal\imgsensor\[sensor name]\camera_info_[sensor name].h**

```
#define SENSOR_ID                       IMX398_SENSOR_ID
#define SENSOR_DRVNAME                  SENSOR_DRVNAME_IMX398_MIPI_RAW
#define INCLUDE_FILENAME_ISP_REGS_PARAM "camera_isp_regs_imx398mipiraw.h"
#define INCLUDE_FILENAME_ISP_PCA_PARAM  "camera_isp_pca_imx398mipiraw.h"
#define INCLUDE_FILENAME_ISP_LSC_PARAM  "camera_isp_lsc_imx398mipiraw.h"
#define INCLUDE_FILENAME_TSF_PARA       "camera_tsf_para_imx398mipiraw.h"
#define INCLUDE_FILENAME_TSF_DATA       "camera_tsf_data_imx398mipiraw.h"
#define INCLUDE_FILENAME_FLASH_AWB_PARA "camera_flash_awb_para_imx398mipiraw.h"
#define INCLUDE_FILENAME_FEATURE_PARA   "camera_feature_para_imx398mipiraw.h"
#define INCLUDE_FILENAME_BPCI_PARA      "camera_bpci_tbl_imx398mipiraw.h"
```

  - **INCLUDE_FILENAME_BPCI_PARA** should be include in

    camera_tuning_para_ [Sensor Name]mipiraw.cpp

  - BPC_En, PDC_En read from tuning data should be enabled (PDO)

```
    .con    ={.bits={.BPC_EN=1, .rsv_1=0, .BPC_LUT_EN=0, .BPC_TABLE_END_MODE=0,
    .con    ={.bits={.PDC_EN=1, .rsv_1=0, .PDC_CT=1, .rsv_5=0, .PDC_MODE=1,
```

camera_isp_regs_capture_{$sensor}mipiraw.h
camera_isp_regs_video_{$sensor}mipiraw.h
camera_isp_regs_preview_{$sensor}mipiraw.h

PD buffer manager Porting guide – BPCI table 2.0 (3)

- BPCI table should be ported during using ***PDBuf_PDO***.
  - After BPCI table is generated, the table should be included in
    - **/[platform]/hal/imgsensor/[sensor name]/camera_tuning_para_[sensor name].cpp**

```
#define NVRAM_TUNING_PARAM_NUM  6181001
#include INCLUDE_FILENAME_BPCI_PARA
```

```
const CAMERA_BPCI_STRUCT CAMERA_PCI_DEFAULT_VALUE =
{
    .bpci_xsize = (UINT32)bpci_xsize,
    .bpci_ysize = (UINT32)bpci_ysize,
    .pdo_xsize = (UINT32)pdo_xsize,
    .pdo_ysize = (UINT32)pdo_ysize,
    .bpci_array = (UINT8*)bpci_array
};

const NVRAM_CAMERA_FEATURE_STRUCT CAMERA_FEATURE_DEFAULT_VALUE =
{
#include INCLUDE_FILENAME_FEATURE_PARA
};
```

```
UINT32 dataSize[CAMERA_DATA_TYPE_NUM] = {sizeof(NV
    sizeof(NVRAM_CAMERA_3A_STRUCT),
    sizeof(NVRAM_CAMERA_SHADING_STRUCT),
    sizeof(NVRAM_LENS_PARA_STRUCT),
    sizeof(AE_PLINETABLE_T),
    0,
    sizeof(CAMERA_TSF_TBL_STRUCT),
    sizeof(CAMERA_BPCI_STRUCT),
    0,
    sizeof(NVRAM_CAMERA_FEATURE_STRUCT)
};
```

```
case CAMERA_DATA_TSF_TABLE:
  memcpy(pDataBuf,&CAMERA_TSF_DEFAULT_VALUE,sizeof(CAMERA_TSF_TBL_STRUCT));
  break;
case CAMERA_DATA_PDC_TABLE:
  memcpy(pDataBuf,&CAMERA_PCI_DEFAULT_VALUE, sizeof(CAMERA_BPCI_STRUCT));
  break;
case CAMERA_NVRAM_DATA_FEATURE:
```

PD buffer manager Porting guide – BPCI table 3.0 (1)

- The **Multi** BPCI table format is shown as below :

Camera_bpci_tbl_[sensor name].h

```
#define PDC_TBL_1_XSIZE (7359)
#define PDC_TBL_1_YSIZE (0)
#define PDC_PDO_1_XSIZE (663)
#define PDC_PDO_1_YSIZE (551)
static const MUINT8 PDC_1_array[(PDC_TBL_1_XSIZE+1)]={
0x17,0xC0,0x00,0x00,0xB5,0x14,0x01,0x00,0x34,0x00,0x0C,0x00,0x1B,0xC0,0x00,0x00,
```

Pass1

```
#define PDC_TBL_2_XSIZE (7775)
#define PDC_TBL_2_YSIZE (0)
#define PDC_PDO_2_XSIZE (2239)
#define PDC_PDO_2_YSIZE (323)
static const MUINT8 PDC_2_array[(PDC_TBL_2_XSIZE+1)]={
0x0D,0xC0,0x5A,0x00,0x7A,0x11,0x07,0x00,0x01,0x40,0x07,0x00,0x01,0x40,0x07,0x00,
```

Pass1

```
#define PDC_TBL_3_XSIZE (7359)
#define PDC_TBL_3_YSIZE (0)
#define PDC_PDO_3_XSIZE (663)
#define PDC_PDO_3_YSIZE (551)
static const MUINT8 PDC_3_array[(PDC_TBL_3_XSIZE+1)]={
0x17,0xC0,0x00,0x00,0xB5,0x14,0x01,0x00,0x34,0x00,0x0C,0x00,0x1B,0xC0,0x00,0x00,
```

Pass2

```
const CAMERA_BPCI_STRUCT CAMERA_PCI_DEFAULT_VALUE =
{
    .PDC_TBL_1 = {
        .bpci_xsize = (UINT32)PDC_TBL_1_XSIZE,
        .bpci_ysize = (UINT32)PDC_TBL_1_YSIZE,
        .pdo_xsize = (UINT32)PDC_PDO_1_XSIZE,
        .pdo_ysize = (UINT32)PDC_PDO_1_YSIZE,
        .bpci_array = (MUINT8*)PDC_1_array
    },

    .PDC_TBL_2 = {
        .bpci_xsize = (UINT32)PDC_TBL_2_XSIZE,
        .bpci_ysize = (UINT32)PDC_TBL_2_YSIZE,
        .pdo_xsize = (UINT32)PDC_PDO_2_XSIZE,
        .pdo_ysize = (UINT32)PDC_PDO_2_YSIZE,
        .bpci_array = (MUINT8*)PDC_2_array
    },

    .PDC_TBL_3 = {
        .bpci_xsize = (UINT32)PDC_TBL_3_XSIZE,
        .bpci_ysize = (UINT32)PDC_TBL_3_YSIZE,
        .pdo_xsize = (UINT32)PDC_PDO_3_XSIZE,
        .pdo_ysize = (UINT32)PDC_PDO_3_YSIZE,
        .bpci_array = (MUINT8*)PDC_3_array
    }
};
```

- Add **bpci table (.h)**

vendor\mediatek\proprietary\custom\[platform]\hal\imgsensor\[sensor name]\
Camera_bpci_tbl_[sensor name].h

PD buffer manager Porting guide – BPCI table 3.0 (2)

PD supporting type (link)	Table index	Sensor mode	Frontal binning	PD buffer manager	Note
PDAF_SUPPORT_RAW	1	SENSOR_SCENARIO_ID_NORMAL_CAPTURE	Off	EPDBUF_PDO	For pass1. Mapping rule should be modified inside both isp_mgr_bnr and pd_mgr.
	2	SENSOR_SCENARIO_ID_NORMAL_VIDEO	Off	EPDBUF_PDO	
	3	SENSOR_SCENARIO_ID_NORMAL_CAPTURE	Off	NA	For pass2. it is used for capturing image

RAW-type PD sensor

Table1	Capture **full size** BPCI table
Table2	Video size or ¼ size BPCI table (若video or preivew不支持PDAF可不修改此table，若支持则需用新的PD INI生成新的table)
Table3	Pass2使用，请填写跟Table1相同的Capture full size BPCI table

pd_mgr.cpp

```
if( PDAF_support_type == PDAF_SUPPORT_RAW)
{
    EIndex_PDC_TBL_T tbl_idx = eIDX_PDC_NUM;

    if( m_profile.i4SensorMode==SENSOR_SCENARIO_ID_NORMAL_CAPTURE)
        tbl_idx = eIDX_PDC_1;
    else if( m_profile.i4SensorMode==SENSOR_SCENARIO_ID_NORMAL_VIDEO)
        tbl_idx = eIDX_PDC_2;
    else
        tbl_idx = eIDX_PDC_NUM;
```

用于获取PD点做PDAF

isp_mgr_bnr.cpp

```
switch( m_u4PDSensorFmt)
{
    case PDAF_SUPPORT_RAW:
        if( !rRawIspCamInfo.BinInfo.fgBIN)
        {
            if( rRawIspCamInfo.eSensorMode==SENSOR_SCENARIO_ID_NORMAL_CAPTURE)
                ePDC_table_type = eIDX_PDC_1;
            else if( rRawIspCamInfo.eSensorMode==SENSOR_SCENARIO_ID_NORMAL_VIDEO)
                ePDC_table_type = eIDX_PDC_2;
            else
                ePDC_table_type = eIDX_PDC_NUM;
        }
        else
            ePDC_table_type = eIDX_PDC_NUM;
        break;
```

用于补偿PD点提升图像品质

PD buffer manager Porting guide – BPCI table 3.0 (3)

- BPCI table should be ported during using ***PDBuf_PDO***.
  - After BPCI table is generated, the table should be included in
    - **custom\[platform]\hal\imgsensor\[sensor name]\camera_info_[sensor name].h**

```
#define SENSOR_ID                       IMX398_SENSOR_ID
#define SENSOR_DRVNAME                  SENSOR_DRVNAME_IMX398_MIPI_RAW
#define INCLUDE_FILENAME_ISP_REGS_PARAM "camera_isp_regs_imx398mipiraw.h"
#define INCLUDE_FILENAME_ISP_PCA_PARAM  "camera_isp_pca_imx398mipiraw.h"
#define INCLUDE_FILENAME_ISP_LSC_PARAM  "camera_isp_lsc_imx398mipiraw.h"
#define INCLUDE_FILENAME_TSF_PARA       "camera_tsf_para_imx398mipiraw.h"
#define INCLUDE_FILENAME_TSF_DATA       "camera_tsf_data_imx398mipiraw.h"
#define INCLUDE_FILENAME_FLASH_AWB_PARA "camera_flash_awb_para_imx398mipiraw.h"
#define INCLUDE_FILENAME_FEATURE_PARA   "camera_feature_para_imx398mipiraw.h"
#define INCLUDE_FILENAME_BPCI_PARA      "camera_bpci_tbl_imx398mipiraw.h"
```

  - **INCLUDE_FILENAME_BPCI_PARA** should be include in

    camera_tuning_para_ [Sensor Name]mipiraw.cpp

```
#define NVRAM_TUNING_PARAM_NUM  6181001
#include INCLUDE_FILENAME_BPCI_PARA
```

PD buffer manager Porting guide – BPCI table 3.0 (4)

- BPCI table should be ported during using *PDBuf_PDO*.

  Modify impGetDefaultData() in *camera_tuning_para_[Sensor Name]mipiraw.cpp*

```
impGetDefaultData(CAMERA_DATA_TYPE_ENUM const CameraDataType, VOID*const pDataBuf, UINT32 const size) const
{
  UINT32 dataSize[CAMERA_DATA_TYPE_NUM] = {sizeof(NVRAM_CAMERA_ISP_PARAM_STRUCT),
      sizeof(NVRAM_CAMERA_3A_STRUCT),
      sizeof(NVRAM_CAMERA_SHADING_STRUCT),
      sizeof(NVRAM_LENS_PARA_STRUCT),
      sizeof(AE_PLINETABLE_T),
      0,
      sizeof(CAMERA_TSF_TBL_STRUCT),
      sizeof(CAMERA_BPCI_STRUCT),
      0,
      sizeof(NVRAM_CAMERA_FEATURE_STRUCT)
  };
```

```
    switch(CameraDataType)
    {
      case CAMERA_NVRAM_DATA_ISP:
        memcpy(pDataBuf,&CAMERA_ISP_DEFAULT_VALUE,sizeof(NVRAM_CAMERA_ISP_PARAM_STRUCT));
        break;
      case CAMERA_NVRAM_DATA_3A:
        memcpy(pDataBuf,&CAMERA_3A_NVRAM_DEFAULT_VALUE,sizeof(NVRAM_CAMERA_3A_STRUCT));
        break;
      case CAMERA_NVRAM_DATA_SHADING:
        memcpy(pDataBuf,&CAMERA_SHADING_DEFAULT_VALUE,sizeof(NVRAM_CAMERA_SHADING_STRUCT));
        break;
      case CAMERA_DATA_AE_PLINETABLE:
        memcpy(pDataBuf,&g_PlineTableMapping,sizeof(AE_PLINETABLE_T));
        break;
      case CAMERA_DATA_TSF_TABLE:
        memcpy(pDataBuf,&CAMERA_TSF_DEFAULT_VALUE,sizeof(CAMERA_TSF_TBL_STRUCT));
        break;
      case CAMERA_DATA_PDC_TABLE:
        memcpy(pDataBuf,&CAMERA_PCI_DEFAULT_VALUE, sizeof(CAMERA_BPCI_STRUCT));
        break;
```

PD buffer manager Porting guide – BPCI table 3.0 (5)

- BPCI table should be ported during using ***PDBuf_PDO***.
  - BPC_En, PDC_En read from tuning data should be enabled in all PDAF scene

camera_isp_regs_capture_{$sensor}mipiraw.h
camera_isp_regs_video_{$sensor}mipiraw.h
camera_isp_regs_preview_{$sensor}mipiraw.h

```
.con     ={.bits={.BPC_EN=1, .rsv_1=0, .BPC_LUT_EN=0, .BPC_TABLE_END_MODE=0,
.con     ={.bits={.PDC_EN=1, .rsv_1=0, .PDC_CT=1, .rsv_5=0, .PDC_MODE=1,
```

五、lens参数配置 - PD部分

lens_para_XXxxxxAF.cpp 新AF参数版本

```
//i4PDAFCoefs[64]

//=============================================================/
//[0] name: pd_default_param
//     range: 0 or 1
//     default: 1
//     constraints: none
//     effect: set 0 will use default value pre-define in algorithm, the following
//             parameter will be invalid.
//-------------------------------------------------------------/
1, //[0] pd_default_param

//=============================================================/
//[1] name: pd_enable
//     range: 0 or 1
//     default: 1
//     constraints: none
//     effect: set 1 to enable PD source, set 0 OFF
//-------------------------------------------------------------/
1, //[1] pd_enable
```

enabe

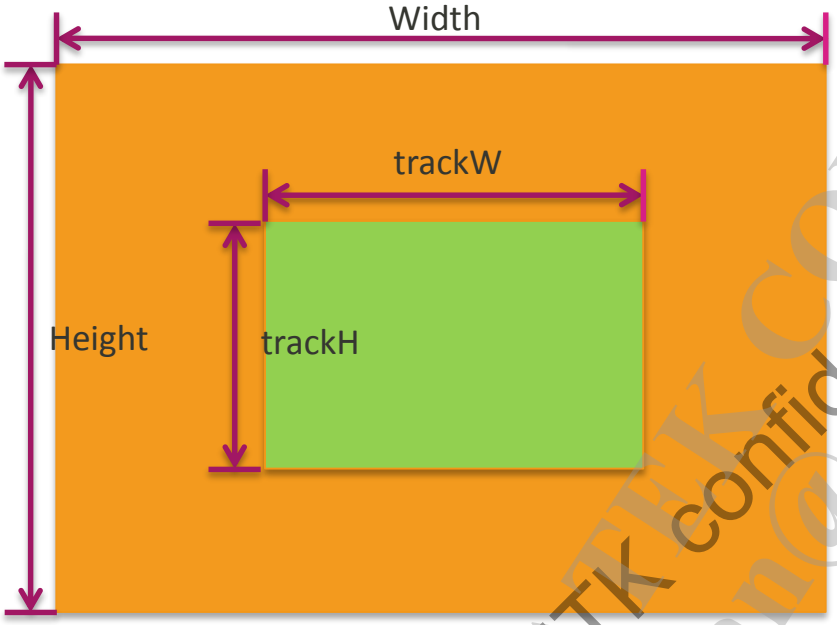
```
//PD_ALGO_TUNING_T
64,
64,
{10, 30, 50, 70, 100},
{384,410,435,461,486},
{
    {0,0,0,0,0,0},
    {0,0,0,0,0,0},
    {0,20,20,20,20,20},
    {0,20,60,60,60,60},
    {0,20,60,60,60,60},
    {0,20,60,60,60,100}
},
230,      //i4SaturateLevel
8,        //i4SaturateThr
10,       //i4ConfThr
{0},
```

Check使用的参数是否合理：

1. 保证参数非全0；
2. PDAF enable
3. 红框中的参数合理

五、lens参数配置 - PD部分

```
//PD_ALGO_TUNING_T
64,     //pdnumW
64,     //pdnumH
```




```
//=============================================================/
// Section: Hybrid AF window config
// Description: this section control AF ROI and how many sub PD block use in the ROI.
//
//[1] name: tracking_width
//[2] name: tracking_height
//     range: 1 ~ 100 (%)
//     default: [1]37  [2]39
//     constraints:
//     effect: set how many percent image width and height to be hybrid af ROI at
//             continue mode.
//[3] name: max_pd_win_x
//[4] name: max_pd_win_y
//     range: 1~3
//     default: [3]3 [4]3
//     effect: maximal number of sub PD block use in the tracking_width and
//             tracking_height. default using 3x3 window.
//-------------------------------------------------------------/
 37, //[1] tracking_width
 39, //[2] tracking_height
  3, //[3] max_pd_win_x
  3, //[4] max_pd_win_y
```

lens_para_XXxxxxAF.cpp

```
static SET_PD_BLOCK_INFO_T imgsensor_pd_info =

{

    .i4OffsetX = 64,
    .i4OffsetY = 65,
    .i4PitchX  = 64,
    .i4PitchY  = 64,
    .i4PairNum  =16,
    .i4SubBlkW  =16,
    .i4SubBlkH  =16,
    .i4PosL = {{64,65},{116,65},{80,69},{100,69},{68
    .i4PosR = {{64,69},{116,69},{80,73},{100,73},{68

};
```

pdnumW <= Width * trackW(%) / MaxPdWinX / DensityW
pdnumH <= Height * trackH(%) / MaxPdWinY / DensityH

pdnumW和pdnumH都填4的整数倍

DensityW = 16
DensityH = 16

数据来源于sensor driver 或 INI档

五、lens参数配置 - PD部分

lens_para_XXxxxxAF.cpp

PDAF sub window 长和宽上的PD pixel数量

```
{//PD_ALGO_TUNING_T
    32,
    24,
    {30, 100,200,300,400},
    {384,410,435,461,486},
    {{0,0,0,0,0,0},
    {0,0,0,0,0,0},
    {0,20,20,20,20,20},
    {0,20,60,60,60,60},
    {0,20,60,60,60,60},
    {0,20,60,60,60,100}},
    230,        //  i4SaturateLevel
    8,          //i4SaturateThr
    10,         //i4ConfThr
```

PDAF sub window中亮度>230 的pd pixel 个数，经数量转换若大于i4SaturateThr， Confidence = 0

值越小，confidence越大，设为default值不需要调

MEDIATEK

PDAF Bring Up

PDAF Porting Example of Case

MTK PDAF Porting Case - Task

任务分类	任务细项
Prepare	确认sensor PDAF类型(RAW/VC)，确认手机上sensor **Mirror/Flip**设定方向 确认模组厂PD Calibration时是否有**Mirror/Flip**，拿到正确的PD INI 明确PD spec (根据sensor PD datasheet，明确**PD点传送方式和数据量**) 获得模组PD Calibration对应eeprom的layout文档 确认Contrast AF对焦正常
Kernel Driver	设置pd_info信息 设置PDAF_Support类型 设置feature_control()函数::case SENSOR_FEATURE_GET_PDAF_INFO 设置feature_control()函数::case SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY 设置feature_control()函数::case SENSOR_FEATURE_SET_PDAF 设置feature_control()函数::case SENSOR_FEATURE_GET_VC_INFO 设置VC_info信息 **VC ONLY**
Load Calibration Data	读取Calibration data并由平台解析成功
PD Buffer Manager	确认当前sensor type对应的Buffer Type，并在pd_buf_list中添加匹配关系 移植相同类型sensor的demo code (根据Buffer Type和PD Pixel排列确定相同类型) 设置**pd_{$sensor}mipiraw.cpp**中的**IsSupport()** 和 **ConvertPDBufFormat()** 合入**BPCI** table **PDO(RAW) ONLY**
Lens Parameter	Enable PDAF；正确设置PD 相关参数；

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
Prepare	1. 确认sensor PDAF类型(RAW/VC)，确认手机上sensor **Mirror/Flip**设定方向 2. 确认模组厂PD Calibration时是否有**Mirror/Flip**，拿到正确的PD INI 3. 明确PD spec (根据sensor PD datasheet，明确PD点传送方式和数据量) 4. 获得模组PD Calibration对应eeprom的layout文档 5. 确认Contrast AF对焦正常

1. (1)确认sensor PDAF类型(RAW/VC)

OV13855 sensor type：**RAW**

-> 参考PD sensor Reference Manual、Datasheet 或 咨询sensor vendor

1. (2)确认手机上sensor **Mirror/Flip**设定方向

table 4-1 mirror and flip registers

address	register name	default value	R/W	description
0x3820	FORMAT1	0xA0	RW	Timing Control Register Bit[5]: Black line vertical flip control 0: Black line never flips 1: Black line flips with normal image Bit[4]: Image vertical flip enable 0: Normal 1: Vertical flip Bit[3]: Image horizontal mirror disable 0: Image horizontal mirror enable 1: Normal

ov13855 datasheet - mirror/flip

```
00864:      write_cmos_sensor(0x3820, 0xa8);
```

Sensor driver 中写0x3820寄存器 0xa8 = 10101000
Bit[4]=0 Bit[3]=1 => Normal+Normal

手机上sensor 设定为**Normal - Mirror OFF Flip OFF**

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
Prepare	1. 确认sensor PDAF类型(RAW/VC)，确认手机上sensor **Mirror/Flip**设定方向 2. 确认模组厂PD Calibration时是否有**Mirror/Flip**，拿到正确的PD INI 3. 明确PD spec (根据sensor PD datasheet，明确PD点**传送方式和数据量**) 4. 获得模组PD Calibration对应eeprom的layout文档 5. 确认Contrast AF对焦正常

2. 确认模组厂PD Calibration时是否有**Mirror/Flip**，拿到正确的PD INI

```
RAW_WIDTH=4224;
RAW_HEIGHT=3136;
RAW_BITS=10;
RAW_BYTE_ORDER=0;
RAW_BAYER_PATTERN=3;
PD_OFFSET_X=0;
PD_OFFSET_Y=0;
PD_PITCH_X=32;
PD_PITCH_Y=32;
PD_DENSITY_X=16;
PD_DENSITY_Y=8;
PD_BLOCK_NUM_X=132;
PD_BLOCK_NUM_Y=98;
PD_BINNING_TYPE=0;
```

```
PD_POS_R=
[14 2]
[30 2]
[6 14]
[22 14]
[14 18]
[30 18]
[6 30]
[22 30];

PD_POS_L=
[14 6]
[30 6]
[6 10]
[22 10]
[14 22]
[30 22]
[6 26]
[22 26];
```

(注意INI L/R之间要有空行)

本模组calibration设定
Mirror Normal Flip Normal
sensor设定
Mirror OFF Flip OFF
两者是**一致的**

OV13855_4224x3136_mirror_normal_flip_normal_mtk3.0.0.ini

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

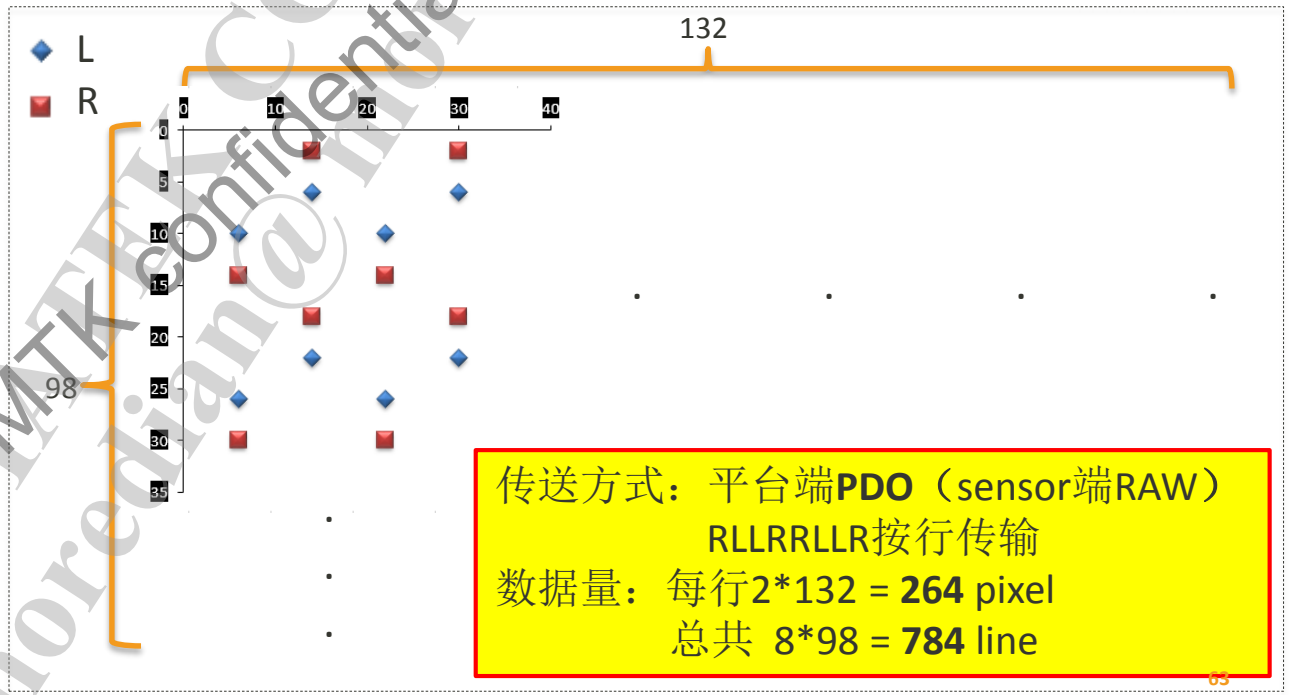
任务分类	任务细项
Prepare	1. 确认sensor PDAF类型(RAW/VC)，确认手机上sensor **Mirror/Flip**设定方向 2. 确认模组厂PD Calibration时是否有**Mirror/Flip**，拿到正确的PD INI 3. 明确PD spec (根据sensor PD datasheet，明确PD点**传送方式和数据量**) 4. 获得模组PD Calibration对应eeprom的layout文档 5. 确认Contrast AF对焦正常

3. 明确PD spec (根据sensor PD datasheet，明确PD点**传送方式和数据量**)

```
RAW_WIDTH=4224;
RAW_HEIGHT=3136;
RAW_BITS=10;
RAW_BYTE_ORDER=0;
RAW_BAYER_PATTERN=3;
PD_OFFSET_X=0;
PD_OFFSET_Y=0;
PD_PITCH_X=32;
PD_PITCH_Y=32;
PD_DENSITY_X=16;
PD_DENSITY_Y=8;
PD_BLOCK_NUM_X=132;
PD_BLOCK_NUM_Y=98;
PD_BINNING_TYPE=0;
```

(注意INI L/R之间要有空行)

```
PD_POS_R=
[14 2]
[30 2]
[6 14]
[22 14]
[14 18]
[30 18]
[6 30]
[22 30];

PD_POS_L=
[14 6]
[30 6]
[6 10]
[22 10]
[14 22]
[30 22]
[6 26]
[22 26];
```



传送方式：平台端**PDO**（sensor端RAW）
RLLRRLLR按行传输
数据量：每行2*132 = **264** pixel
总共 8*98 = **784** line

MTK PDAF Porting Case : **PDO**-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
Prepare	1. 确认sensor PDAF类型(RAW/VC)，确认手机上sensor **Mirror/Flip**设定方向 2. 确认模组厂PD Calibration时是否有**Mirror/Flip**，拿到正确的PD INI 3. 明确PD spec (根据sensor PD datasheet，明确PD点传送方式和数据量) 4. 获得模组PD Calibration对应eeprom的layout文档 5. 确认Contrast AF对焦正常

4. 获得模组PD Calibration对应eeprom的layout文档

MTK PDAF Proc1			
0x1400	Proc1	The first byte of Proc1	496
......			
0x15ef		The 496th byte of Proc1	
0x15f0	Flag of Proc1	0x01: Valid 其他：数据为空或无效	
0x15f1	Checksum of Proc1	CHECKSUM	
0x15f2		Reserved	Default value:0xff
......			
0x15ff		Reserved	Default value:0xff
MTK PDAF Proc2			
0x1600	Proc2	The first byte of Proc2	876
......			
0x196b		The 876th byte of Proc2	
0x196c	Flag of Proc2	0x01: Valid 其他：数据为空或无效	
0x196d	Checksum of Proc2	CHECKSUM	
0x196e		Reserved	Default value:0xff
......			
0x1a67		Reserved	Default value:0xff

eeprom layout of PD Calibration
Proc1: 0x1400~0x15EF, 496 bytes
Proc2: 0x1600~0x196B, 876 bytes

5. 确认Contrast AF对焦正常

没有PDAF只用ContrastAF时，TouchAF能准确合焦即可

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
Kernel Driver	1. 设置pd_info信息； 2. 设置PDAF_Support类型 3. 设置feature_control()函数::case SENSOR_FEATURE_GET_PDAF_INFO 4. 设置feature_control()函数::case SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY 5. 设置feature_control()函数::case SENSOR_FEATURE_SET_PDAF 6. 设置feature_control()函数::case SENSOR_FEATURE_GET_VC_INFO 7. 设置VC_info信息 VC ONLY

1. 设置pd_info信息； 2. 设置PDAF_Support类型

```
static SET_PD_BLOCK_INFO_T imgsensor_pd_info =
{
     .i4OffsetX = 0,
     .i4OffsetY = 0,
     .i4PitchX  = 32,
     .i4PitchY  = 32,
     .i4PairNum  =8,
     .i4SubBlkW  =16,
     .i4SubBlkH  =8,
     .i4BlockNumX = 132,
     .i4BlockNumY = 98,
     .i4PosL = {{14, 6},{30, 6},{6, 10},{22, 10},{14, 22},{30, 22},{6, 26},{22, 26}},
     .i4PosR = {{14, 2},{30, 2},{6, 14},{22, 14},{14, 18},{30, 18},{6, 30},{22, 30}},
     .iMirrorFlip = 0,
};
```

2. 设置PDAF_Support类型

```
/* 0: NO PDAF, 1: PDAF Raw data PDO mode, 2:PDAF VC mode, 3:PDAF VC Legacy mode,
*  4: PDAF DualPD Raw Data mode, 5: PDAF DualPD VC mode */

sensor_info->PDAF_Support = PDAF_SUPPORT_RAW;    //PDAF_SUPPORT_RAW = 1
```

```
RAW_WIDTH=4224;
RAW_HEIGHT=3136;
RAW_BITS=10;
RAW_BYTE_ORDER=0;
RAW_BAYER_PATTERN=3;
PD_OFFSET_X=0;
PD_OFFSET_Y=0;
PD_PITCH_X=32;
PD_PITCH_Y=32;
PD_DENSITY_X=16;
PD_DENSITY_Y=8;
PD_BLOCK_NUM_X=132;
PD_BLOCK_NUM_Y=98;
PD_BINNING_TYPE=0;
```

```
PD_POS_R=
[14 2]
[30 2]
[6 14]
[22 14]
[14 18]
[30 18]
[6 30]
[22 30];

PD_POS_L=
[14 6]
[30 6]
[6 10]
[22 10]
[14 22]
[30 22]
[6 26]
[22 26];
```

(注意INI L/R之间要有空行)

1. pd_info根据正确的INI信息填写到driver
2. 设置PDAF_Support=1 //PDAF_SUPPORT_RAW

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
Kernel Driver	1. 设置pd_info信息; 2. 设置PDAF_Support类型 3. 设置feature_control()函数::case SENSOR_FEATURE_GET_PDAF_INFO 4. 设置feature_control()函数::case SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY 5. 设置feature_control()函数::case SENSOR_FEATURE_SET_PDAF 6. 设置feature_control()函数::case SENSOR_FEATURE_GET_VC_INFO 7. 设置VC_info信息 VC ONLY

3. 设置feature_control()函数::case SENSOR_FEATURE_GET_PDAF_INFO
4. 设置feature_control()函数::case SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY

```
case SENSOR_FEATURE_GET_PDAF_INFO:
    LOG_INF("SENSOR_FEATURE_GET_PDAF_INFO scenarioId:%lld\n", *feature_data);
    PDAFinfo= (SET_PD_BLOCK_INFO_T *)(uintptr_t)(*(feature_data+1));

    switch (*feature_data) {
        case MSDK_SCENARIO_ID_CAMERA_CAPTURE_JPEG:
            memcpy((void *)PDAFinfo,(void *)&imgsensor_pd_info,sizeof(SET_PD_BLOCK_INFO_T));
            break;
        case MSDK_SCENARIO_ID_VIDEO_PREVIEW:
        case MSDK_SCENARIO_ID_HIGH_SPEED_VIDEO:
        case MSDK_SCENARIO_ID_SLIM_VIDEO:
        case MSDK_SCENARIO_ID_CAMERA_PREVIEW:
        default:
            break;
    }
    break;
```

GET_PDAF_INFO

3. 根据scenario正确设置pd_info的memory copy
4. 根据scenario正确设置支持PD 的sensor mode

```
case SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY:
    LOG_INF("SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY scenarioId:%lld\n", *feature_data);
    //PDAF capacity enable or not, 2p8 only full size support PDAF
    switch (*feature_data) {
        case MSDK_SCENARIO_ID_CAMERA_CAPTURE_JPEG:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 1;
            break;
        case MSDK_SCENARIO_ID_VIDEO_PREVIEW:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
        case MSDK_SCENARIO_ID_HIGH_SPEED_VIDEO:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
        case MSDK_SCENARIO_ID_SLIM_VIDEO:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
        case MSDK_SCENARIO_ID_CAMERA_PREVIEW:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
        default:
            *(MUINT32 *)(uintptr_t)(*(feature_data+1)) = 0;
            break;
    } ? end switch *feature_data ?
    break;
```

GET_SENSOR_PDAF_CAPACITY

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
Kernel Driver	1. 设置pd_info信息； 2. 设置PDAF_Support类型 3. 设置feature_control()函数::case SENSOR_FEATURE_GET_PDAF_INFO 4. 设置feature_control()函数::case SENSOR_FEATURE_GET_SENSOR_PDAF_CAPACITY 5. 设置feature_control()函数::case SENSOR_FEATURE_SET_PDAF 6. 设置feature_control()函数::case SENSOR_FEATURE_GET_VC_INFO 7. 设置VC_info信息 VC ONLY

5. 设置feature_control()函数::case SENSOR_FEATURE_SET_PDAF

```
case SENSOR_FEATURE_SET_PDAF:
    LOG_INF("PDAF mode :%d\n", imgsensor.pdaf_mode);
    break;
```

SET_PDAF

5. ov13855未使用imgsensor.pdaf_mode这个flag来控制写其他的寄存器
6、7 PDO(RAW) PDAF不需要配置

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
Load Calibration Data	读取Calibration data并由平台解析成功

```
UINT32 DoCamCalPDAF(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize,
    UINT32* pGetSensorCalData)
{
    stCAM_CAL_INFO_STRUCT  cam_calCfg;
    PCAM_CAL_DATA_STRUCT pCamCalData = (PCAM_CAL_DATA_STRUCT)pGetSensorCalData;
    MUINT32 idx;
    UINT32 ioctlerr;
    UINT32 err =  CamCalReturnErr[pCamCalData->Command];

        pCamCalData->PDAF.Size_of_PDAF = BlockSize;
        CAM_CAL_LOG_IF(dumpEnable,"PDAF start_addr =%x table_size=%d\n",start_addr, BlockSize);

        cam_calCfg.u4Offset = 0x1400;
        cam_calCfg.u4Length = 496;
        cam_calCfg.pu1Params= (u8 *)&pCamCalData->PDAF.Data[0];//PDAF.Data[0];
        cam_calCfg.sensorID = pCamCalData->sensorID;
        cam_calCfg.deviceID = pCamCalData->deviceID;
        CAM_CAL_LOG_IF(dumpEnable,"u4Offset1=%d u4Length1=%d", cam_calCfg.u4Offset, cam_calCfg.u4Length);
        ioctlerr= ioctl(CamcamFID, CAM_CALIOC_G_READ, &cam_calCfg);
        if (ioctlerr > 0)
        {
            err = CAM_CAL_ERR_NO_ERR;
            CAM_CAL_LOG_IF(dumpEnable,"Poc1 = 0x%x \n", err);
        }

        cam_calCfg.u4Offset = 0x1600;
        cam_calCfg.u4Length = 876;
        cam_calCfg.pu1Params= (u8 *)&pCamCalData->PDAF.Data[496];////PDAF.Data[496]; 496=proc1 size
        cam_calCfg.sensorID = pCamCalData->sensorID;
        cam_calCfg.deviceID = pCamCalData->deviceID;
        CAM_CAL_LOG_IF(dumpEnable,"u4Offset2=%d u4Length2=%d", cam_calCfg.u4Offset, cam_calCfg.u4Length);
        ioctlerr= ioctl(CamcamFID, CAM_CALIOC_G_READ, &cam_calCfg);
        if (ioctlerr > 0)
        {
            err = CAM_CAL_ERR_NO_ERR;
            CAM_CAL_LOG_IF(dumpEnable,"Poc2 = 0x%x \n", err);
        }
```

客制化Proc**1** start_size和block_size

客制化Proc**2** start_size和block_size

Load calibration data from eeprom

eeprom **layout** of PD Calibration

Proc1: 0x1400~0x15EF,   496 bytes
Proc2: 0x1600~0x196B, 876 bytes

平台解析calibration data成功**log**

PdAlgo : [parseCaliData] PD01 version = 15031311, size = 484

PdAlgo : [parseCaliData] PD02 version = 15031311, size = 794

PdAlgo : [parseCaliData] PD03 version = 15031311, size = 58

pd_mgr : [Core] configure PD algo done 576 224

根据layout实现DoCamCamPDAF
并通过log确认读取和解析都正确

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
PD Buffer Manager	1. 确认当前sensor type对应的Buffer Type，并在pd_buf_list中添加匹配关系 2. 移植相同类型sensor的demo code (根据Buffer Type和PD Pixel排列确定相同类型) 3. 设置**pd_{$sensor}mipiraw.cpp**中的**IsSupport()** 和 **ConvertPDBufFormat()** 4. 合入**BPCI** table **PDO(RAW) ONLY**

1. 确认当前sensor type对应的Buffer Type，并在pd_buf_list中添加匹配关系

```
PDBuf_List_t PDList_main[MAX_SIZE_OF_PD_SENSOR_LIST] =
{
#if defined(OV13855_MIPI_RAW)
    {OV13855_SENSOR_ID, EPDBUF_PDO},
#endif
```

OV13855 是 RAW-type PDAF
需设置为**PDO** buffer type

2. 移植相同类型sensor的demo code (根据Buffer Type和PD Pixel排列确定相同类型)

(1) 移植 pd_ov16880mipiraw.cpp 、 pd_ov16880mipiraw.h
=> pd_ov13855mipiraw.cpp 、 pd_ov13855mipiraw.h

(2) Add ov13855 in *pd_buf_mgr.cpp*

```
#include <pd_ov13855mipiraw.h>
```

```
PDBufMgr::createInstance(SPDProfile_t &iPdProfile)
{
    PDBufMgr *instance = NULL;
    PDBufMgr *ret      = NULL;

    switch( iPdProfile.i4CurrSensorId)
    {
#if defined(OV13855_MIPI_RAW)
    case OV13855_SENSOR_ID :
        instance = PD_OV13855MIPIRAW::getInstance();
        break;
#endif
```

OV13855 参考 OV16880 PDO部分

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
PD Buffer Manager	1. 确认当前sensor type对应的Buffer Type，并在pd_buf_list中添加匹配关系 2. 移植相同类型sensor的demo code (根据Buffer Type和PD Pixel排列确定相同类型) 3. 设置**pd_{$sensor}mipiraw.cpp**中的**IsSupport()** 和 **ConvertPDBufFormat()** 4. 合入**BPCI** table **PDO(RAW) ONLY**

3. (1)设置**pd_ov13855mipiraw.cpp**中的**IsSupport()**

```
MBOOL PD_OV13855MIPIRAW::IsSupport( SPDProfile_t &iPdProfile)
{
    MBOOL ret = MFALSE;

    //PDAF is supported in Full size mode. (4224x3136)
    if(iPdProfile.uImgYsz == 3136 || iPdProfile.isZSD)
    {
        if( m_PDBuf) {
            delete m_PDBuf;
            m_PDBufSz = 0;
            m_PDBuf = NULL;
        }

        m_PDXSz    = 264;
        m_PDYSz    = 784;
        m_PDBufSz = m_PDXSz*m_PDYSz;
        m_PDBuf = new MUINT16 [m_PDBufSz];

        ret = MTRUE;
        AAA_LOGD("PDAF Mode is Supported with (%d, %d)\n", iPdProfile.uImgXsz, iPdProfile.uImgYsz);
    }
```

支持PDAF的条件，如ZSD full size mode

删除前一次m_PDBuf

重新排列后PD buffer的大小，X size为每行pixel数，Y size为buffer行数

重新申请m_PDBuf，供后续排列PD buffer使用

填写PDAF支持条件
设置PD buffer的size

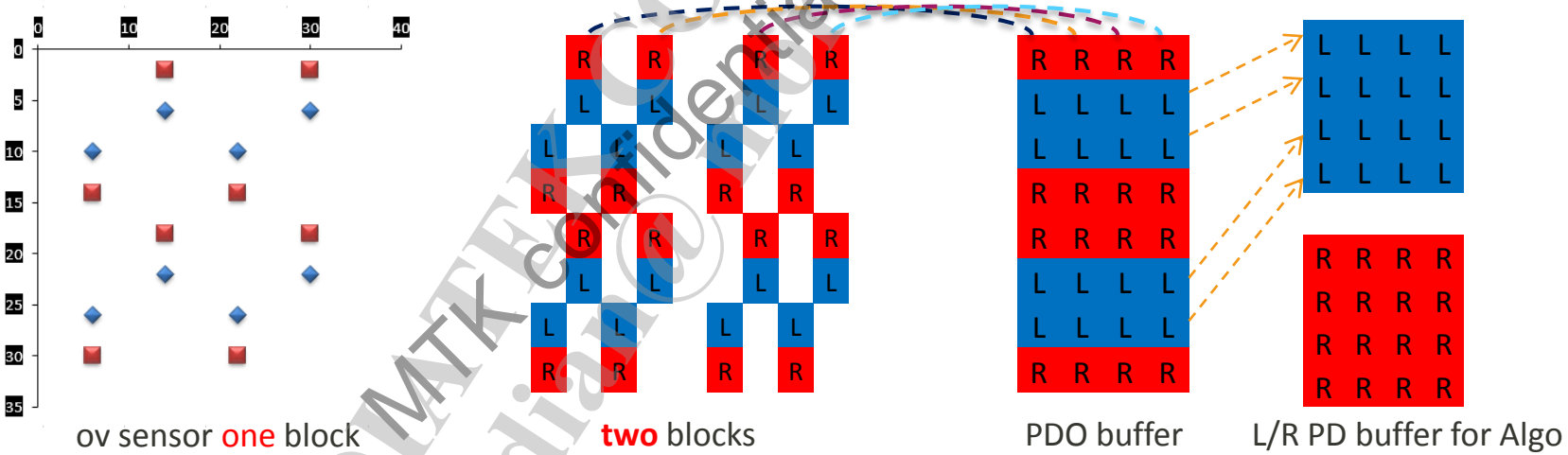
注意：PD buffer的大小与前面计算的PD传送数据量，有联系，但并不等价！

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
PD Buffer Manager	1. 确认当前sensor type对应的Buffer Type，并在pd_buf_list中添加匹配关系 2. 移植相同类型sensor的demo code (根据Buffer Type和PD Pixel排列确定相同类型) 3. 设置**pd_{$sensor}mipiraw.cpp**中的**IsSupport()** 和 **ConvertPDBufFormat()** 4. 合入**BPCI** table　　**PDO(RAW) ONLY**

3. (2)设置**pd_ov13855mipiraw.cpp**中的**ConvertPDBufFormat()**



传送数据量：每行2*132 = **264** pixel 总共 8*98 = **784** line，RLLRRLLR.....RLLRRLLR
PD buffer ：每行2*132 = **264** pixel 总共 8*98 = **784** line，LLLLLLLL.....RRRRRRRR

注意：**PD buffer的大小与前面计算的PD传送数据量相等**，只是**特例**

ConvertPDBufFormat()作用是把平台接收到的PD data，转换为PdAlgo需要的L/R PD buffer

MTK PDAF Porting Case : PDO-RAW

3. (2)设置pd_ov13855mipiraw.cpp中的ConvertPDBufFormat()

函数调用

```
seprate( i4Stride, ptrBufAddr, m_PDXSz, m_PDYSz, m_PDBufSz, m_PDBuf, (b14BitData ? 4 : 2)
```

函数实现

```
void PD_OV13855MIPIRAW::seprate( int stride, unsigned char *ptr, int pd_x_num, int pd_y_num, int LROutSz, unsigned short *ptrLROut, int iShift)
{
    unsigned int pdW = pd_x_num;
    unsigned int pdH = pd_y_num;

    //separate L and R pd data
    unsigned short *ptrbuf = (unsigned short *)ptr;
    unsigned short **ptrtmp = NULL;
    unsigned short *ptrL = ptrLROut;
    unsigned short *ptrR = &ptrLROut[LROutSz/2];

    for ( unsigned int i=0; i < pdH; i++ )
    {
        //RLLR
        if(i%4==0 || i%4==3)
            ptrtmp = &ptrR;
        else
            ptrtmp = &ptrL;


        for ( unsigned int j=0; j < pdW; j++ )
        {
            unsigned short val = ptrbuf[i*stride/2+j];
            (*ptrtmp)[j] = val>>iShift;
        }
        (*ptrtmp) += pdW;
    }
} ? end seprate ?
```

PD buffer **Width** (pixel num)
PD buffer **Height** (line num)

PDO buffer

L PD buffer
R PD buffer

逐行处理PDO buffer

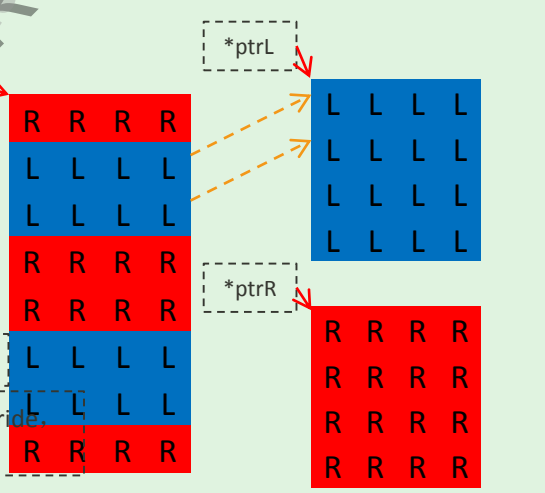
根据PDO buffer排列，把Pixel分别放入L/R buffer

逐Pixel(2Byte)处理，直到处理完一行数据

PDO buffer一行数据总共byte数为stride，
包含pd pixel numer * 2 + padding(0)

*ptrL或*ptrR转到下一行

14bit数据去除后4位0
12bit数据去除后2位0
该特性由PDO HW module决定



ConvertPDBufFormat()调用seprate()，
逐行逐Pixel将PDO buffer中的PD data，
按PdAlgo顺序要求放入L/R PD buffer中

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
PD Buffer Manager	1. 确认当前sensor type对应的Buffer Type，并在pd_buf_list中添加匹配关系 2. 移植相同类型sensor的demo code (根据Buffer Type和PD Pixel排列确定相同类型) 3. 设置**pd_{$sensor}mipiraw.cpp**中的**IsSupport()** 和 **ConvertPDBufFormat()** 4. 合入**BPCI** table **PDO(RAW) ONLY**

4. 合入**BPCI** table

下面这几个文件都必须不重不漏地按照porting guide进行移植修改

(1) *camera_isp_regs_capture_ov13855mipiraw.h*

```
.con    ={.bits={.BPC_EN=1, .rsv_1=0, .BPC_LUT_EN=0, .BPC_TABLE_END_MODE=1,
.con    ={.bits={.PDC_EN=1, .rsv_2=0, .PDC_CT=1, .rsv_5=0, .PDC_MODE=1,
```

(2) *camera_info_ov13855mipiraw.h*

```
#define INCLUDE_FILENAME_BPCI_PARA    "camera_bpci_tbl_ov13855mipiraw.h"
```

(3) *camera_tuning_para_ ov13855mipiraw.cpp*

```
#include INCLUDE_FILENAME_BPCI_PARA
```

```
impGetDefaultData(CAMERA_DATA_TYPE_ENUM cor
{
  UINT32 dataSize[CAMERA_DATA_TYPE_NUM] = {sizeof
      sizeof(NVRAM_CAMERA_3A_STRUCT),
      sizeof(NVRAM_CAMERA_SHADING_STRUCT),
      sizeof(NVRAM_LENS_PARA_STRUCT),
      sizeof(AE_PLINETABLE_T),
      0,
      sizeof(CAMERA_TSF_TBL_STRUCT),
      sizeof(CAMERA_BPCI_STRUCT),
      0,
      sizeof(NVRAM_CAMERA_FEATURE_STRUCT)
  };
```

```
case CAMERA_DATA_TSF_TABLE:
  memcpy(pDataBuf,&CAMERA_TSF_DEFAULT_VALUE,sizeof(CAMERA_TSF_TBL_STRUCT));
  break;
case CAMERA_DATA_PDC_TABLE:
  memcpy(pDataBuf,&CAMERA_PCI_DEFAULT_VALUE,sizeof(CAMERA_BPCI_STRUCT));
  break;
case CAMERA_NVRAM_DATA_FEATURE:
  memcpy(pDataBuf,&CAMERA_FEATURE_DEFAULT_VALUE,sizeof(NVRAM_CAMERA_FEATURE_STRUCT));
  break;
```

设置好这三个文件
BPCI table才能正确生效

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
PD Buffer Manager	1. 确认当前sensor type对应的Buffer Type，并在pd_buf_list中添加匹配关系 2. 移植相同类型sensor的demo code (根据Buffer Type和PD Pixel排列确定相同类型) 3. 设置pd_{$sensor}mipiraw.cpp中的IsSupport() 和 ConvertPDBufFormat() 4. 合入BPCI table PDO(RAW) ONLY

4. 合入BPCI table

下面这几个文件都必须不重不漏地按照porting guide进行移植修改

(4) *camera_bpci_tbl_ov13855mipiraw.h*

```
#define PDC_TBL_1_XSIZE (9407)
#define PDC_TBL_1_YSIZE (0)
#define PDC_PDO_1_XSIZE (527)
#define PDC_PDO_1_YSIZE (783)
static const MUINT8 PDC_1_array[(PDC_TBL_1_XSIZE+1)]={

#define PDC_TBL_2_XSIZE (7103)
#define PDC_TBL_2_YSIZE (0)
#define PDC_PDO_2_XSIZE (527)
#define PDC_PDO_2_YSIZE (591)
static const MUINT8 PDC_2_array[(PDC_TBL_2_XSIZE+1)]={


#define PDC_TBL_3_XSIZE (9407)
#define PDC_TBL_3_YSIZE (0)
#define PDC_PDO_3_XSIZE (527)
#define PDC_PDO_3_YSIZE (783)
static const MUINT8 PDC_3_array[(PDC_TBL_3_XSIZE+1)]={

const CAMERA_BPCI_STRUCT CAMERA_PCI_DEFAULT_VALUE =
{
```

BPCI table用工具生成后，需要手动填入
Table1 for ZSD – full size
Table2 for further use
Table3 for capture pass2 use – full size

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明

任务分类	任务细项
Lens Parameter	Enable PDAF；正确设置PD 相关参数；

```
//i4PDAFCoefs[64]

//=============================================================
//[0] name: pd_default_param
//    range: 0 or 1
//    default: 1
//    constraints: none
//    effect: set 0 will use default value pre- define in algorithm, the following
//          parameter will be invalid.
//-------------------------------------------------------------
1, //[0] pd_default_param

//=============================================================
//[1] name: pd_enable
//    range: 0 or 1
//    default: 1
//    constraints: none
//    effect: set 1 to enable PD source, set 0 OFF
//-------------------------------------------------------------
1, //[1] pd_enable
```

```
//
// Section: Hybrid AF window config
// Description: this section control AF ROI and how many sub PD block use in the ROI.
//
//[1] name: tracking_width
//[2] name: tracking_height
//    range: 1 ~ 100 (%)
//    default: [1]37  [2]39
//    constraints:
//    effect: set how many percent image width and height to be hybrid af ROI at
//          continue mode.
//[3] name: max_pd_win_x
//[4] name: max_pd_win_y
//    range: 1~3
//    default: [3]3 [4]3
//    effect: maximal number of sub PD block use in the tracking_width and
//          tracking_height. default using 3x3 window.
//-------------------------------------------------------------
37, //[1] tracking_width
38, //[2] tracking_height
3, //[3] max_pd_win_x
3, //[4] max_pd_win_y
```

```
//PD_ALGO_TUNING_T
32,
48,
{40, 140,280,420,560},
{384,410,435,461,486},
{
    {0,0,0,0,0,0},
    {0,0,0,0,0,0},
    {0,20,20,20,20,20},
    {0,20,60,60,60,60},
    {0,20,60,60,60,60},
    {0,20,60,60,60,100}
},
240,      //i4SaturateLevel
77,       //i4SaturateThr
1,        //i4ConfThr
```

```
RAW_WIDTH=4224;
RAW_HEIGHT=3136;
PD_DENSITY_X=16;
PD_DENSITY_Y=8;
```

pdnumW <= Width * trackW(%) / MaxPdWinX / DensityW
= 4224 * 37% / 3 / 16 = 32.56

pdnumH <= Height * trackH(%) / MaxPdWinY / DensityH
= 3136 * 38% / 3 / 8 = 49.65

pdnumW和pdnumH都填4的整数倍 => pdnumW=32 pdnumH=48

Lens参数要保证PDAF enable，且其他PD相关参数合理

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明 —— Log分析

GET_SENSOR_PDAF_CAPACITY

当前sensor mode

设置PDAF_Support类型

```
D af_mgr_v3: [getPdInfoForSttCtrl] SensorMode(1), PDAF_Support(1),
  bSensorModeSupportPDAF(1), Info for 3A framework to Cfg sttPipe :PDAFStatus(1),
  PDOSzW(528 PIXELS), PDOSzH(784)
D pd_mgr_list: ID 0xd855, Type 4
D pd_buf_mgr: createInstance SensorId(0xd855) instance(0xecbf8c88) PDBufMgrType(4)
  IsZSD(1) ImgSz(4224 3136) FullSz(4224 3136) PDAF_support(1) IsFrontalBin(0)
  IsPBINen(0) DualPDSeparateMode(0) SensorMode(1) AETargetMode(0) pipeCfg(1)
  curSensorModeSupportPD(1)
D pd_mgr  : ConfThr=1, SaturateLevel=240, i4SaturateThr=77, FocusPDSizeX=36,
  FocusPDSizeY=28, sz=157704
D pd_mgr  : SensorMode: 1(1), SensorInfo : OffsetX(0), OffsetY(0), PitchX(32), PitchY(32),
  PairNu(8), SubBlkW(16), SubBlkH(8), BlockNumX(132), BlockNumY(98), LeFirst(0),
  MirrorFlip(0)
D pd_mgr  : VC2 Info : DataType(0x0), ID(0x0), SIZEH(0x0), SIZEV(0x0)
D pd_mgr  : Orientation 0
D pd_mgr  : [Core] bits(10) isPack(1) w(4224) h(3136) stride(5280) blknumX(132)
  blknumY(98) OffX(0) OffY(0) pairNum(8) pitchX(32) pitchY(32) subblkH(8) subblkW(16)
  crop(0 0 4224 3136)
```

Sensor与 buffer type设定

Lens Parameters

PD_info

这部分log包含了PD Driver + Buffer Type + Lens Para的设定状况

MTK PDAF Porting Case : PDO-RAW

PDO(Raw) type 以 OV13855 举例说明 —— Log分析

```
D PdAlgo  : [parseStep3] cali raw size = (4224, 3136)
D PdAlgo  : [parseStep3] cali raw pd pair num = 8, block num = (132, 98)
D PdAlgo  : [parseStep3] cali raw offset = (0, 0), pitch = (32, 32), density = (16, 8)
D PdAlgo  : [setPDBlockInfo] raw size = (4224, 3136)
D PdAlgo  : [setPDBlockInfo] raw pd pair num = 8, block num = (132, 98)
D PdAlgo  : [setPDBlockInfo] raw offset = (0, 0), pitch = (32, 32), density = (16, 8)
D PdAlgo  : [PdAlgo][setPDBlockInfo] Rotate Type: 0
D pd_mgr  : [Core] configure PD algo done 576 224
D pd_mgr  : ConfigurePDHWSetting BPCI table first 4 dword: ec002 11071 40104010 ec006
D pd_mgr  : ConfigurePDHWSetting PDOHWInfo Bpci_Xsz(9407) Bpci_Ysz(0) Bpci_tbl(0x0)
  pa(0x5400000) va(0xf3462000) memID(42) Pdo_Xsz(527) Pdo_Ysz(783) BitDepth(10) IsDualPD(0)
  PBinType(0) PBinStartLine(0) PdSeparateMode(0)
D pd_mgr  : SensorID(0xd855), PDAF_Support(1), SensorModeSupportPD(1), BufType(0x4) :
  (NOTDEF(0x00), VC(0x01), VC_OPEN(0x11), RAW_LEGACY(0x02), RAW_LEGACY_OPEN(0x12),
  PDO(0x04), PDO_OPEN(0x14), DUALPD_VC(0x21), DUALPD_RAW(0x24)
D af_mgr_v3: AF-Start      : Dev(0x0001), PD Flow (1), SensorMode(1), PDAF_Support_Type(1),
  SensorModeSupportPDAF(1), PDPipeCtrl(1), ImgSz(4224, 3136), FullSz(4224, 3136), IsZSD 1,
  AETargetMode(0), PD Info to Hybrid AF : Sz(10), Data(576 224..)
```

Cali data 解析

Driver设置结果

两者匹配

PDAF Flow启动的log标志

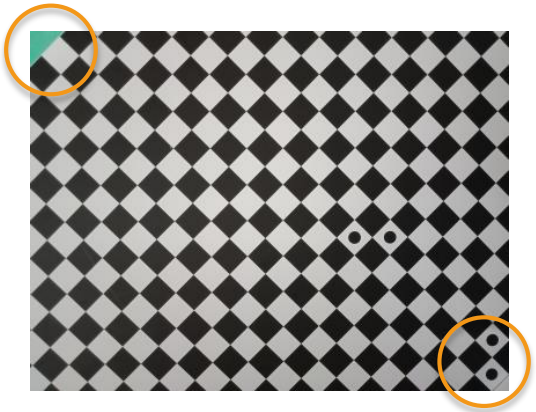
Log显示PDAF Flow开始正常启动，是否导通还需要进行Verification

MEDIATEK

PDAF Verification

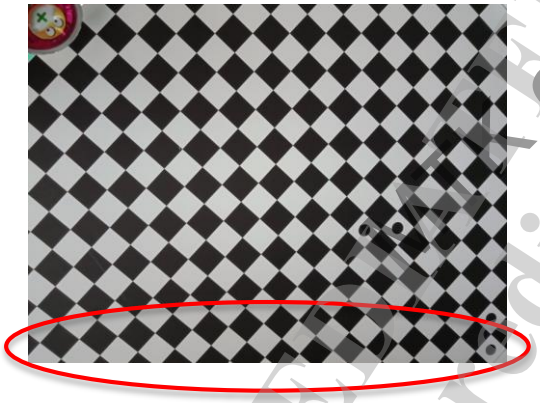
1. 环境准备

（1）在三脚架上固定手机，**30cm**对着菱形图

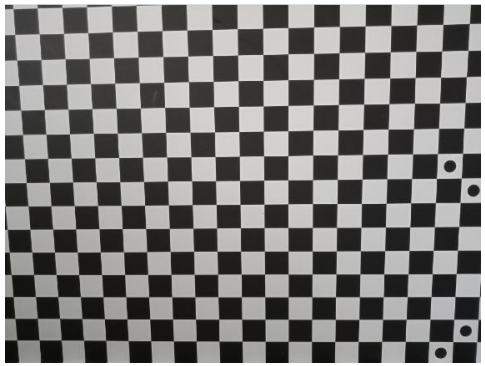


OK：**菱形图**场景，**水平线垂直线**摆正
边角处有明显标志

NG：普通办公室场景



NG：菱形图场景，未摆正



NG：方形图场景

1. 环境准备

（2）实时确认当前测试场景中所有PD windows的confidence大于60

adb shell logcat | find "Confidence"

```
pd_mgr   : win 0 [Confidence] 100 [LensPos] 483->506 [value] -862
pd_mgr   : win 0 [Confidence] 100 [LensPos] 483->497 [value] -526
pd_mgr   : win 0 [Confidence] 100 [LensPos] 483->499 [value] -591
pd_mgr   : win 0 [Confidence] 100 [LensPos] 483->484 [value] -81
pd_mgr   : win 0 [Confidence] 100 [LensPos] 483->488 [value] -218
pd_mgr   : win 0 [Confidence] 100 [LensPos] 483->496 [value] -494
pd_mgr   : win 0 [Confidence] 100 [LensPos] 483->480 [value] 94
```

OK,可在该场景做PD Verification

```
pd_mgr   : win 0 [Confidence] 0 [LensPos] 321->526 [value] -7368
pd_mgr   : win 0 [Confidence] 20 [LensPos] 321->386 [value] -2378
pd_mgr   : win 0 [Confidence] 0 [LensPos] 321->312 [value] 335
pd_mgr   : win 0 [Confidence] 60 [LensPos] 321->344 [value] -866
pd_mgr   : win 0 [Confidence] 60 [LensPos] 344->368 [value] -879
pd_mgr   : win 0 [Confidence] 60 [LensPos] 368->419 [value] -1878
pd_mgr   : win 0 [Confidence] 20 [LensPos] 414->462 [value] -1759
```

NG, 需要更换场景做PD Verification

*pd windows数量需根据项目需求由lens参数调整

2. 确认信息

(1) 固定手机，做一次Touch AF，在log中找到inf/macro/in-focus三个DAC

```
af mgr v3: readOTP : [Inf]348 [Macro]706
AfAlgo  : [Speed][AdpComp][BlackFaceAF][ZEE] ------ adjusted Peak 612 pos
```

(2) 确认Algo打印信息满足如下条件：

- [CalC] r > 0.90，NG请检查PDAF porting

```
PdAlgo  : [calC] r = 0.542208,   NG
PdAlgo  : [calC] r = 0.984958,   OK
```

- (L-m)与(r-m) 的相差值 < 300，NG请检查PDAF porting

```
PdAlgo  : [sPD] l-m = 6261, r-m = 5310   NG
PdAlgo  : [sPD] l-m = 6187, r-m = 5383

PdAlgo  : [sPD] l-m = 4833, r-m = 4757   OK
PdAlgo  : [sPD] l-m = 6604, r-m = 6341
```

- ISO < 200，NG请更换明亮场景做Verification

```
                                                                                                    NG
AfAlgoC : [handleAFin0](1) Mode=3,Status=0,StatusC=0,AFSTrigger=0,FirstEnterCam=1,CurPos=0,ISO=3438(100,9600),
AfAlgoC : [handleAFin0](1) Mode=3,Status=0,StatusC=0,AFSTrigger=0,FirstEnterCam=1,CurPos=0,ISO=142(100,9600),
                                                                                                    OK
```

(3) 测试场景截屏

*adb shell **screencap** /sdcard/pd_verification_scene01.png*

3. PD线性度测试

(1) Set log property

adb shell setprop debug.af_mgr.enable 1
adb shell setprop debug.pd.enable 1
adb shell setprop debug.af.enable 1
adb shell setprop debug.dump_pdaf_cali.enable 1
adb shell setprop debug.af_motor.disable 1

(2) 开始录制log，进入camera（此时VCM已经被固定)

(3) 测试录屏

adb shell screenrecord /sdcard/pd_verification.mp4

(3) Move lens 到想要的DAC – 从inf点到Macro点分10步以上移动

adb shell setprop debug.af_motor.position [$value]

(4) 实验结束后pull出data

/sdcard/mtklog/mobilelog　　/sdcard/pdo/

4. Dump PD Buffer & Log

(1) 设定属性

adb shell setprop pd.dump.enable 1

adb shell setprop pdo.dump.enable 1

adb shell setprop debug.af_mgr.enable 1

adb shell setprop debug.pd.enable 1

adb shell setprop debug.af.enable 1

(2) 进入camera PDAF(ZSD)预览**2秒**立刻退出camera，并把dump data和log取出

*adb pull **/sdcard/pdo***

adb pull /sdcard/mtklog/mobilelog

pd_LR_8_11.raw	2017/9/18 12:04	IrfanView RAW File
pd_LR_10_13.raw	2017/9/18 12:04	IrfanView RAW File
pd_LR_14_17.raw	2017/9/18 12:04	IrfanView RAW File
pdo_1_142.raw	2017/9/18 12:04	IrfanView RAW File
pdo_2_143.raw	2017/9/18 12:04	IrfanView RAW File
pdo_3_144.raw	2017/9/18 12:04	IrfanView RAW File

(3) 退出时把下列属性置零，避免影响后续camera performance

adb shell setprop pd.dump.enable 0

adb shell setprop pdo.dump.enable 0

5. 数据分析

(1) 把PD线性度测试结果log，放入AF basic tuning pre-check excel中分析

```
pd_mgr  : win 0 [Confidence] 100 [LensPos] 483->506 [value] -862
pd_mgr  : win 0 [Confidence] 100 [LensPos] 483->497 [value] -526
```

Confidence　CurrentPostion　TargetPostion　PD Value

第(1)项分析结果

(2) 把PD Buffer dump data打开，check L/R转换是否正常，或发送给MTK进一步分析

L

R

第(2)项分析结果

MEDIATEK

PDAF Debug and Trouble Shooting

PDAF CR

【反馈log】
请开如下log属性，30厘米固定手机对着【菱形图】(若是方格图请旋转45度)，进行verification，并反馈log
adb shell setprop debug.af_mgr.enable 1
adb shell setprop debug.af.enable 1
adb shell setprop debug.pd.enable 1

【反馈File】
请反馈如下File or Folder到e-service

Sensor Driver	/{$kernel}/drivers/misc/mediatek/imgsensor/src/{$project or platform}/{$sensor}
EEPROM Driver	/vendor/mediatek/proprietary/custom/{$project or platform}/hal/imgsensor_src /{$kernel}/drivers/misc/mediatek/cam_cal/
Pd_Buf_Mgr	/vendor/mediatek/proprietary/custom/{$project or platform}/hal/pd_buf_mgr/{$sensor} /vendor/mediatek/proprietary/custom/{$project or platform}/hal/pd_buf_mgr/src /vendor/mediatek/proprietary/custom/{$project or platform}/hal/inc/pd_buf_mg
Lens Para	/vendor/mediatek/proprietary/custom/{$project or platform}/hal/lens/{$VCM}
BPCI Table + ISP Para	/vendor/mediatek/proprietary/custom/{$project or platform}/hal/imgsensor/{$sensor}
PD sensor INI file	来自sensor vendor或模组厂，描述PD Block信息及L/R坐标， porting时根据此文档填写sensor driver中 pd_info
PD sensor Reference Manuel	关于sensor PDAF相关spec，如Virtual Channel如何传输、Sensor端BPC如何补偿等
PD module eeprom layout	模组eeprom烧录表格

Debug LogProp & ScreenSave

请参考PDAF Verification章节进行导通测试

- 打开 “MTKLogger”
- adb cmd:
  - ***adb shell setprop debug.af_mgr.enable 1***
  - ***adb shell setprop debug.af.enable 1***
  - ***adb shell setprop debug.pd.enable 1***
- Run camera (**ZSD**)
- Go back to MTKLogger to stop logging
- Pull out logs
  - ***adb pull /sdcard/mtklog/mobilelog***

- 录制手机屏幕：
  - ***adb shell screenrecord /sdcard/sr_default.mp4***
  - 视频格式为mp4，默认录制时间为180s，可以按Ctrl+C来停止录制。

- 测试场景截屏：
  - ***adb shell screencap /sdcard/sc_default.png***

Debug points

（一）PD thread not run

- 1. 检查是否打开ZSD，isZSD = 1
- 2. "Load PDAF calib data error!!“
  - 检查下面打印出来的读取的pd calibration data， 正确是0x50,0x44,0x30,0x31,否则检查pdafotp driver
  - (1)看是否没忽略flag和checksum；
  - (2)查起始地址和size是否正确
  - (3)查i2c地址是否正确
  - (4) 查当前是从eeprom读取还是NVRAM读取数据
- 3. 是否按照porting guide仔细导通driver和buf_mgr；

Debug points

（二）Confidence始终为零

- 1. 修正lens_para参数，确保PDAF开关打开
- 2. dump L/R raw data

Debug points

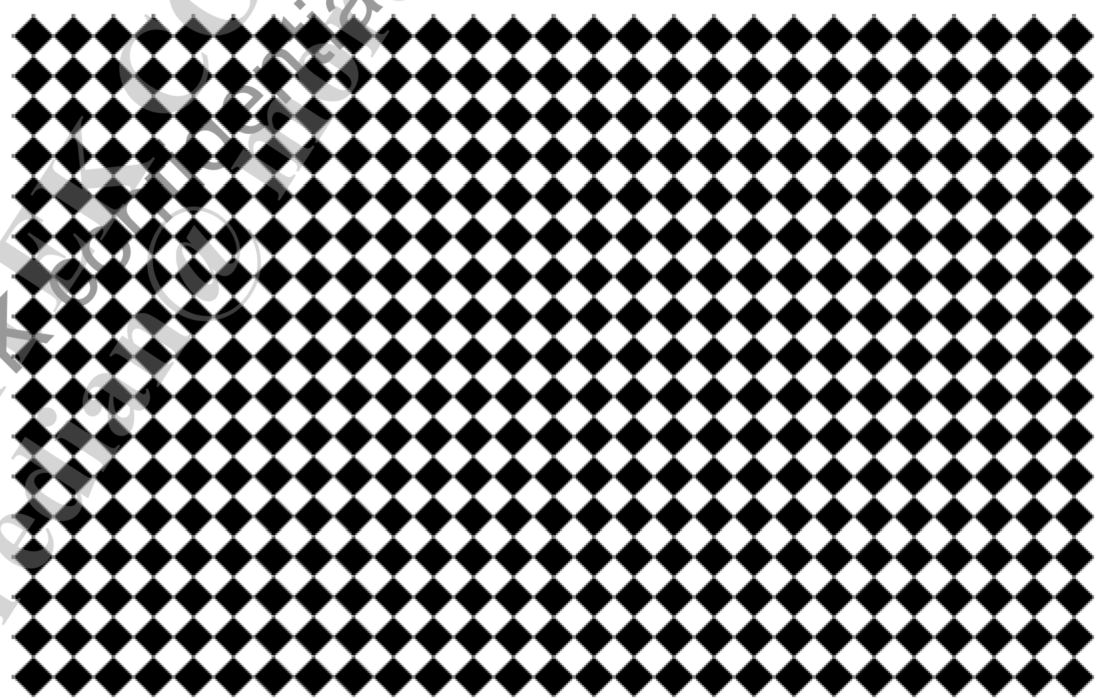
（三）Confidence低，经常秀对焦框

- 1. 确认lens_para中的PD参数是否合理；
- 2. log解析的calibration数据是否正常，且block size相等；
- 3. 用adb手动移动lens，看PD算法结果是否算反、若反了可互换L/R buffer
- 4. dump calibration data；

Debug points

PDAF导通现象（必要非充分）

- 1. 用MTK Camera AP进行远近景切换时，lens有移动进行对焦，且不显示对焦框；
- 2. 对着菱形图log中confidence有100，失焦时pd value大于1500



菱形图

block size不一致

异常log：

【模组烧录的PD calibration data如下】

PdAlgo : [parseStep3] cali raw size = (4208, 3120)

PdAlgo : [parseStep3] cali raw pd pair num = 16, block num = (65, 48)

PdAlgo : [parseStep3] cali raw offset = (28, 31), pitch = (64, 64), density = (16, 16)

【driver中的PD信息log打印如下】

PdAlgo : [setPDBlockInfo] raw size = (4208, 3120)

PdAlgo : [setPDBlockInfo] raw pd pair num = 16, block num = (64, 47)

PdAlgo : [setPDBlockInfo] raw offset = (28, 31), pitch = (64, 64), density = (16, 16)

→driver和cali数据不匹配

【Solution】

1. 首先确认两个**raw size** 是否一致； *parseStep3*打印出的是模组calibration时的full size；*setPDBlockInfo*打印出的是sensor driver实际输出的size；若不一致，先check哪个size是正确的

2. 若两个**raw size**修正为一致后，block num仍不一致，可参考**如下**两种改法：

查看*kd_imgsensor_define.h*中的*SET_PD_BLOCK_INFO_T*结构体是否包含*.i4BlockNumX*和 *.i4BlockNumY*

(1) **若有，**则可以在sensor driver的*imgsensor_pd_info*结构体中配置正确的*.i4BlockNumX* 和 *.i4BlockNumY*

(2) **若没有**，则将*pd_mgr.cpp*中如下代码修改：

a_sPDConfig.sPdBlockInfo.i4BlockNumX = (m_profile.uImgXsz-2*sPDSensorInfo.i4OffsetX)/sPDSensorInfo.i4PitchX;

改为

a_sPDConfig.sPdBlockInfo.i4BlockNumX = (**int**)(((**double**)(m_profile.uImgXsz-2*sPDSensorInfo.i4OffsetX)/(**double**) sPDSensorInfo.i4PitchX)+**0.5**);

nvram size和read size不匹配

异常log:

10-18 10:37:09.104082 421 2889 E PdAlgo :
[parseCaliData()] Err: 1570:, ParseCaliData error size: (nvram size, read size) = (1404, 1372)

→nvram size和read size不匹配

【Solution】

1. nvram size 是在pd_XXXxxxmipiraw.cpp文件中的GetPDCalSz()函数定义的

```
MINT32 PD_XXXxxxMIPIRAW::GetPDCalSz()
{
        return 0x57c;          //1404
}
```

2. read size 是在pdaf otp read eeprom实际读取出来的data size，在pdaf otp driver中

3. 跟模组厂确认，eeprom中烧录的有效数据的size正确的是多少，
注意参考本文档Page20-23 (读取PD calibration data 部分)

4. 本例中是模组厂给的size为1404 (X)，实际上应该是1372，所以修改pd_XXXxxxmipiraw.cpp文件中的 GetPDCalSz() 为return 0x54c; 即可

读取calibration data error

72900 01-01 00:02:56.318 784 5264 D af_mgr : NVRAM NO PDAF calib data, read from EEPROM !!
73010 01-01 00:02:56.405 784 5264 E PdAlgo : [parseCaliData()] Err: 1489:, ParseCaliData error tag in PD01:
73076 01-01 00:02:56.418 784 5264 D pd_mgr : Load PDAF calib data error!!
73195 01-01 00:02:56.445 784 5264 D pd_mgr : [Core] PD init data error, close PDAF
73196 01-01 00:02:56.446 784 5264 D pd_mgr : close PD mgr thread

-> 修改pdafotp driver

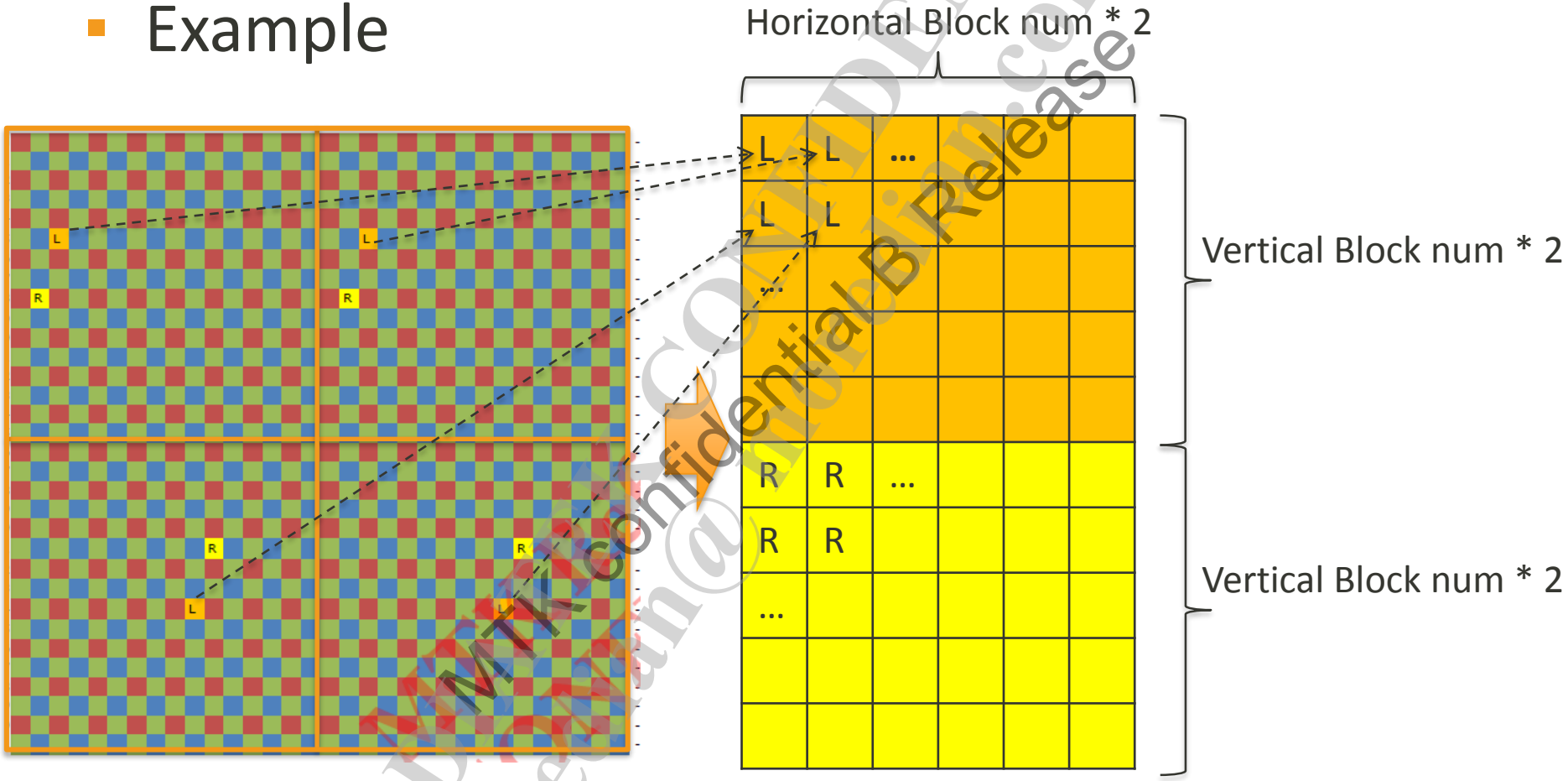
Appendix

MEDIATEK

PDAF Convert PdBuffer Guide

PD Data Format Conversion

- Example



以OV和三星的sensor为例，分两种情况说明： （1）VC （2）PDO

PD Data Format Conversion(VC)

OV sensor ConvertPDBufFormat( )函数的实现

– 1. Extract pd data from dma buffer

(1) 若buffer送来的是16bit的数据，则可参考**imx258**实现方式：

```
int k=0;
MUINT16 *tmpbuf = new MUINT16 [pdSz];
for( int i=0; i<sz2Bbuf; i++)
{
    if(ptr2Bbuf[i]!=0)
    {
        tmpbuf[k] = ptr2Bbuf[i];
        k++;
    }
}
```

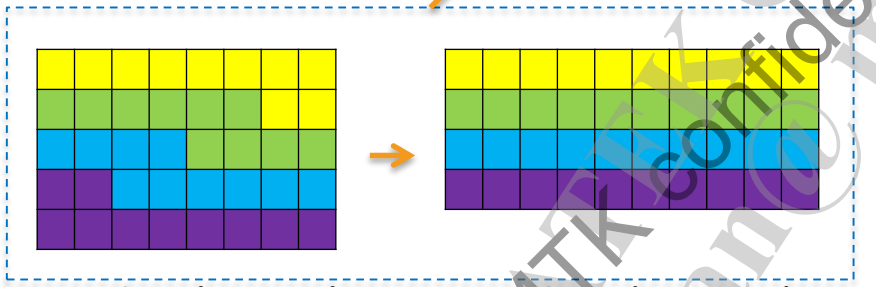
imx258

PD Data Format Conversion(VC)

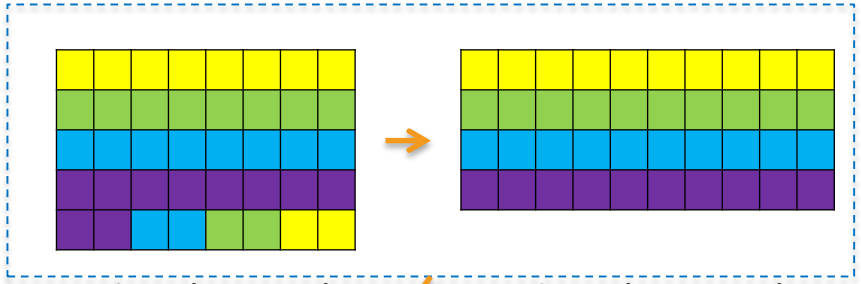
OV sensor ConvertPDBufFormat( )函数的实现

(2)若buffer送来的是10bit的数据，则需参考code进行转换(分两种情况)

```
//convert format from DMA buffer format(Raw10) to pixel format
MUINT16 *ptrbuf = new MUINT16 [0xA8*0x800];
MUINT32 i,j,k;
for( j=0, k=0; j<0x800; j++)
{
    for( i=0; i<0xD2; i+=5)
    {
        //*
        ptrbuf[k  ] = ( ((ptrBufAddr[ j*0xD4 + (i+1)]&0x3 ) <<8) &0x300) | ((ptrBufAddr[ j*0xD4 + (i  )]>>0)&0xFF);
        ptrbuf[k+1] = ( ((ptrBufAddr[ j*0xD4 + (i+2)]&0xF ) <<6) &0x3C0) | ((ptrBufAddr[ j*0xD4 + (i+1)]>>2)&0x3F);
        ptrbuf[k+2] = ( ((ptrBufAddr[ j*0xD4 + (i+3)]&0x3F) <<4) &0x3F0) | ((ptrBufAddr[ j*0xD4 + (i+2)]>>4)&0xF );
        ptrbuf[k+3] = ( ((ptrBufAddr[ j*0xD4 + (i+4)]&0xFF) <<2) &0x3FC) | ((ptrBufAddr[ j*0xD4 + (i+3)]>>6)&0x3 );
        k+=4;
    }
}
```



10bits(1 Byte)　　10bits (2 Bytes)
高位补0



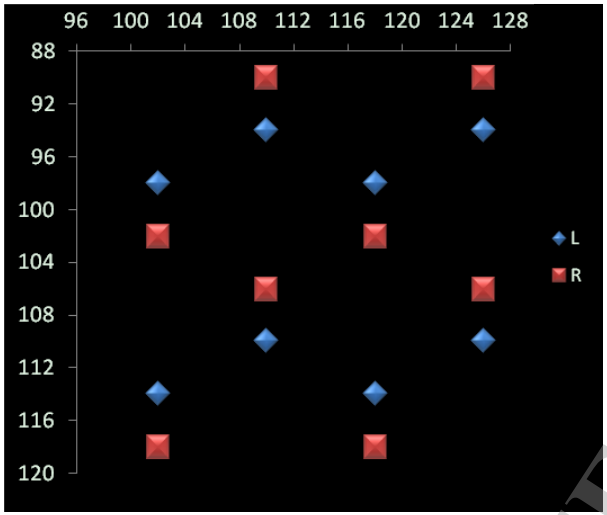
10bits(1 Byte)　　10bits (2 Bytes)
高位补0

```
ptrbuf[k  ] = ((ptrBufAddr[ j*0xD4 + (i  )] << 2)&0x3FC) | ((ptrBufAddr[ j*0xD4 + (i+4)]>>0) &0x3);
ptrbuf[k+1] = ((ptrBufAddr[ j*0xD4 + (i+1)] << 2)&0x3FC) | ((ptrBufAddr[ j*0xD4 + (i+4)]>>2) &0x3);
ptrbuf[k+2] = ((ptrBufAddr[ j*0xD4 + (i+2)] << 2)&0x3FC) | ((ptrBufAddr[ j*0xD4 + (i+4)]>>4) &0x3);
ptrbuf[k+3] = ((ptrBufAddr[ j*0xD4 + (i+3)] << 2)&0x3FC) | ((ptrBufAddr[ j*0xD4 + (i+4)]>>6) &0x3);
```

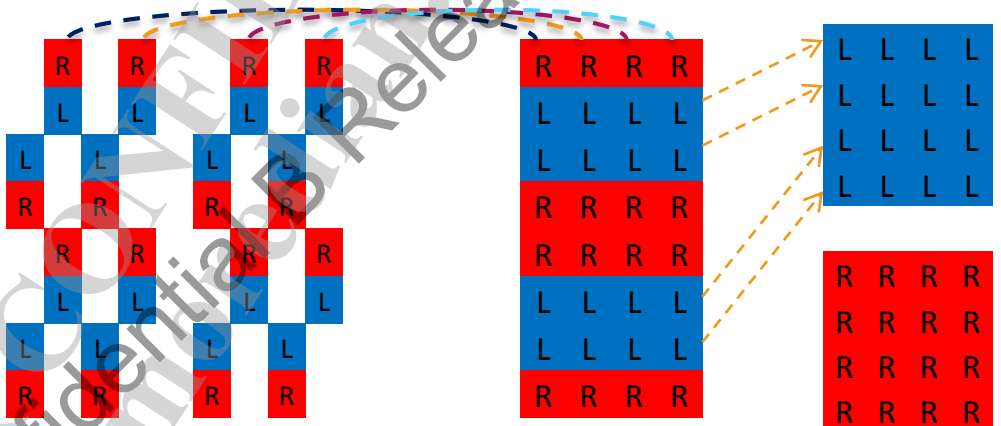
PD Data Format Conversion(VC)

OV sensor ConvertPDBufFormat( )函数的实现

– 2. Convert format to PD core algorithm input



ov sensor block

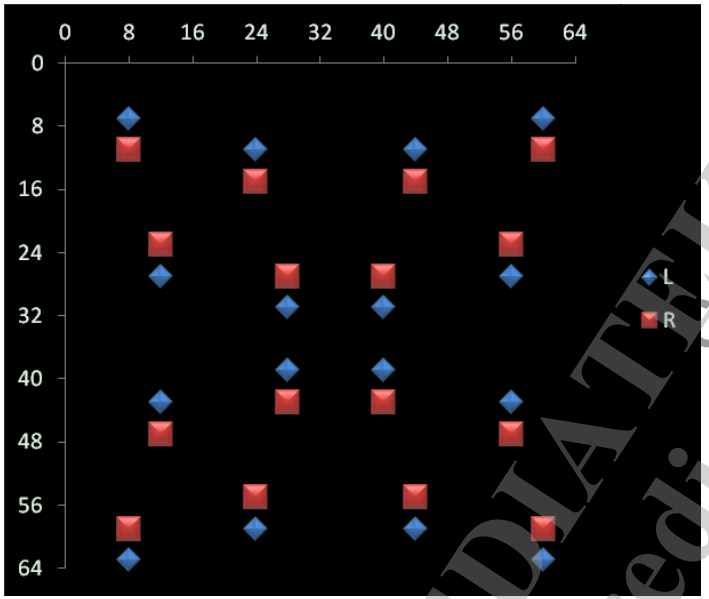



```
MUINT16 **ptr=NULL;
MUINT16 *ptrL =    m_PDBuf;
MUINT16 *ptrR = &(m_PDBuf[(h/2)*w]);

for ( i=0; i < h; i++ )
{
    if(i%4==0 || i%4==3)
        ptr = &ptrR;
    else
        ptr = &ptrL;


    for ( int j=0; j < w; j++ )
    {
        (*ptr)[j] = ptrbuf[i*w+j];
    }
    (*ptr) += w;
}
```

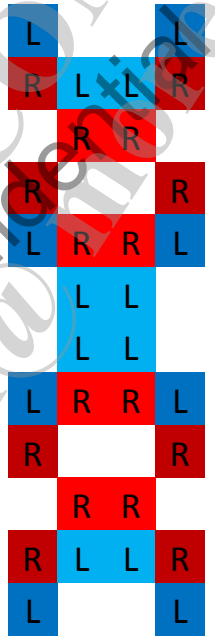
PD Data Format Conversion(VC)

Samsung sensor ConvertPDBufFormat( )函数的实现

- 1. Extract pd data from dma buffer
  - 与前述OV sensor相同，不再赘述
- 2. Convert format to PD core algorithm input

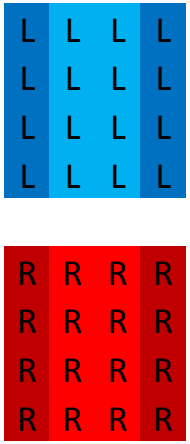


Samsung sensor block



one blocks

buffer

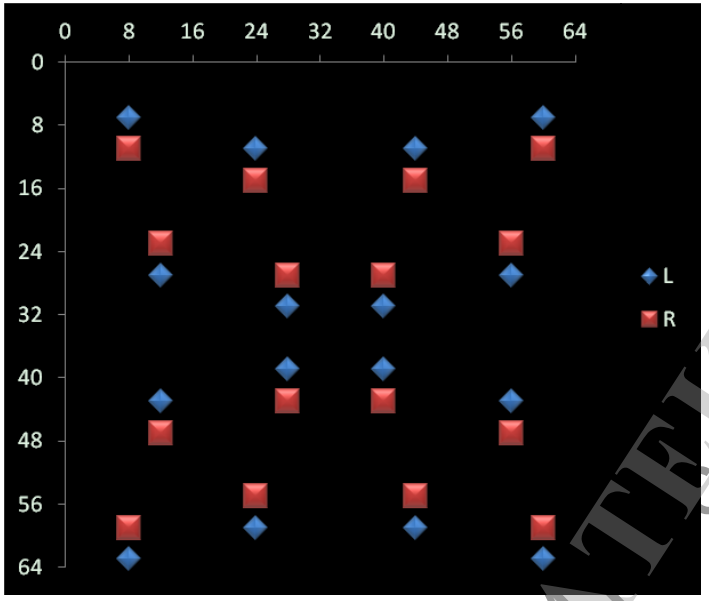


L/R

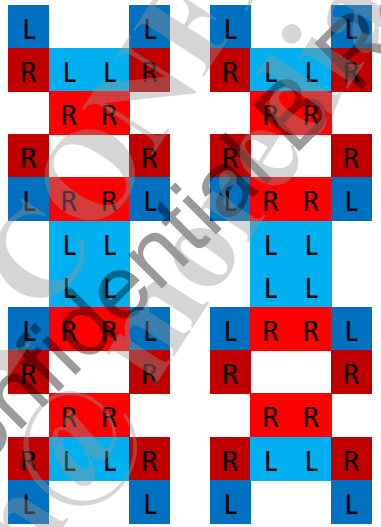
PD Data Format Conversion(VC)

Samsung sensor ConvertPDBufFormat( )函数的实现

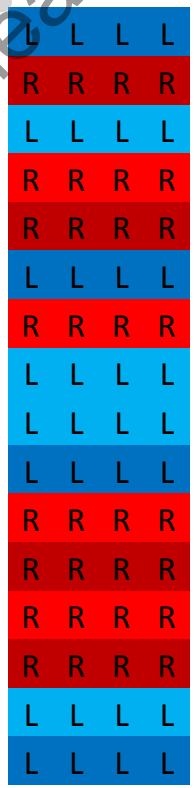
– 2. Convert format to PD core algorithm input



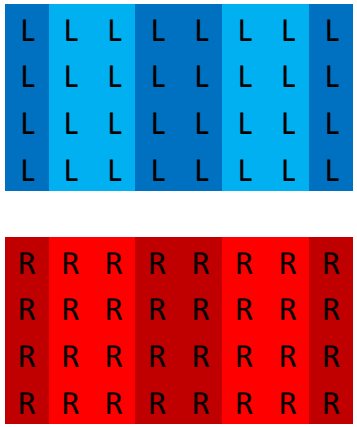
Samsung sensor block



two blocks



buffer



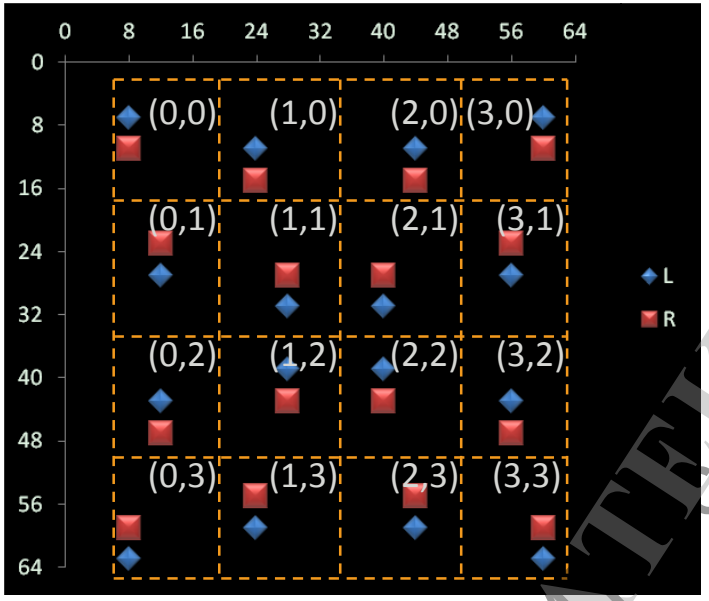
L/R

PD Data Format Conversion(VC)

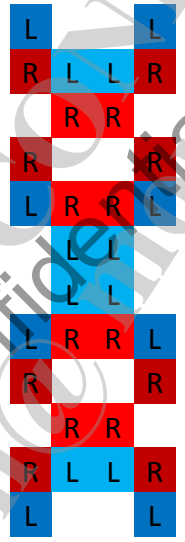
Samsung sensor ConvertPDBufFormat( )函数的实现

– 2. Convert format to PD core algorithm input

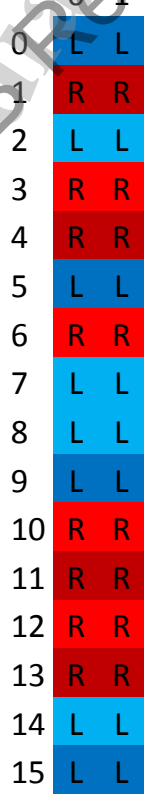
(1) 对每个PD点的位置进行描述，建立block mapping table



Samsung sensor block



one blocks



buffer

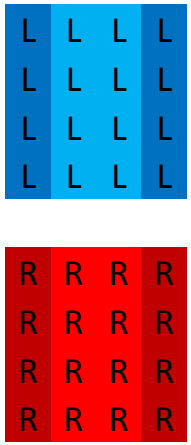
	buffer 列号	buffer 行号	block x坐标	block y坐标	[0] L [1] R
	0	0	0	0	0
	1	0	3	0	0
	0	1	0	0	1
	1	1	3	0	1
	0	2	1	0	0
	1	2	2	0	0
	0	3	1	0	1
	1	3	2	0	1
	0	4	0	1	1
	1	4	3	1	1
	0	14	1	3	0
	1	14	2	3	103 0
	0	15	0	3	0
	1	15	3	3	0

Samsung sensor ConvertPDBufFormat( )函数的实现

– 2. Convert format to PD core algorithm input

(2) 应用mapping table，分离L/R图，block by block

buffer



L/R

i	j	m	n	d
0	0	0	0	0
1	0	3	0	0
0	1	0	0	1
1	1	3	0	1
0	2	1	0	0
1	2	2	0	0
0	3	1	0	1
1	3	2	0	1
0	4	0	1	1
1	4	3	1	1
0	14	1	3	0
1	14	2	3	0
0	15	0	3	0
1	15	3	3	0

```
typedef struct {
    MUINT32 PD_Data_Index_i;
    MUINT32 PD_Data_Index_j;
    MUINT32 Sub_Block_Index_m;
    MUINT32 Sub_Block_Index_n;
    MUINT32    L_OR_R_Pixel_Flag;
}PD_Map_Table_Type;
/* Constuct a mapping table from the pd output data index */
static const PD_Map_Table_Type  PD_Map_Table[PD_MAP_TABLE_SIZE]={
{0,0,0,0,0},{1,0,3,0,0},{0,1,0,0,1},{1,1,3,0,1},      /*1*/
{0,2,1,0,0},{1,2,2,0,0},{0,3,1,0,1},{1,3,2,0,1},      /*2*/
{0,4,0,1,1},{1,4,3,1,1},{0,5,0,1,0},{1,5,3,1,0},      /*3*/
{0,6,1,1,1},{1,6,2,1,1},{0,7,1,1,0},{1,7,2,1,0},      /*4*/
{0,8,1,2,0},{1,8,2,2,0},{0,9,0,2,0},{1,9,3,2,0},      /*5*/
{0,10,1,2,1},{1,10,2,2,1},{0,11,0,2,1},{1,11,3,2,1},  /*6*/
{0,12,1,3,1},{1,12,2,3,1},{0,13,0,3,1},{1,13,3,3,1},  /*7*/
{0,14,1,3,0},{1,14,2,3,0},{0,15,0,3,0},{1,15,3,3,0}   /*8*/
};
```

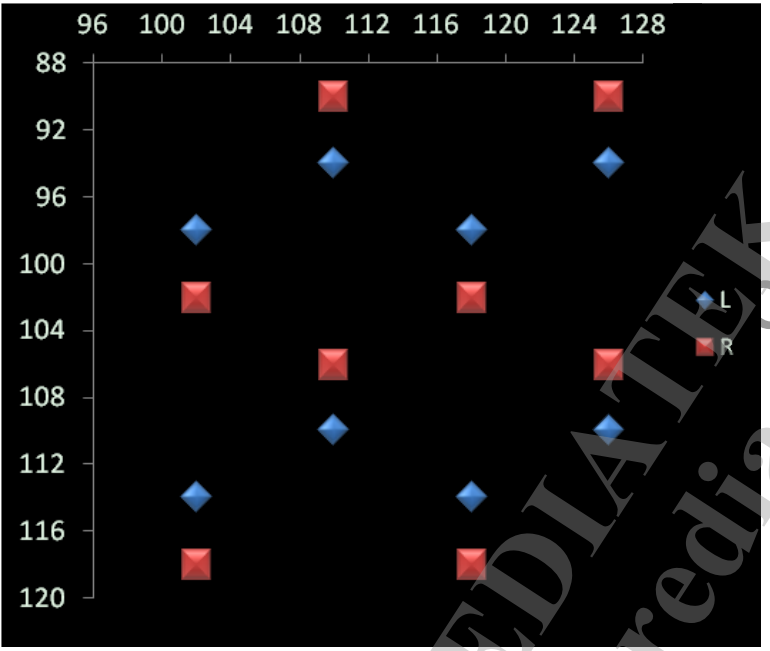
3个for循环

```
/* lookup the mapping table to fill the L/R PD buffer block by block */
    /*convert format to PD core algorithm input */
   MUINT16 *   L_PDBuf = m_PDBuf;
   MUINT16 *   R_PDBuf = &(m_PDBuf[m_PDBufSz/2]);
   for(MUINT32 y = 0;y < Block_Num_H;y++)                  /* Block_Num_H = 48 */
    {
       for(MUINT32 x =0;x < Block_Num_W; x++)               /*  Block_Num_W = 65  */
       {
         for(MUINT32 k =0; k < PD_MAP_TABLE_SIZE; k++)   /* PD_MAP_TABLE_SIZE = 32 */
          {

            if(PD_Map_Table[k].L_OR_R_Pixel_Flag == 0)   /* L PD pixel */
              {

                L_PDBuf[(4*y+PD_Map_Table[k].Sub_Block_Index_n)*260 + x*4 + PD_Map_Table[k].Sub_Block_Index_m] =  \
                                        PD_Data[(y*16+PD_Map_Table[k].PD_Data_Index_j)*132 + x*2 + PD_Map_Table[k].PD_Data_Index_i];

              }
            else if(PD_Map_Table[k].L_OR_R_Pixel_Flag == 1) /*  R PD pixel */
              {

                R_PDBuf[(4*y+PD_Map_Table[k].Sub_Block_Index_n)*260 + x*4 + PD_Map_Table[k].Sub_Block_Index_m] = \
                                        PD_Data[(y*16+PD_Map_Table[k].PD_Data_Index_j)*132 + x*2 + PD_Map_Table[k].PD_Data_Index_i];
              }


          } /* _END_ for  k */
       } ? end for MUINT32x=0;x<Block_Nu... ?  /* _END_ for  x */
    } ? end for MUINT32y=0;y<Block_Nu... ?  /* _END_ for  y */
```

PD Data Format Conversion(PDO)

OV sensor ConvertPDBufFormat( )函数的实现

- 1. 收到的是pixel数据, no need convert PD buffer format
- 2. Convert format to PD core algorithm input



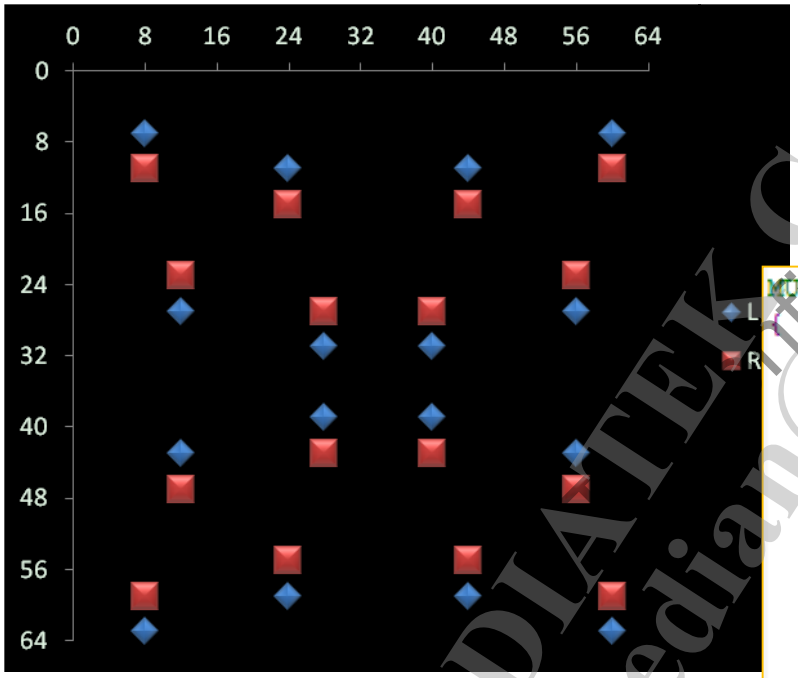
OV sensor

```
for ( int i=0; i < pdH; i++ )
{
    //RLLR
    if(i%4==0 || i%4==3)
        ptrtmp = &ptrR;
    else
        ptrtmp = &ptrL;

    for ( int j=0; j < pdW; j++ )
    {
        unsigned short val = ptrbuf[i*stride/2+j];
        (*ptrtmp)[j] = val>>2;
    }

    (*ptrtmp) += pdW;
}
```

PD Data Format Conversion(PDO)

Samsung sensor ConvertPDBufFormat( )函数的实现

- 1. 收到的是pixel数据, no need convert PD buffer format
- 2. Convert format to PD core algorithm input



Samsung sensor

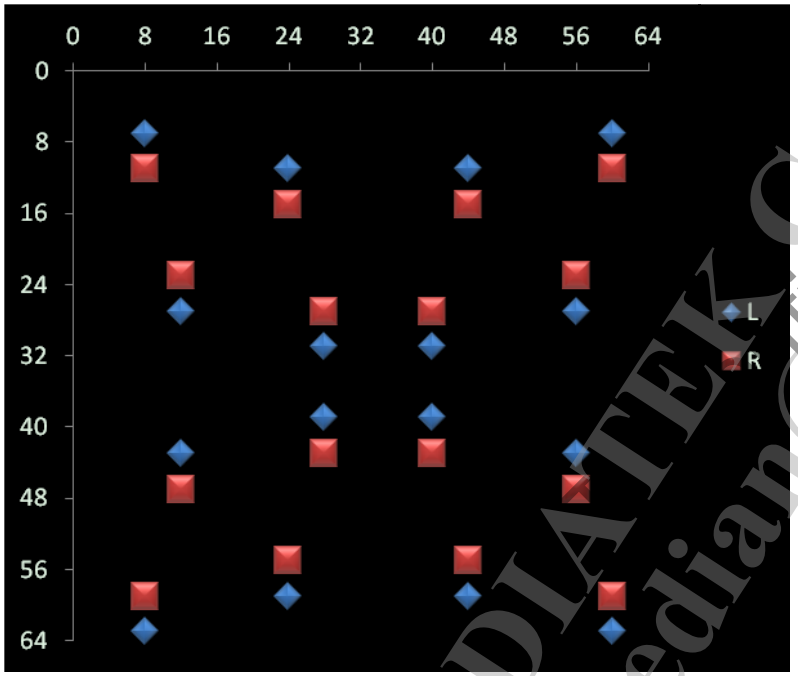
*PD_S5K2X8MIPIRAW::seprate()*函数将同一行的LR数据分开，具体实现可查看code

```
MUINT16* PD_S5K2X8MIPIRAW::ConvertPDBufFormat( MUINT32 i4Size,
{
    //s5k2x8 is EPDBuf_Raw type, no need convert PD buffer format.

    //first in allocate local PD buffer directly.
    if( m_PDBuf==NULL)
    {
        //vaild pd data size
        m_PDXSz    = (pdo_xsize_s5k2x8+1)/2;
        m_PDYSz    = (pdo_ysize_s5k2x8+1)*2/3;
        m_PDBufSz = m_PDXSz*m_PDYSz;
        m_PDBuf = new MUINT16 [m_PDBufSz];
    }

    seprate( i4Stride, ptrBufAddr, m_PDXSz, m_PDYSz, m_PDBuf);

    return m_PDBuf;
}
```

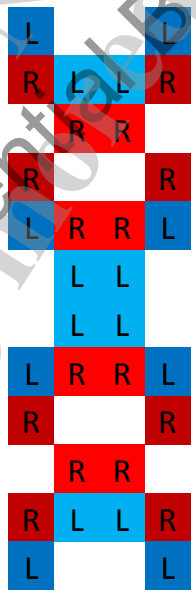
PD Data Format Conversion(PDO)

Samsung sensor ConvertPDBufFormat( )函数的实现

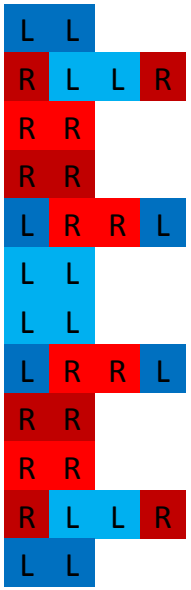
- 1. 收到的是pixel数据, no need convert PD buffer format
- 2. Convert format to PD core algorithm input



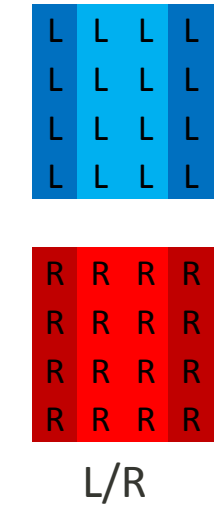
Samsung sensor



one blocks



buffer

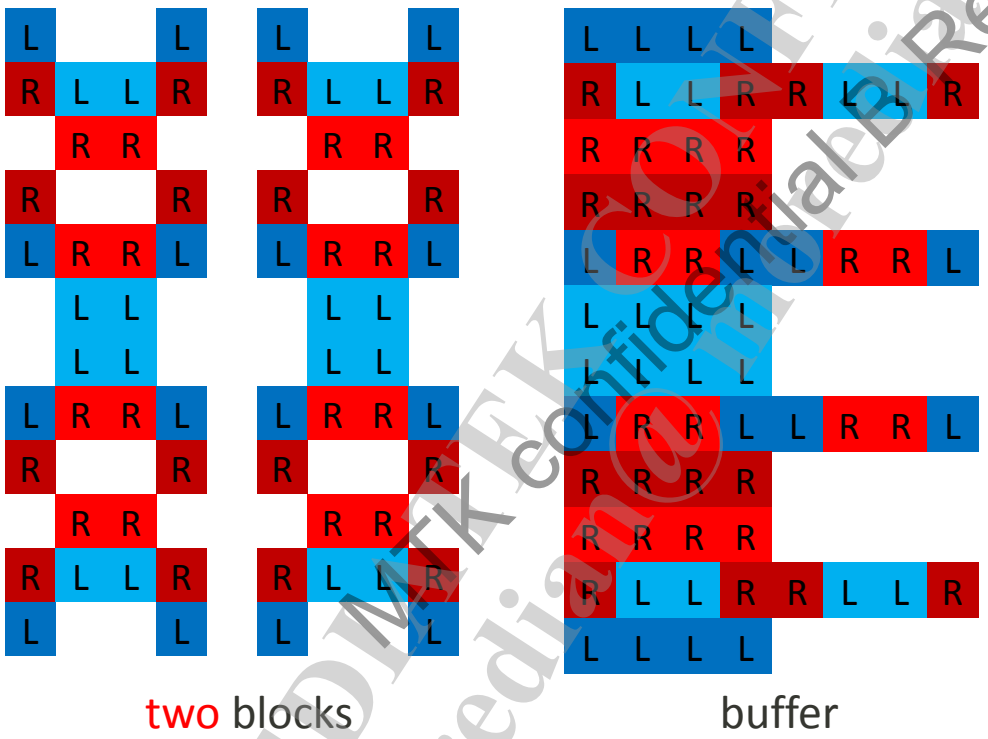


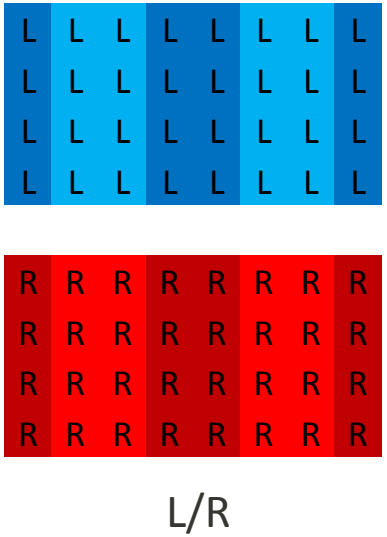
L/R

PD Data Format Conversion(PDO)

Samsung sensor ConvertPDBufFormat( )函数的实现

– 2. Convert format to PD core algorithm input



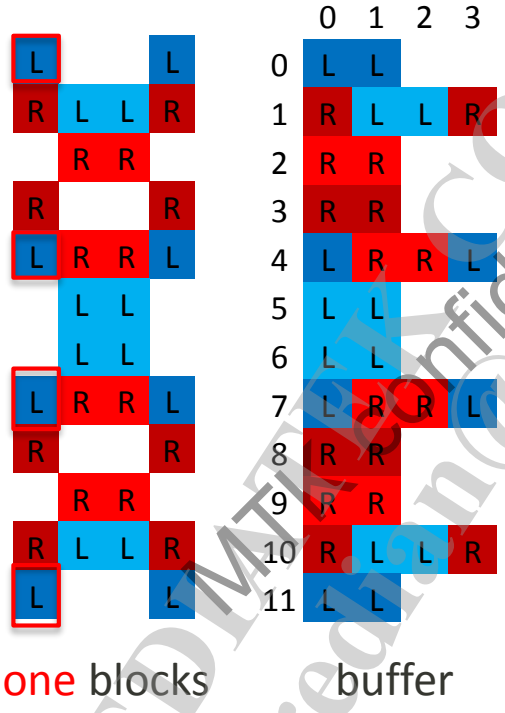


PD Data Format Conversion(PDO)

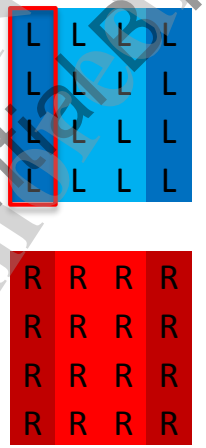
Samsung sensor ConvertPDBufFormat( )函数的实现

– 2. Convert format to PD core algorithm input

(1) 对每个PD点的位置进行描述，对应L/R，结合buffer坐标，建立table



one blocks buffer



L/R

0,0	1,1	1,2	0,1
4,0	5,0	5,1	4,3
7,0	6,0	6,1	7,3
11,0	10,1	10,2	11,1
1,0	2,0	2,1	1,3
3,0	4,1	4,2	3,1
8,0	7,1	7,2	8,1
10,0	9,0	9,1	10,3

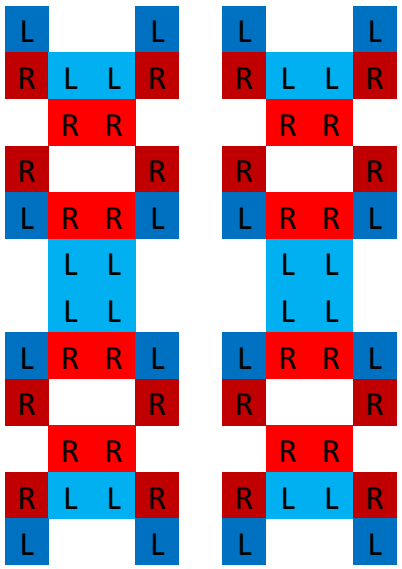
stride是buffer一行的总byte数，除以2换算为pixel数

```
table[4*8]=
        0,    stride*1/2+1,  stride*1/2+2,              1,
stride*4/2,   stride*5/2+0,  stride*5/2+1,  stride*4/2+3,
stride*7/2,   stride*6/2+0,  stride*6/2+1,  stride*7/2+3,
stride*11/2,  stride*10/2+1, stride*10/2+2, stride*11/2+1,

stride*1/2,   stride*2/2+0,  stride*2/2+1,  stride*1/2+3,
stride*3/2,   stride*4/2+1,  stride*4/2+2,  stride*3/2+1,
stride*8/2,   stride*7/2+1,  stride*7/2+2,  stride*8/2+1,
stride*10/2,  stride*9/2+0,  stride*9/2+1,  stride*10/2+3
```

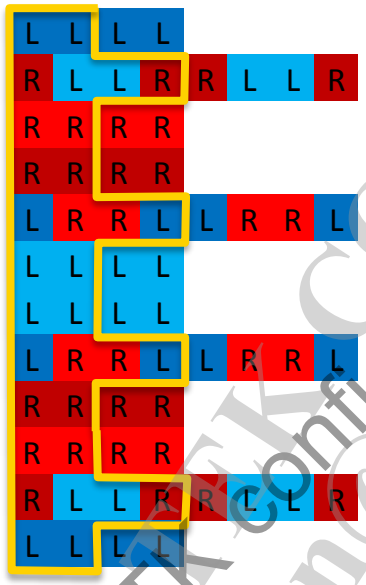
PD Data Format Conversion(PDO)

Samsung sensor ConvertPDBufFormat( )函数的实现

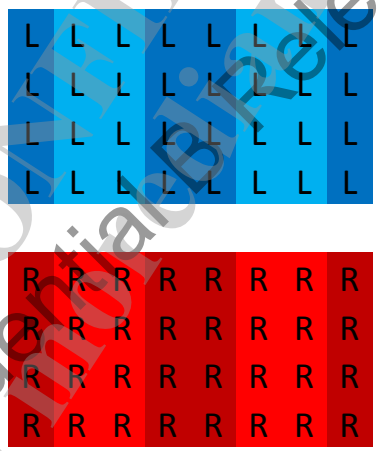
– 2. Convert format to PD core algorithm input



two blocks



buffer



L/R

2	4	4	2
4	2	2	4
4	2	2	4
2	4	4	2
4	2	2	4
2	4	4	2
2	4	4	2
4	2	2	4

```
multiple[4*8]= { 2, 4, 4, 2,
                 4, 2, 2, 4,
                 4, 2, 2, 4,
                 2, 4, 4, 2,

                 4, 2, 2, 4,
                 2, 4, 4, 2,
                 2, 4, 4, 2,
                 4, 2, 2, 4
               };
```

```
void PD_S5K2X8MIPIRAW::seprate( int stride, unsigned char *ptr, int pd_x_num, int pd_y_num, unsigned short *ptrLROut)
{
    unsigned int table[4*8]= {                                  0,    stride*1/2+1,  stride*1/2+2,              1,
                                                          stride*4/2,   stride*5/2+0,  stride*5/2+1,  stride*4/2+3,
                                                          stride*7/2,   stride*6/2+0,  stride*6/2+1,  stride*7/2+3,
                                                          stride*11/2,  stride*10/2+1, stride*10/2+2, stride*11/2+1,

                                                          stride*1/2,   stride*2/2+0,  stride*2/2+1,  stride*1/2+3,
                                                          stride*3/2,   stride*4/2+1,  stride*4/2+2,  stride*3/2+1,
                                                          stride*8/2,   stride*7/2+1,  stride*7/2+2,  stride*8/2+1,
                                                          stride*10/2,  stride*9/2+0,  stride*9/2+1,  stride*10/2+3
                                };

    unsigned int multiple[4*8]= { 2, 4, 4, 2,
                                  4, 2, 2, 4,
                                  4, 2, 2, 4,
                                  2, 4, 4, 2,

                                  4, 2, 2, 4,
                                  2, 4, 4, 2,
                                  2, 4, 4, 2,
                                  4, 2, 2, 4
                                };


    unsigned short *tempMap = (unsigned short *)ptr;                        //input ptrBufAddr
    unsigned short *ConvBuf1_tmpMap = ptrLROut;                             //output
    unsigned short *pout5 = ConvBuf1_tmpMap;                                //L port0
    unsigned short *pout6 = ConvBuf1_tmpMap + pd_x_num*1;                   //L port1
    unsigned short *pout7 = ConvBuf1_tmpMap + pd_x_num*2;                   //L port2
    unsigned short *pout8 = ConvBuf1_tmpMap + pd_x_num*3;                   //L port3

    unsigned short *pout1 = ConvBuf1_tmpMap + pd_x_num*(pd_y_num/2);        //R port0
    unsigned short *pout2 = ConvBuf1_tmpMap + pd_x_num*((pd_y_num/2)+1);    //R port1
    unsigned short *pout3 = ConvBuf1_tmpMap + pd_x_num*((pd_y_num/2)+2);    //R port2
    unsigned short *pout4 = ConvBuf1_tmpMap + pd_x_num*((pd_y_num/2)+3);    //R port3

    int count=0, idx=0;
```

```
for( int i=0; i<pd_y_num/8; i++)
{
    for( int j=0; j<pd_x_num; j+=4)
    {
        unsigned int *ptable1 = table;
        unsigned int *ptable2 = ptable1+4;
        unsigned int *ptable3 = ptable2+4;
        unsigned int *ptable4 = ptable3+4;
        unsigned int *ptable5 = ptable4+4;
        unsigned int *ptable6 = ptable5+4;
        unsigned int *ptable7 = ptable6+4;
        unsigned int *ptable8 = ptable7+4;

        unsigned int *pmultiple1 = multiple;
        unsigned int *pmultiple2 = pmultiple1+4;
        unsigned int *pmultiple3 = pmultiple2+4;
        unsigned int *pmultiple4 = pmultiple3+4;
        unsigned int *pmultiple5 = pmultiple4+4;
        unsigned int *pmultiple6 = pmultiple5+4;
        unsigned int *pmultiple7 = pmultiple6+4;
        unsigned int *pmultiple8 = pmultiple7+4;

        for( int k=0; k<4; k++)
        {
            *pout1 = tempMap[(*ptable1)+((*pmultiple1)*count)+idx]>>2;
            *pout2 = tempMap[(*ptable2)+((*pmultiple2)*count)+idx]>>2;
            *pout3 = tempMap[(*ptable3)+((*pmultiple3)*count)+idx]>>2;
            *pout4 = tempMap[(*ptable4)+((*pmultiple4)*count)+idx]>>2;
            *pout5 = tempMap[(*ptable5)+((*pmultiple5)*count)+idx]>>2;
            *pout6 = tempMap[(*ptable6)+((*pmultiple6)*count)+idx]>>2;
            *pout7 = tempMap[(*ptable7)+((*pmultiple7)*count)+idx]>>2;
            *pout8 = tempMap[(*ptable8)+((*pmultiple8)*count)+idx]>>2;
            ptable1++;
            ptable2++;
            ptable3++;
            ptable4++;
            ptable5++;
            ptable6++;
            ptable7++;
            ptable8++;
            pmultiple1++;
            pmultiple2++;
            pmultiple3++;
            pmultiple4++;
            pmultiple5++;
            pmultiple6++;
            pmultiple7++;
            pmultiple8++;
            pout1++;
            pout2++;
            pout3++;
            pout4++;
            pout5++;
            pout6++;
            pout7++;
            pout8++;
        } ? end for intk=0;k<4;k++ ?
        count++;
    } ? end for intj=0;j<pd_x_num;j+=4 ?
    pout1 += 3*pd_x_num;
    pout2 += 3*pd_x_num;
    pout3 += 3*pd_x_num;
    pout4 += 3*pd_x_num;
    pout5 += 3*pd_x_num;
    pout6 += 3*pd_x_num;
    pout7 += 3*pd_x_num;
    pout8 += 3*pd_x_num;
    count  = 0;
    idx   += stride/2*12;
} ? end for inti=0;i<pd_y_num/8;i++ ?
} ? end seprate ?
```