MTK confidential B

OTP Driver porting guide

Outline

- Terminologies
- Sensor OTP porting
- Platform OTP porting
  - OTP old/new architecture
  - Old architecture OTP porting
  - New architecture OTP porting
- Case study

Terminologies

- OTP: One Time Programmable.
- LSC,AWB,AF Calibration data:
  - **LSC**： 基于单体模组在DNP 光源下进行shading 的基础补偿. 一般是补偿到65%-75%, ISP 在此基础上再做补偿. 一般按照M*N*8+68烧录
  - **AWB**： 统计单体模组中心区域10% 的R/G 和B/G 的数值, 烧录到sensor的寄存器或EEPROM中.
  - **AF**： 记录(近景) 10CM和(远景) 3M 开外甚至更远距离 VCM 的Step . AF OTP calibration需要平台处理.所以该文档只介绍AF OTP数据的读取.

- **Platform OTP**:
  - Sensor 没有 OTP的自校正功能, 需要我们BB 端进行校正.
  - 从存储空间（外挂eeprom或者sensor内部存储空间）中read出数据, 然后将数据送给BB进行calibration.
- **Sensor OTP**：
  - Sensor 有 OTP的自校正功能.
  - 从存储空间（外挂eeprom或者sensor内部存储空间）中read出数据, 然后写回sensor寄存器，
  - 送到BB 端的 RawData 是已经校正过的数据

Sensor OTP porting

- Sensor端OTP只需将烧录的calibration data读出来，写回sensor寄存器。Driver可单独实现一份OTP driver， 也可写在sensor driver里边。

参考DCC上文档： **Sensor_OTP_Porting_Guide.pptx**

INTERNAL USE

Sensor端OTP导通指南

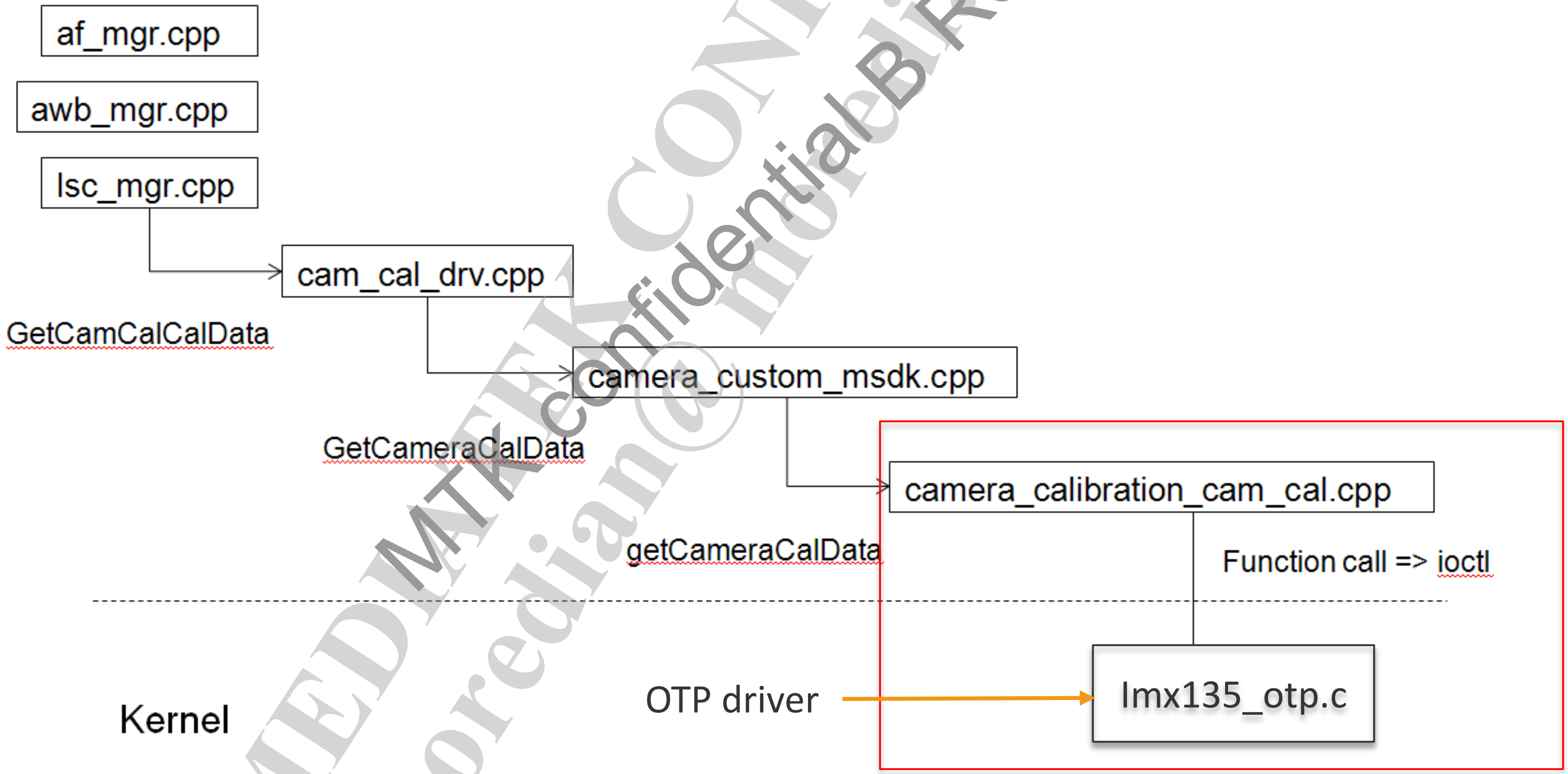
FOR MT6595/K2

Outline

- Terminologies
- Sensor OTP porting
- Platform OTP porting
  - OTP old/new architecture
  - Old architecture OTP porting
  - New architecture OTP porting
- Case study

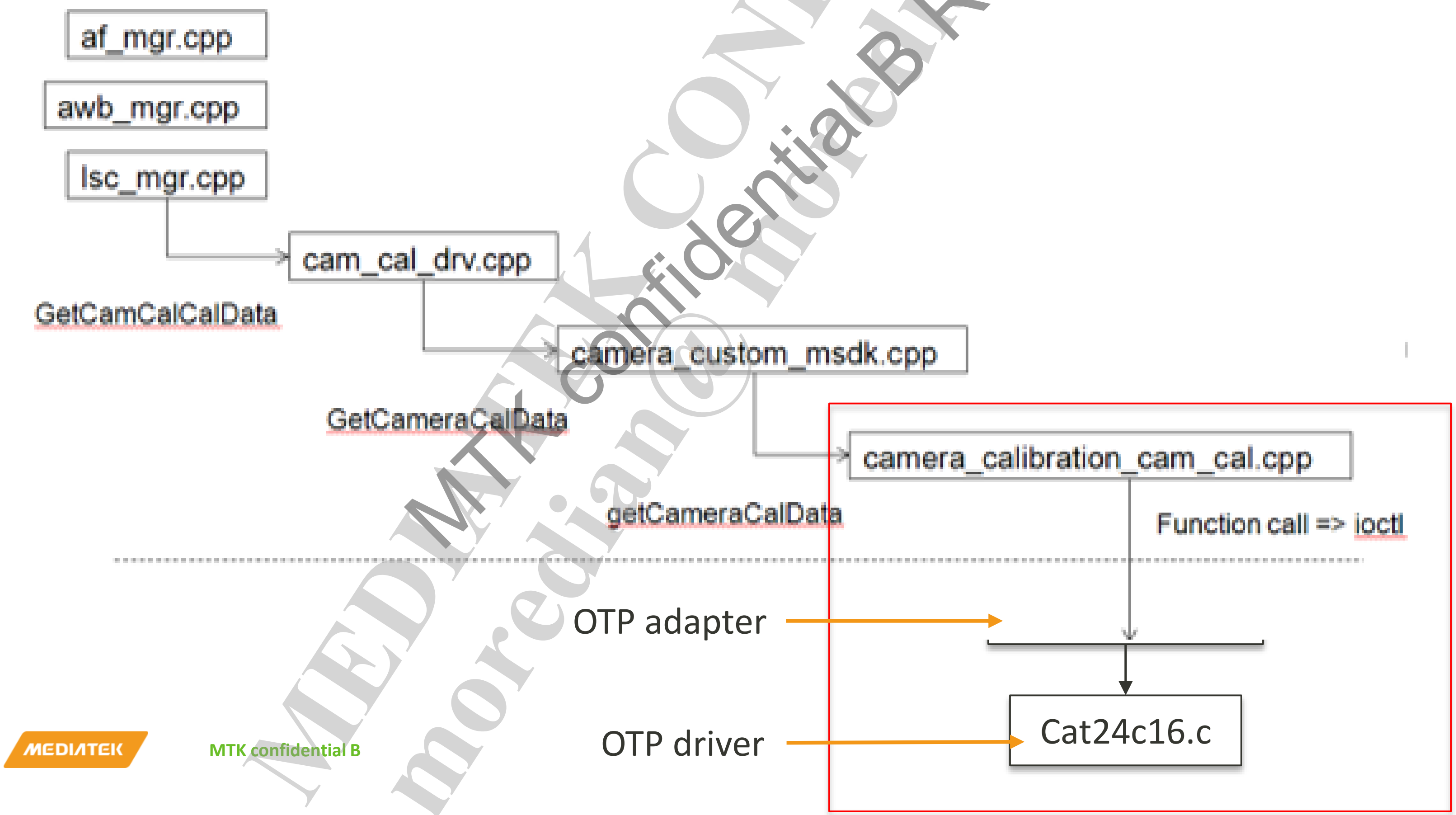
Old arch of Platform OTP

- Old arch：
  - kernel-3.18\drivers\misc\mediatek\cam_cal\src\legacy\$PLATFORM\
  - kernel-3.18\drivers\misc\mediatek\cam_cal\src\legacy\mt6755\dummy_eeprom\**
  - kernel-3.18\drivers\misc\mediatek\cam_cal\src\legacy\mt6755\imx258_eeprom\**
  - kernel-3.18\drivers\misc\mediatek\cam_cal\src\legacy\mt6755\imx135_otp\**



New arch of Platform OTP

- New arch：(using cam_cal_drv.c / cam_cal_list.c as OTP adapter)
  - kernel-3.18\drivers\misc\mediatek\cam_cal\src\cam_cal_drv.c
    kernel-4.4\drivers\misc\mediatek\cam_cal\src\eeprom_driver.c(MT6763)
  - kernel-3.18\drivers\misc\mediatek\cam_cal\src\cam_cal_list.c
  - kernel-3.18\drivers\misc\mediatek\cam_cal\src\common\dummy_eeprom\ **



af_mgr.cpp

awb_mgr.cpp

lsc_mgr.cpp

cam_cal_drv.cpp

GetCamCalCalData

camera_custom_msdk.cpp

GetCameraCalData

camera_calibration_cam_cal.cpp

getCameraCalData

Function call => ioctl

OTP adapter

Cat24c16.c

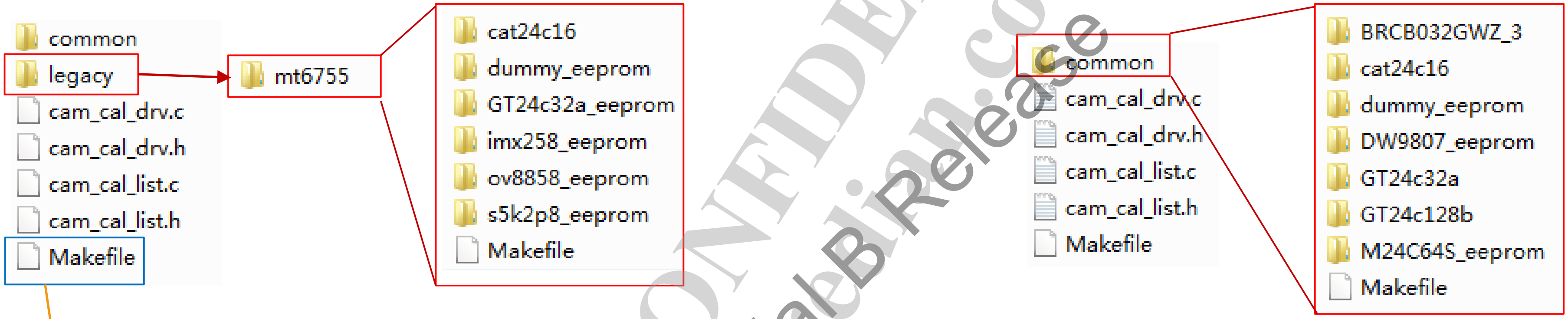
OTP driver

Old arch or New arch?

- Use new arch default from MT6757
  If you using old arch, MTK will not support
- Red highlight means: use old arch default but can be modified to use new arch

	MT6735	MT6755	MT6757	MT6797	MT6799
Android M	Old arch	Old/New arch	New arch	New arch	—
Android **N**	Old/New arch	Old/New arch	New arch	New arch	New arch

Old/New arch contrast



```
common
legacy
cam_cal_drv.c
cam_cal_drv.h
cam_cal_list.c
cam_cal_list.h
Makefile
```

```
mt6755
```

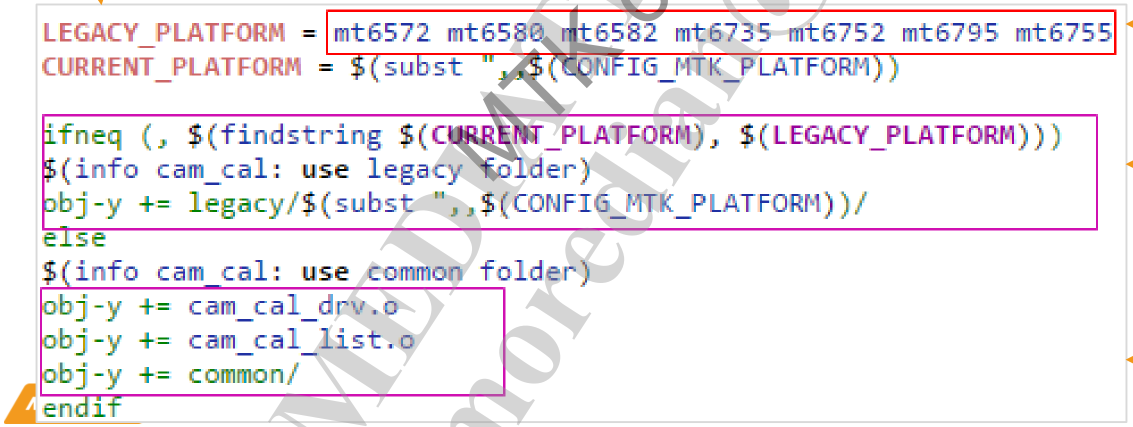
```
cat24c16
dummy_eeprom
GT24c32a_eeprom
imx258_eeprom
ov8858_eeprom
s5k2p8_eeprom
Makefile
```

Old arch

```
common
cam_cal_drv.c
cam_cal_drv.h
cam_cal_list.c
cam_cal_list.h
Makefile
```

```
BRCB032GWZ_3
cat24c16
dummy_eeprom
DW9807_eeprom
GT24c32a
GT24c128b
M24C64S_eeprom
Makefile
```

New arch



```
LEGACY_PLATFORM = mt6572 mt6580 mt6582 mt6735 mt6752 mt6795 mt6755
CURRENT_PLATFORM = $(subst ",,$(CONFIG_MTK_PLATFORM))

ifneq (, $(findstring $(CURRENT_PLATFORM), $(LEGACY_PLATFORM)))
$(info cam_cal: use legacy folder)
obj-y += legacy/$(subst ",,$(CONFIG_MTK_PLATFORM))/
else
$(info cam_cal: use common folder)
obj-y += cam_cal_drv.o
obj-y += cam_cal_list.o
obj-y += common/
endif
```

Use old arch default

Build old arch

Build new arch

Outline

- Terminologies
- Sensor OTP porting
- Platform OTP porting
  - OTP old/new architecture
  - Old architecture OTP porting
  - New architecture OTP porting
- Case study

Old arch OTP Porting Guide

- Please refer to this guide file on DCC

  Platform_OTP_Porting_Guide.pptx

MEDIATEK

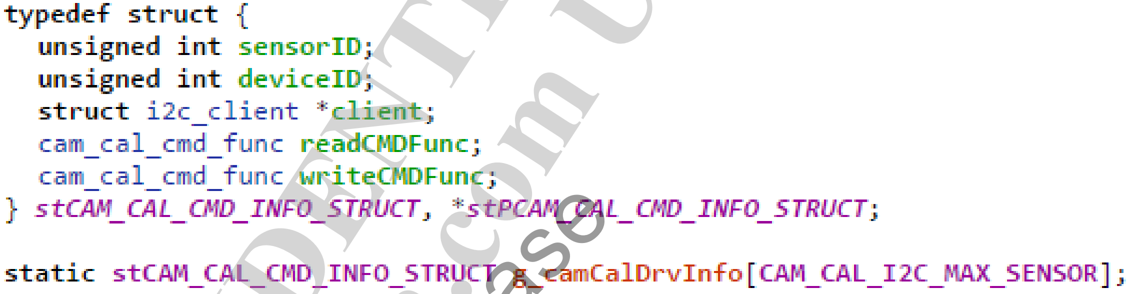
平台端OTP导通指南

FOR MT6595/K2

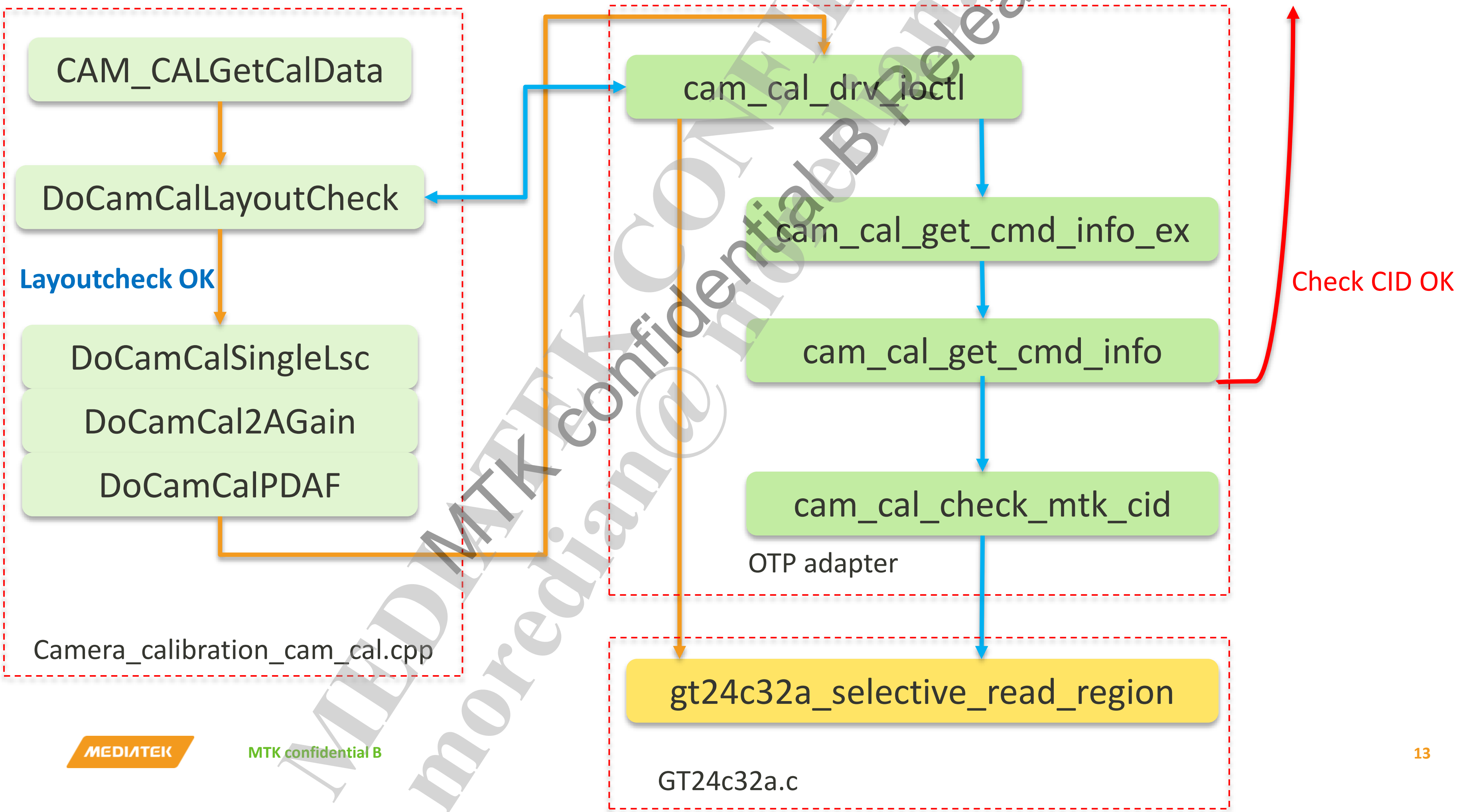
Outline

- Terminologies
- Sensor OTP porting
- Platform OTP porting
  - OTP old/new architecture
  - Old architecture OTP porting
  - New architecture OTP porting
- Case study

OTP driver flow



```
typedef struct {
   unsigned int sensorID;
   unsigned int deviceID;
   struct i2c_client *client;
   cam_cal_cmd_func readCMDFunc;
   cam_cal_cmd_func writeCMDFunc;
} stCAM_CAL_CMD_INFO_STRUCT, *stPCAM_CAL_CMD_INFO_STRUCT;

static stCAM_CAL_CMD_INFO_STRUCT g_camCalDrvInfo[CAM_CAL_I2C_MAX_SENSOR];
```



CAM_CALGetCalData

DoCamCalLayoutCheck

Layoutcheck OK

DoCamCalSingleLsc

DoCamCal2AGain

DoCamCalPDAF

Camera_calibration_cam_cal.cpp

cam_cal_drv_ioctl

cam_cal_get_cmd_info_ex

cam_cal_get_cmd_info

Check CID OK

cam_cal_check_mtk_cid

OTP adapter

gt24c32a_selective_read_region

GT24c32a.c

Config file list

File list:

- Projectconfig.mk中的CAM_CAL不需要配置
- Kernel modification:

  kernel-3.18\arch\arm64\boot\dts\mediatek\mt6757.dtsi
  kernel-4.4\drivers\misc\mediatek\dws\mt6763\$Project.dws (MT6763)
  kernel-3.18\arch\arm64\configs\$Project_debug_defconfig
  kernel-3.18\arch\arm64\configs\$Project_defconfig

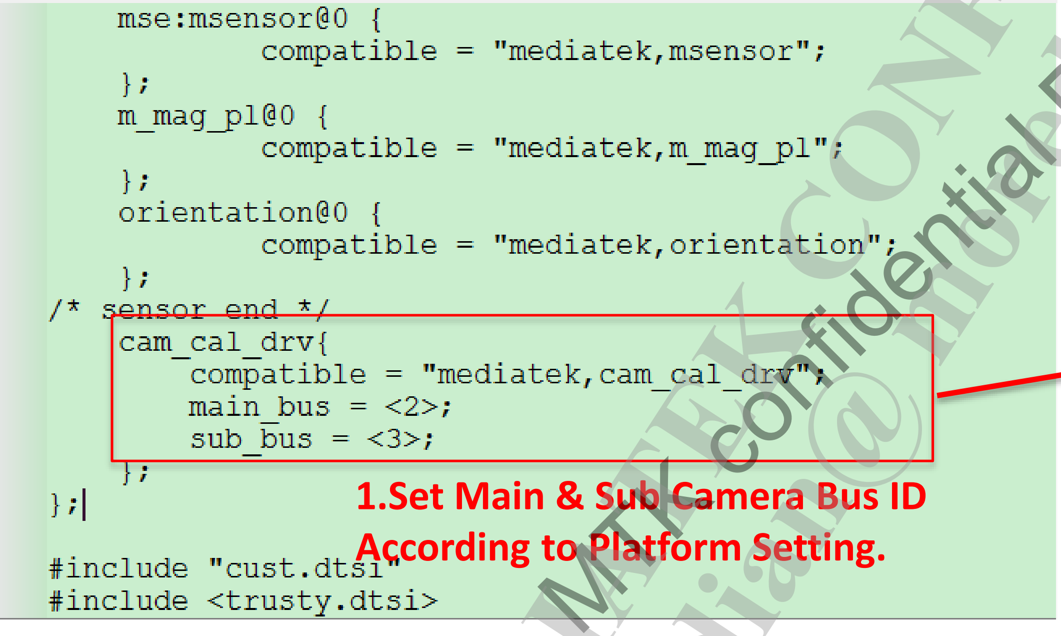
  kernel-3.18\drivers\misc\mediatek\cam_cal\src\cam_cal_drv.c
  kernel-4.4\drivers\misc\mediatek\cam_cal\src\eeprom_driver.c(MT6763)
  kernel-3.18\drivers\misc\mediatek\cam_cal\src\cam_cal_list.c
  kernel-3.18\drivers\misc\mediatek\cam_cal\src\common
  kernel-3.18\drivers\misc\mediatek\cam_cal\src\common\cat24c16\ cat24c16.c
- Hal modification:

  vendor\mediatek\proprietary\custom\mt6757\hal\imgsensor_src\sensorlist.cpp
  vendor\mediatek\proprietary\custom\mt6757\hal\imgsensor_src\camera_calibration_cam_cal.cpp

Step1
Check dtsi and defconfig

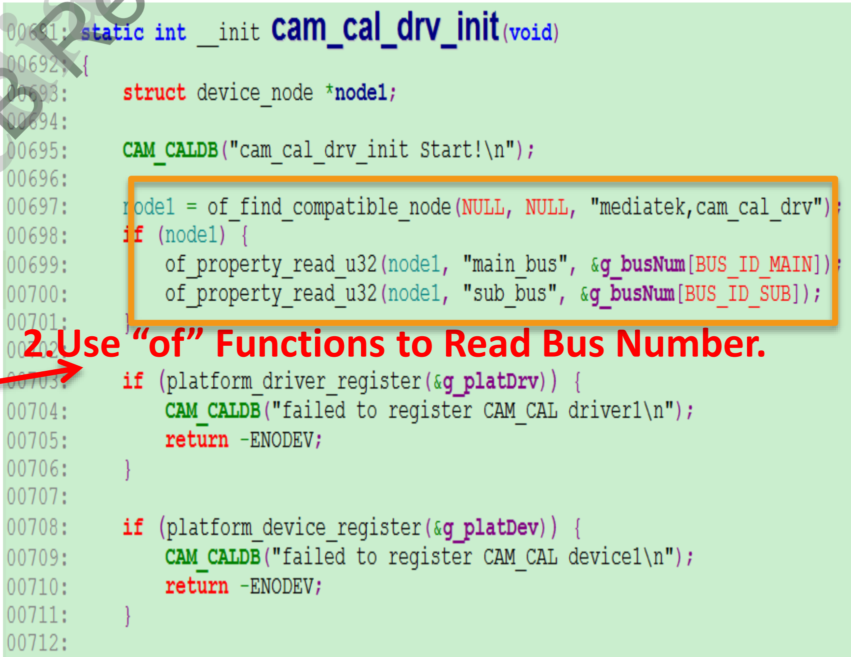
- Config Modify
- Kernel Modify
- Hal Modify
- Customizaton

- Make sure cam_cal_drv node defined in mt(platform).dtsi
  main_bus=<2> means, main OTP use bus i2c-2 to communicate



```
    mse:msensor@0 {
            compatible = "mediatek,msensor";
    };
    m_mag_pl@0 {
            compatible = "mediatek,m_mag_pl";
    };
    orientation@0 {
            compatible = "mediatek,orientation";
    };
/* sensor end */
    cam_cal_drv{
        compatible = "mediatek,cam_cal_drv";
        main_bus = <2>;
        sub_bus = <3>;
    };
};|

#include "cust.dtsi"
#include <trusty.dtsi>
```

1.Set Main & Sub Camera Bus ID According to Platform Setting.



```
static int __init cam_cal_drv_init(void)
{
    struct device_node *node1;

    CAM_CALDB("cam_cal_drv_init Start!\n");

    node1 = of_find_compatible_node(NULL, NULL, "mediatek,cam_cal_drv");
    if (node1) {
        of_property_read_u32(node1, "main_bus", &g_busNum[BUS_ID_MAIN]);
        of_property_read_u32(node1, "sub_bus", &g_busNum[BUS_ID_SUB]);
    }

    if (platform_driver_register(&g_platDrv)) {
        CAM_CALDB("failed to register CAM_CAL driver1\n");
        return -ENODEV;
    }

    if (platform_device_register(&g_platDev)) {
        CAM_CALDB("failed to register CAM_CAL device1\n");
        return -ENODEV;
    }

```

2.Use "of" Functions to Read Bus Number.

红框中的内容离顶端2个tab键,不要使用空格键！
否则编译不到.

cam_cal_drv.c

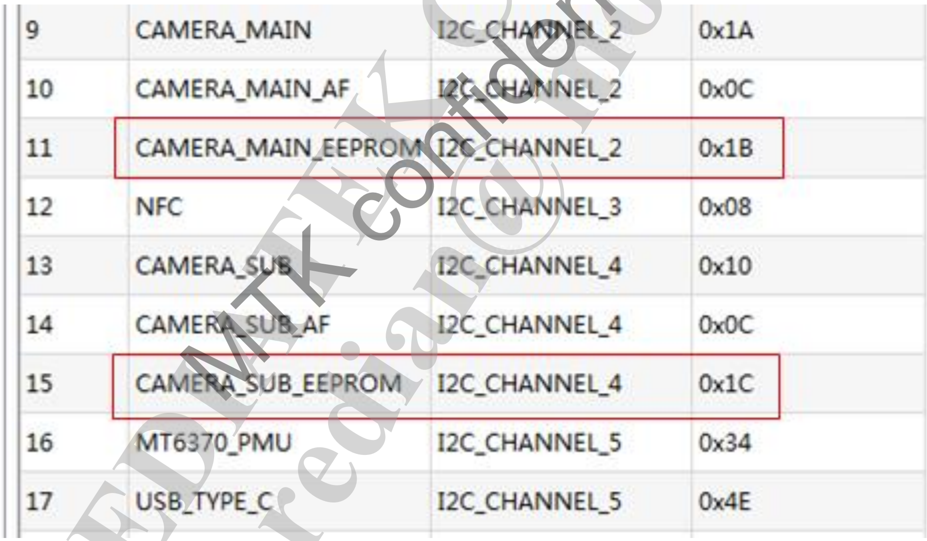
Step1
Check dtsi and defconfig

- Config Modify
- Kernel Modify
- Hal Modify
- Customizaton

- Configure i2c bus for main/sub OTP in $Project.dws

  kernel-4.4\drivers\misc\mediatek\dws\mt6763\$Project.dws (MT6763)

  1. configure i2c bus for main/sub eeprom separately
  2. i2c address should not be the same

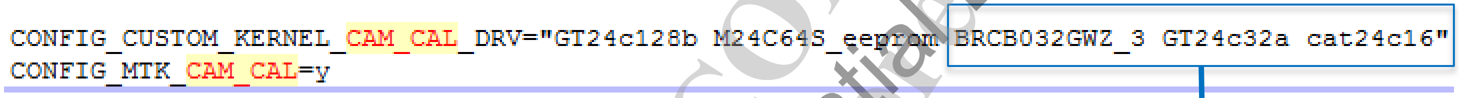


9	CAMERA_MAIN	I2C_CHANNEL_2	0x1A
10	CAMERA_MAIN_AF	I2C_CHANNEL_2	0x0C
11	CAMERA_MAIN_EEPROM	I2C_CHANNEL_2	0x1B
12	NFC	I2C_CHANNEL_3	0x08
13	CAMERA_SUB	I2C_CHANNEL_4	0x10
14	CAMERA_SUB_AF	I2C_CHANNEL_4	0x0C
15	CAMERA_SUB_EEPROM	I2C_CHANNEL_4	0x1C
16	MT6370_PMU	I2C_CHANNEL_5	0x34
17	USB_TYPE_C	I2C_CHANNEL_5	0x4E

eeprom_driver.c

Step1
Check dtsi and defconfig

- Configure defconfig file in kernel
  kernel-3.18\arch\arm64\configs\$Project_debug_defconfig
  kernel-3.18\arch\arm64\configs\$Project_defconfig
- Make sure CONFIG_MTK_CAM_CAL=y



```
CONFIG_CUSTOM_KERNEL_CAM_CAL_DRV="GT24c128b M24C64S_eeprom BRCB032GWZ_3 GT24c32a cat24c16"
CONFIG_MTK_CAM_CAL=y
```

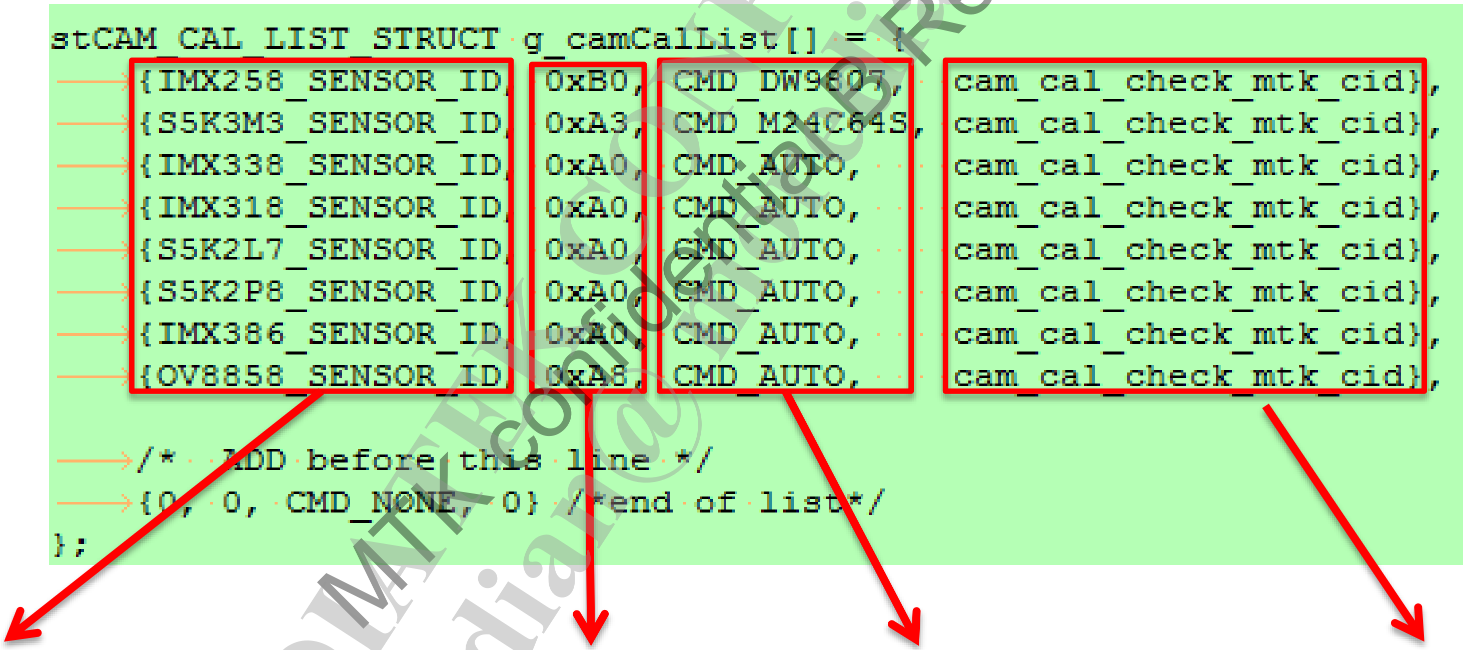
删掉会导致编译不过

Step2-1
OTP adapter modification

- Config Modify
- Kernel Modify
- Hal Modify
- Customizaton

- kernel-3.18\drivers\misc\mediatek\cam_cal\src\cam_cal_list.c

List 中参数代表的含义如下,一般只需要客制化sensor ID和 slave address.



```
stCAM_CAL_LIST_STRUCT g_camCalList[] = {
	{IMX258_SENSOR_ID, 0xB0, CMD_DW9807,  cam_cal_check_mtk_cid},
	{S5K3M3_SENSOR_ID, 0xA3, CMD_M24C64S, cam_cal_check_mtk_cid},
	{IMX338_SENSOR_ID, 0xA0, CMD_AUTO,    cam_cal_check_mtk_cid},
	{IMX318_SENSOR_ID, 0xA0, CMD_AUTO,    cam_cal_check_mtk_cid},
	{S5K2L7_SENSOR_ID, 0xA0, CMD_AUTO,    cam_cal_check_mtk_cid},
	{S5K2P8_SENSOR_ID, 0xA0, CMD_AUTO,    cam_cal_check_mtk_cid},
	{IMX386_SENSOR_ID, 0xA0, CMD_AUTO,    cam_cal_check_mtk_cid},
	{OV8858_SENSOR_ID, 0xA8, CMD_AUTO,    cam_cal_check_mtk_cid},

	/*  ADD before this line */
	{0, 0, CMD_NONE, 0} /*end of list*/
};
```

1. Sensor ID.
(Define in kd_imgsensor.h)

2. Slave Address
of EEPRom.

3. Command Type
of Supported
EEPRom .

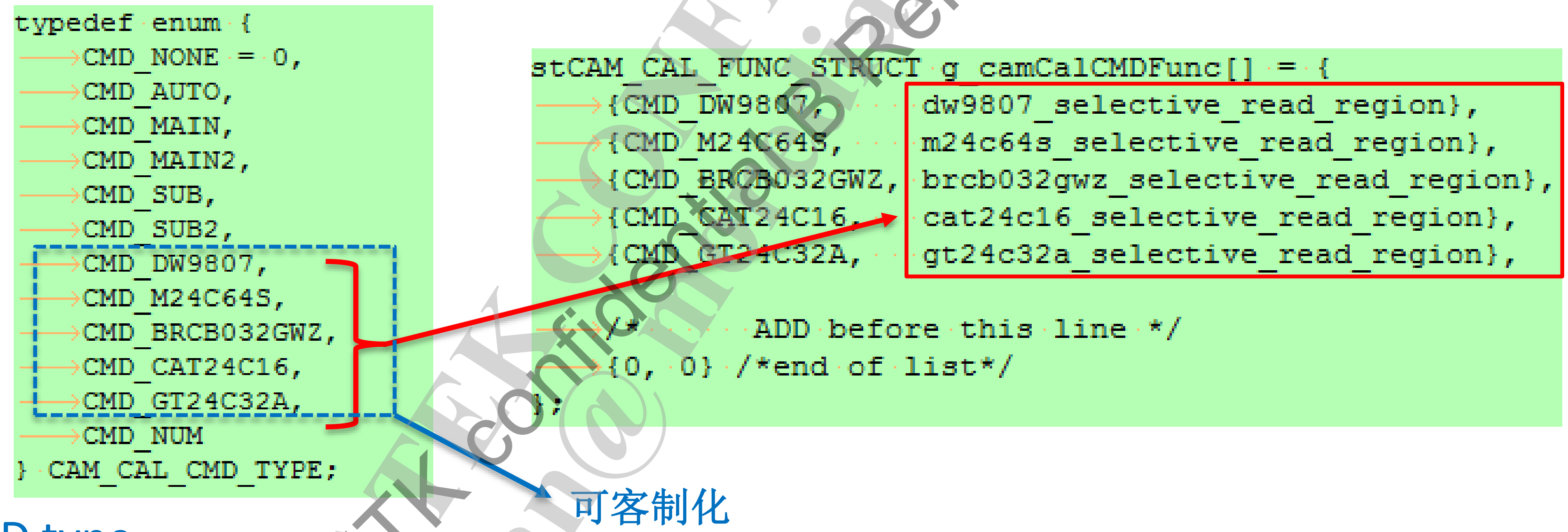
4. Check Function

Step2-1
OTP adapter modification

- kernel-3.18\drivers\misc\mediatek\cam_cal\src\cam_cal_list.c/.h

  1. cam_cal_list.h defines CMD type.

  2. g_camCalCMDFunc[] maps CMD type to relevant otp driver.



```
typedef enum {
    CMD_NONE = 0,
    CMD_AUTO,
    CMD_MAIN,
    CMD_MAIN2,
    CMD_SUB,
    CMD_SUB2,
    CMD_DW9807,
    CMD_M24C64S,
    CMD_BRCB032GWZ,
    CMD_CAT24C16,
    CMD_GT24C32A,
    CMD_NUM
} CAM_CAL_CMD_TYPE;
```

```
stCAM_CAL_FUNC_STRUCT g_camCalCMDFunc[] = {
    {CMD_DW9807,      dw9807_selective_read_region},
    {CMD_M24C64S,     m24c64s_selective_read_region},
    {CMD_BRCB032GWZ,  brcb032gwz_selective_read_region},
    {CMD_CAT24C16,    cat24c16_selective_read_region},
    {CMD_GT24C32A,    gt24c32a_selective_read_region},

    /*      ADD before this line */
    {0, 0} /*end of list*/
};
```

可客制化

- CMD type

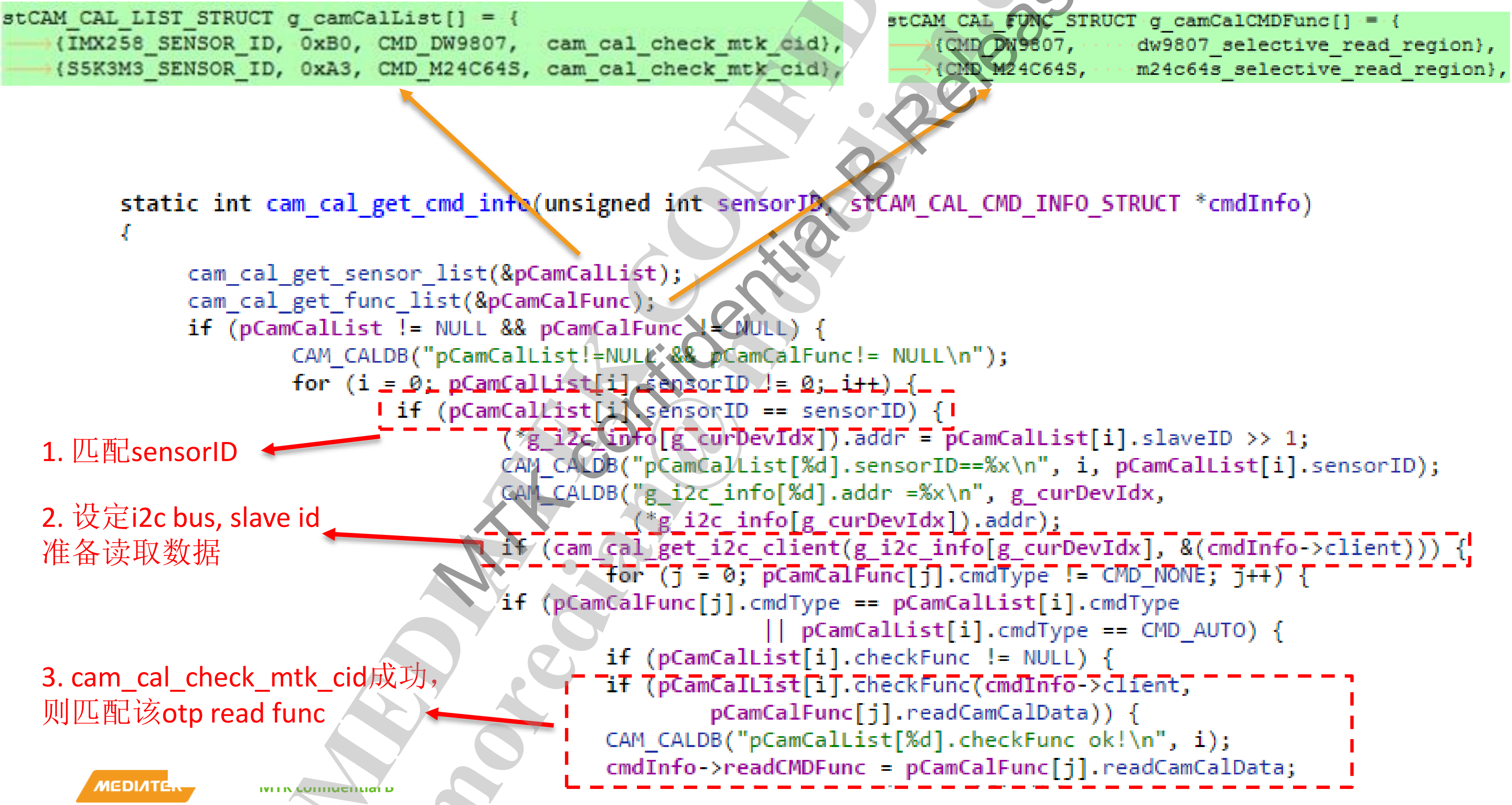
  CMD_AUTO: check function will search all otp driver to get the right driver.

  CMD_MAIN: using main OTP's i2c to commuincae.

  CMD_SUB: using sub OTP's i2c to commuincae.

Step2-1
OTP adapter modification

- Function cam_cal_get_cmd_info



```
stCAM_CAL_LIST_STRUCT g_camCalList[] = {
    {IMX258_SENSOR_ID, 0xB0, CMD_DW9807,  cam_cal_check_mtk_cid},
    {S5K3M3_SENSOR_ID, 0xA3, CMD_M24C64S, cam_cal_check_mtk_cid},
```

```
stCAM_CAL_FUNC_STRUCT g_camCalCMDFunc[] = {
    {CMD_DW9807,     dw9807_selective_read_region},
    {CMD_M24C64S,    m24c64s_selective_read_region},
```

```
static int cam_cal_get_cmd_info(unsigned int sensorID, stCAM_CAL_CMD_INFO_STRUCT *cmdInfo)
{
    cam_cal_get_sensor_list(&pCamCalList);
    cam_cal_get_func_list(&pCamCalFunc);
    if (pCamCalList != NULL && pCamCalFunc != NULL) {
          CAM_CALDB("pCamCalList!=NULL && pCamCalFunc!= NULL\n");
          for (i = 0; pCamCalList[i].sensorID != 0; i++) {
               if (pCamCalList[i].sensorID == sensorID) {
                      (*g_i2c_info[g_curDevIdx]).addr = pCamCalList[i].slaveID >> 1;
                      CAM_CALDB("pCamCalList[%d].sensorID==%x\n", i, pCamCalList[i].sensorID);
                      CAM_CALDB("g_i2c_info[%d].addr =%x\n", g_curDevIdx,
                             (*g_i2c_info[g_curDevIdx]).addr);
                      if (cam_cal_get_i2c_client(g_i2c_info[g_curDevIdx], &(cmdInfo->client))) {
                            for (j = 0; pCamCalFunc[j].cmdType != CMD_NONE; j++) {
                      if (pCamCalFunc[j].cmdType == pCamCalList[i].cmdType
                                     || pCamCalList[i].cmdType == CMD_AUTO) {
                            if (pCamCalList[i].checkFunc != NULL) {
                            if (pCamCalList[i].checkFunc(cmdInfo->client,
                                  pCamCalFunc[j].readCamCalData)) {
                            CAM_CALDB("pCamCalList[%d].checkFunc ok!\n", i);
                            cmdInfo->readCMDFunc = pCamCalFunc[j].readCamCalData;
```

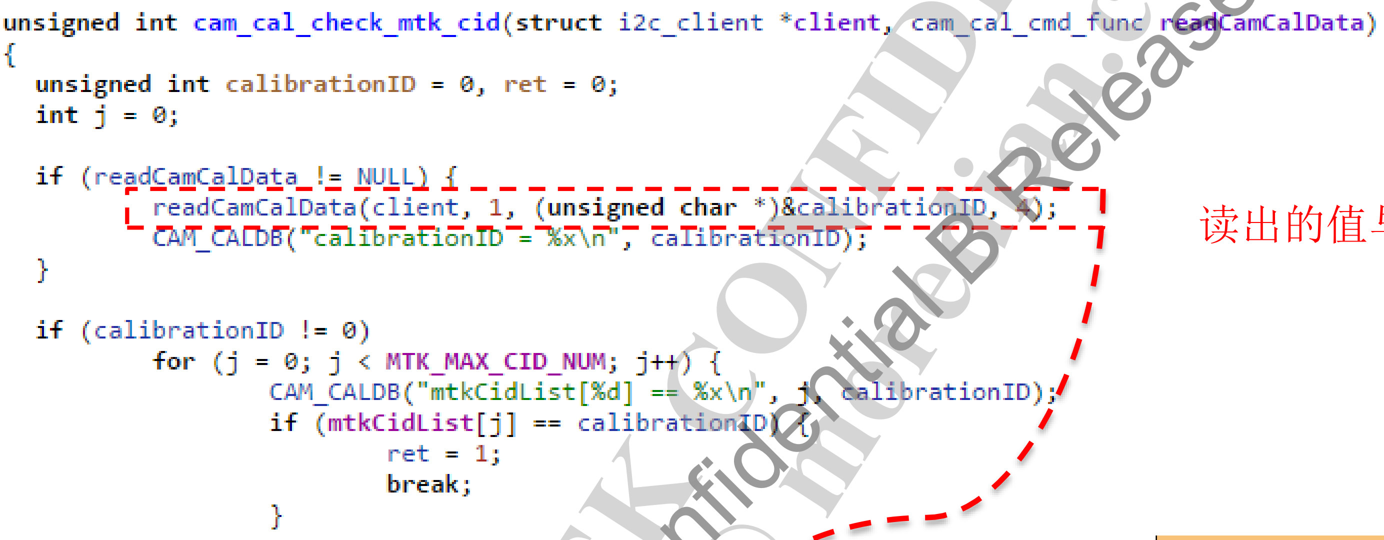
1. 匹配sensorID

2. 设定i2c bus, slave id
准备读取数据

3. cam_cal_check_mtk_cid成功，
则匹配该otp read func

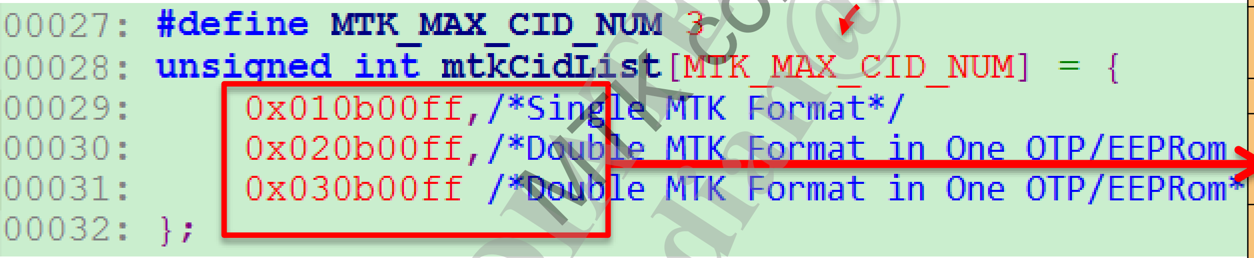
Step2-1
OTP adapter modification

- Check function cam_cal_check_mtk_cid



```
unsigned int cam_cal_check_mtk_cid(struct i2c_client *client, cam_cal_cmd_func readCamCalData)
{
    unsigned int calibrationID = 0, ret = 0;
    int j = 0;

    if (readCamCalData != NULL) {
            readCamCalData(client, 1, (unsigned char *)&calibrationID, 4);
            CAM_CALDB("calibrationID = %x\n", calibrationID);
    }

    if (calibrationID != 0)
            for (j = 0; j < MTK_MAX_CID_NUM; j++) {
                    CAM_CALDB("mtkCidList[%d] == %x\n", j, calibrationID);
                    if (mtkCidList[j] == calibrationID) {
                            ret = 1;
                            break;
                    }
```

读出的值与mtkCid比较



```
#define MTK_MAX_CID_NUM 3
unsigned int mtkCidList[MTK_MAX_CID_NUM] = {
        0x010b00ff,/*Single MTK Format*/
        0x020b00ff,/*Double MTK Format in One OTP/EEPRom
        0x030b00ff /*Double MTK Format in One OTP/EEPRom*
};
```

MTK Format	
0000	Flag 1 (check data correction)
0001 0003	Byte [0,1,2,3] = FF 00 0B 01 : calibration version
0005	Serial number (include vendor or project name, module number)
. .	…………………………

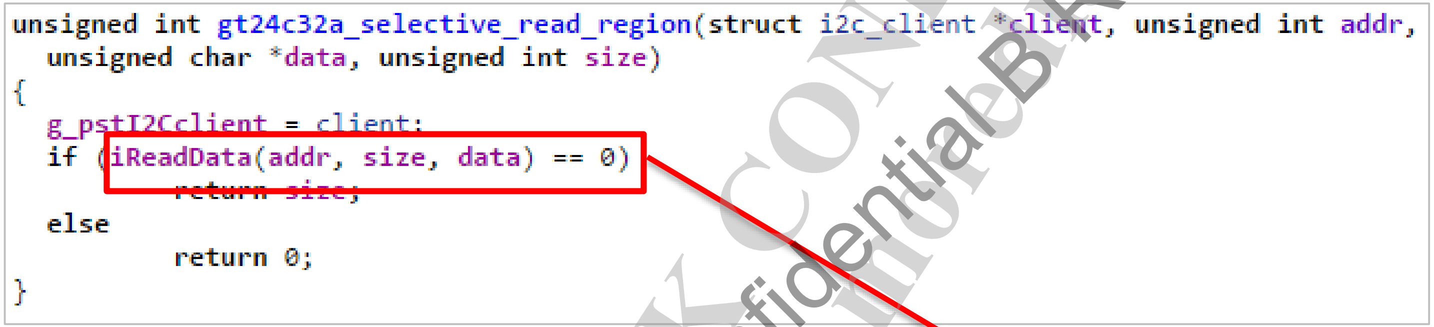
This check function will read the calibration ID in the OTP/EEPRom to verify the command function.

Step2-2
implement OTP driver

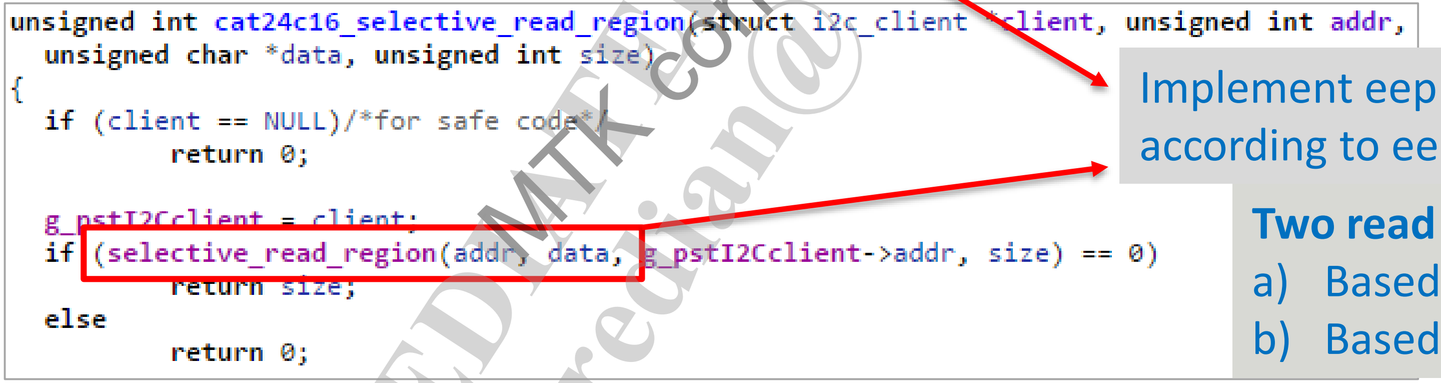
- Config Modify
- Kernel Modify
- Hal Modify
- Customizaton

- kernel-3.18\drivers\misc\mediatek\cam_cal\src\common\GT24c32a\GT24c32a.c

  a) implement **_selective_read_region() as the entrance of OTP driver.



```
unsigned int gt24c32a_selective_read_region(struct i2c_client *client, unsigned int addr,
  unsigned char *data, unsigned int size)
{
  g_pstI2Cclient = client;
  if (iReadData(addr, size, data) == 0)
          return size;
  else
          return 0;
}
```



```
unsigned int cat24c16_selective_read_region(struct i2c_client *client, unsigned int addr,
  unsigned char *data, unsigned int size)
{
  if (client == NULL)/*for safe code*/
          return 0;

  g_pstI2Cclient = client;
  if (selective_read_region(addr, data, g_pstI2Cclient->addr, size) == 0)
          return size;
  else
          return 0;
```

Implement eeprom read function according to eeprom IC datasheet

Two read format:

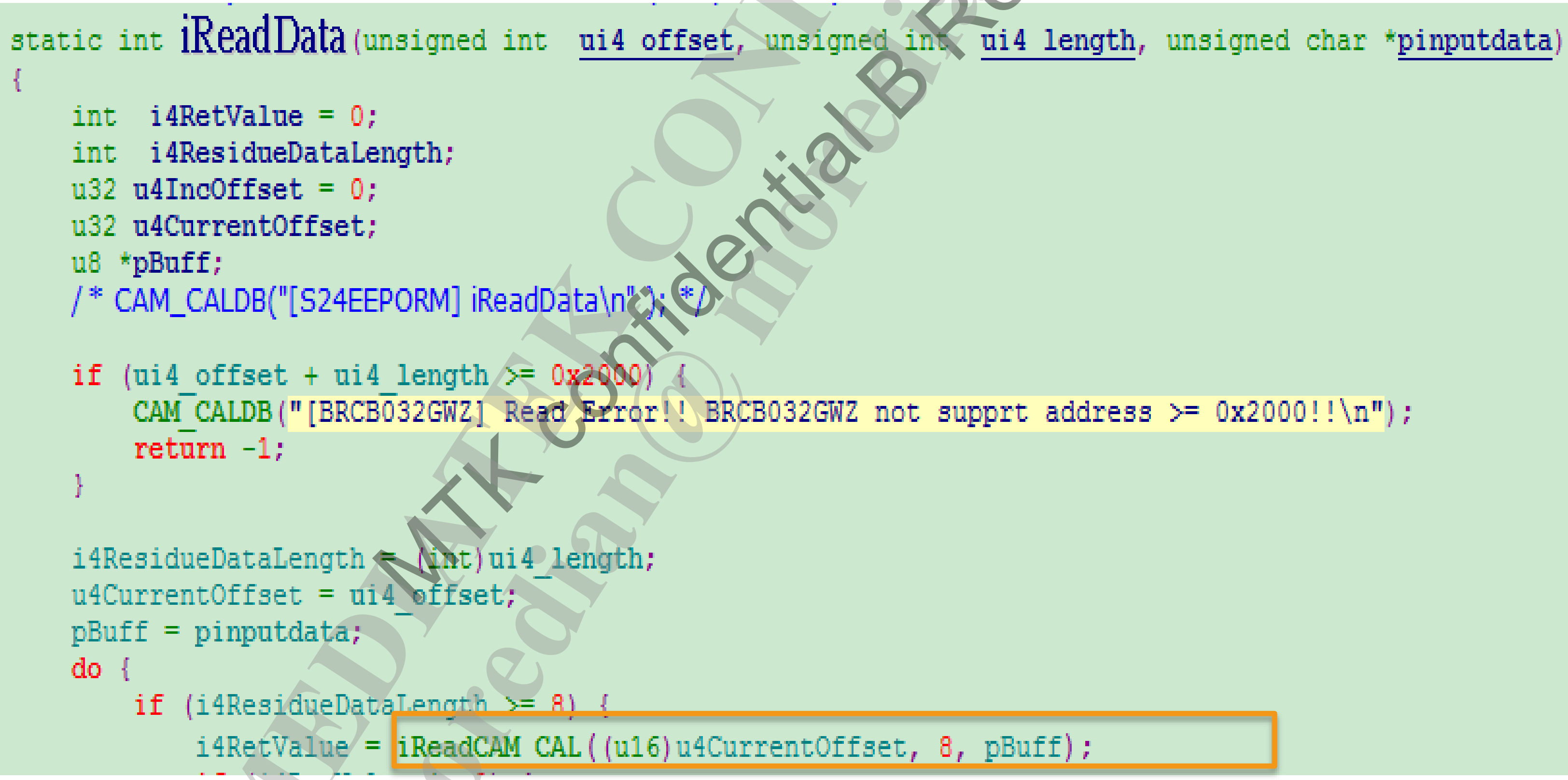
a) Based on address offset

b) Based on page read

Step2-2
implement OTP driver

- Type1(based on address offset)

kernel-3.18\drivers\misc\mediatek\cam_cal\src\common\BRCB032GWZ_3\BRCB032GWZ_3.c

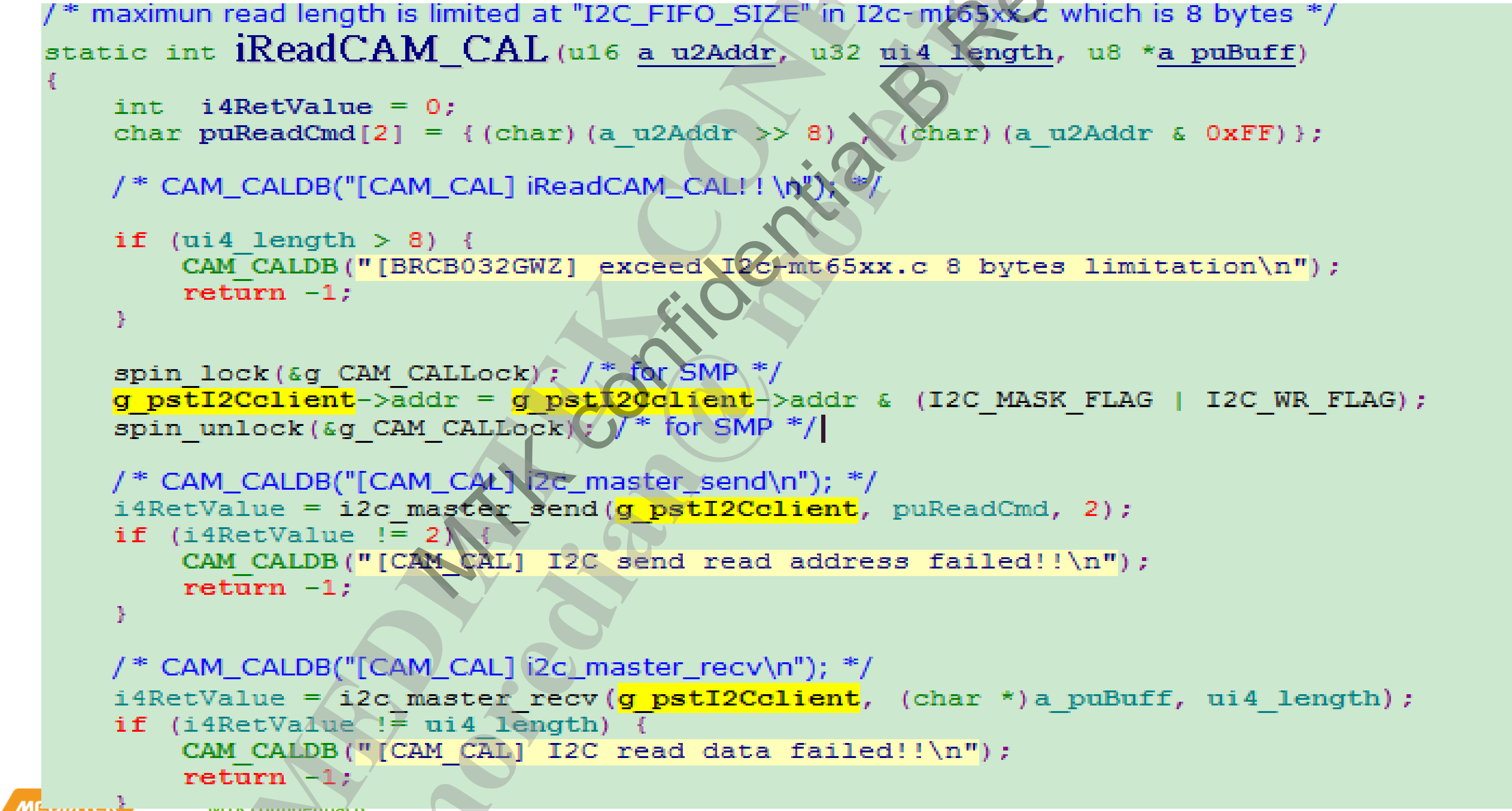


```
static int iReadData(unsigned int  ui4_offset, unsigned int  ui4_length, unsigned char *pinputdata)
{
    int  i4RetValue = 0;
    int  i4ResidueDataLength;
    u32 u4IncOffset = 0;
    u32 u4CurrentOffset;
    u8 *pBuff;
    /* CAM_CALDB("[S24EEPORM] iReadData\n"); */

    if (ui4_offset + ui4_length >= 0x2000) {
        CAM_CALDB("[BRCB032GWZ] Read Error!! BRCB032GWZ not supprt address >= 0x2000!!\n");
        return -1;
    }

    i4ResidueDataLength = (int)ui4_length;
    u4CurrentOffset = ui4_offset;
    pBuff = pinputdata;
    do {
        if (i4ResidueDataLength >= 8) {
            i4RetValue = iReadCAM_CAL((u16)u4CurrentOffset, 8, pBuff);
```

Step2-2
implement OTP driver

- Type1(based on address offset)

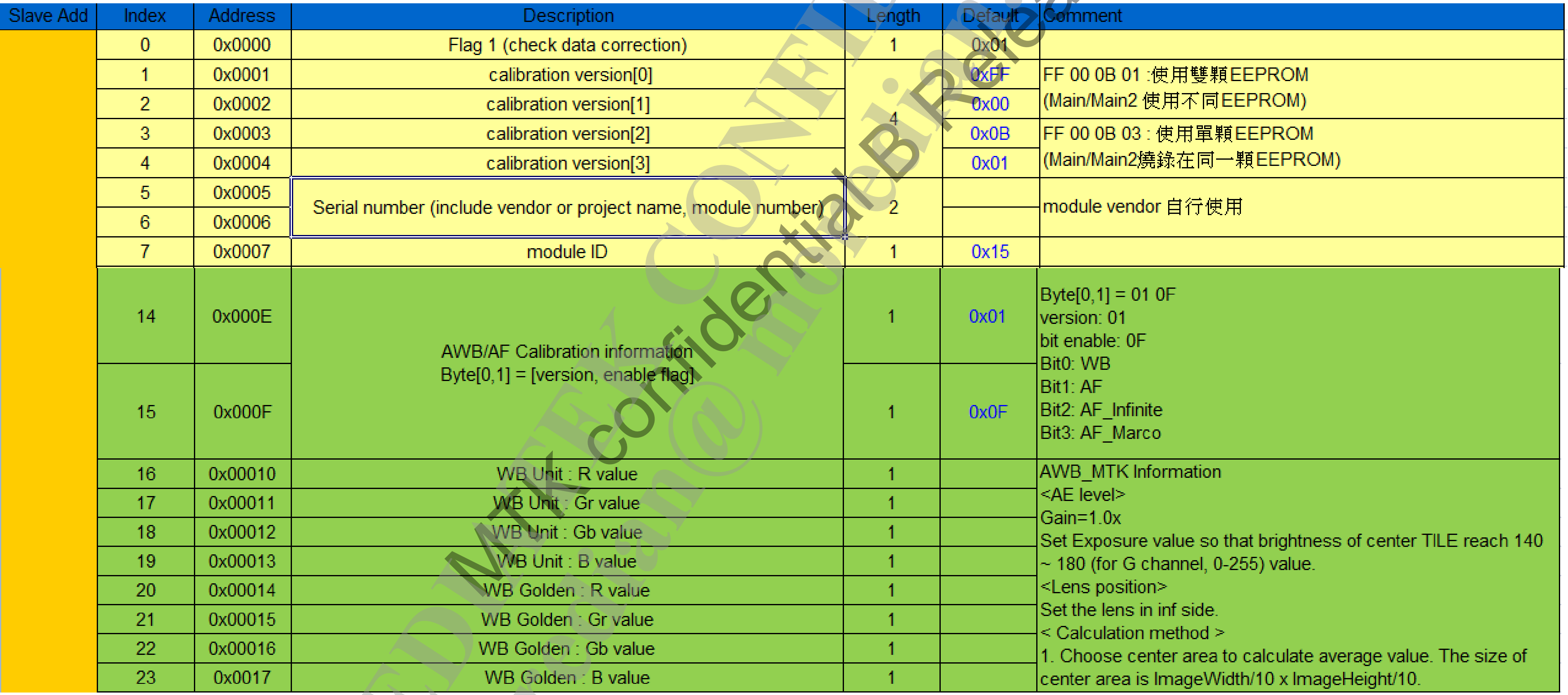
kernel-3.18\drivers\misc\mediatek\cam_cal\src\common\BRCB032GWZ_3\BRCB032GWZ_3.c



```
/* maximun read length is limited at "I2C_FIFO_SIZE" in I2c-mt65xx.c which is 8 bytes */
static int iReadCAM_CAL(u16 a_u2Addr, u32 ui4_length, u8 *a_puBuff)
{
    int  i4RetValue = 0;
    char puReadCmd[2] = {(char)(a_u2Addr >> 8) , (char)(a_u2Addr & 0xFF)};

    /* CAM_CALDB("[CAM_CAL] iReadCAM_CAL!! \n"); */

    if (ui4_length > 8) {
        CAM_CALDB("[BRCB032GWZ] exceed I2c-mt65xx.c 8 bytes limitation\n");
        return -1;
    }

    spin_lock(&g_CAM_CALLock); /* for SMP */
    g_pstI2Cclient->addr = g_pstI2Cclient->addr & (I2C_MASK_FLAG | I2C_WR_FLAG);
    spin_unlock(&g_CAM_CALLock); /* for SMP */

    /* CAM_CALDB("[CAM_CAL] i2c_master_send\n"); */
    i4RetValue = i2c_master_send(g_pstI2Cclient, puReadCmd, 2);
    if (i4RetValue != 2) {
        CAM_CALDB("[CAM_CAL] I2C send read address failed!!\n");
        return -1;
    }

    /* CAM_CALDB("[CAM_CAL] i2c_master_recv\n"); */
    i4RetValue = i2c_master_recv(g_pstI2Cclient, (char *)a_puBuff, ui4_length);
    if (i4RetValue != ui4_length) {
        CAM_CALDB("[CAM_CAL] I2C read data failed!!\n");
        return -1;
    }
```

Step2-2
implement OTP driver

- Type1 layout

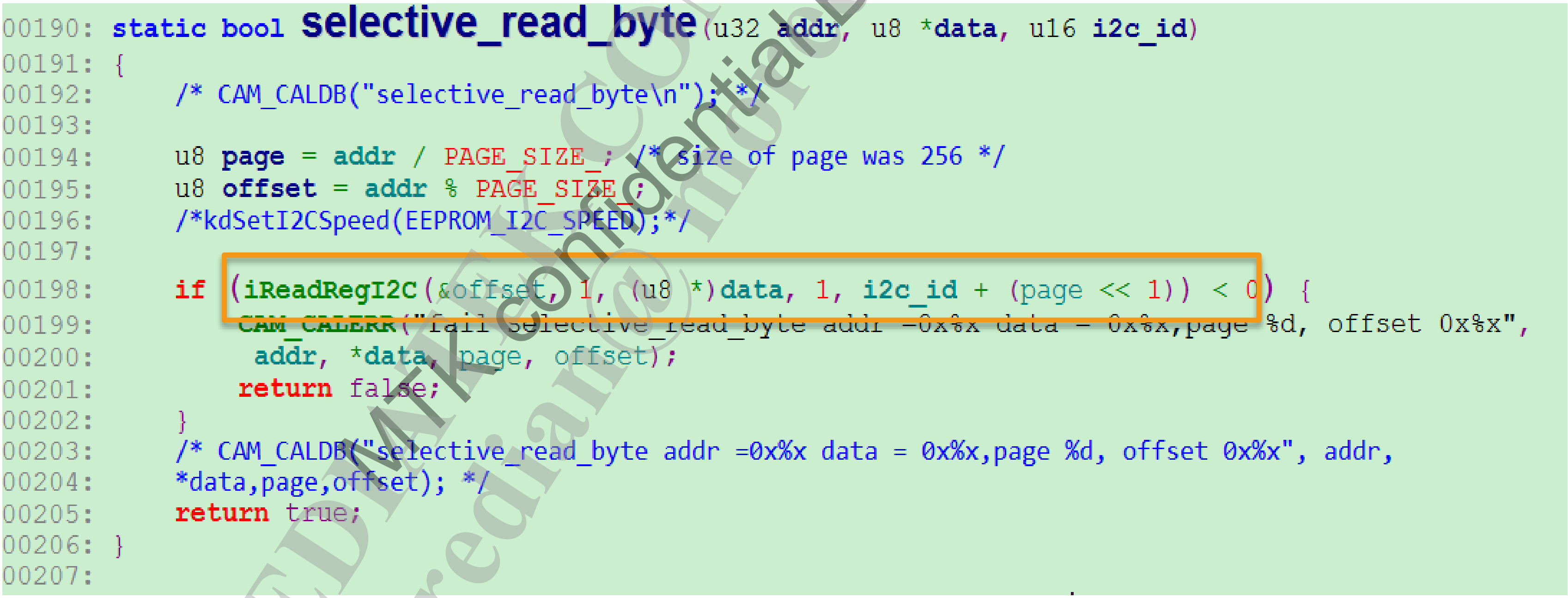


Slave Add	Index	Address	Description	Length	Default	Comment
	0	0x0000	Flag 1 (check data correction)	1	0x01	
	1	0x0001	calibration version[0]	4	0xFF	FF 00 0B 01 :使用雙顆EEPROM (Main/Main2 使用不同EEPROM)
	2	0x0002	calibration version[1]		0x00	
	3	0x0003	calibration version[2]		0x0B	FF 00 0B 03 : 使用單顆EEPROM (Main/Main2燒錄在同一顆EEPROM)
	4	0x0004	calibration version[3]		0x01	
	5	0x0005	Serial number (include vendor or project name, module number)	2		module vendor 自行使用
	6	0x0006				
	7	0x0007	module ID	1	0x15	
	14	0x000E	AWB/AF Calibration information Byte[0,1] = [version, enable flag]	1	0x01	Byte[0,1] = 01 0F version: 01 bit enable: 0F Bit0: WB Bit1: AF Bit2: AF_Infinite Bit3: AF_Marco
	15	0x000F		1	0x0F	
	16	0x00010	WB Unit : R value	1		AWB_MTK Information <AE level> Gain=1.0x Set Exposure value so that brightness of center TILE reach 140 ~ 180 (for G channel, 0-255) value. <Lens position> Set the lens in inf side. < Calculation method > 1. Choose center area to calculate average value. The size of center area is ImageWidth/10 x ImageHeight/10.
	17	0x00011	WB Unit : Gr value	1		
	18	0x00012	WB Unit : Gb value	1		
	19	0x00013	WB Unit : B value	1		
	20	0x00014	WB Golden : R value	1		
	21	0x00015	WB Golden : Gr value	1		
	22	0x00016	WB Golden : Gb value	1		
	23	0x0017	WB Golden : B value	1		

Step2-2
implement OTP driver

- Type2(based on page read)

kernel-3.18\drivers\misc\mediatek\cam_cal\src\common\cat24c16\cat24c16.c

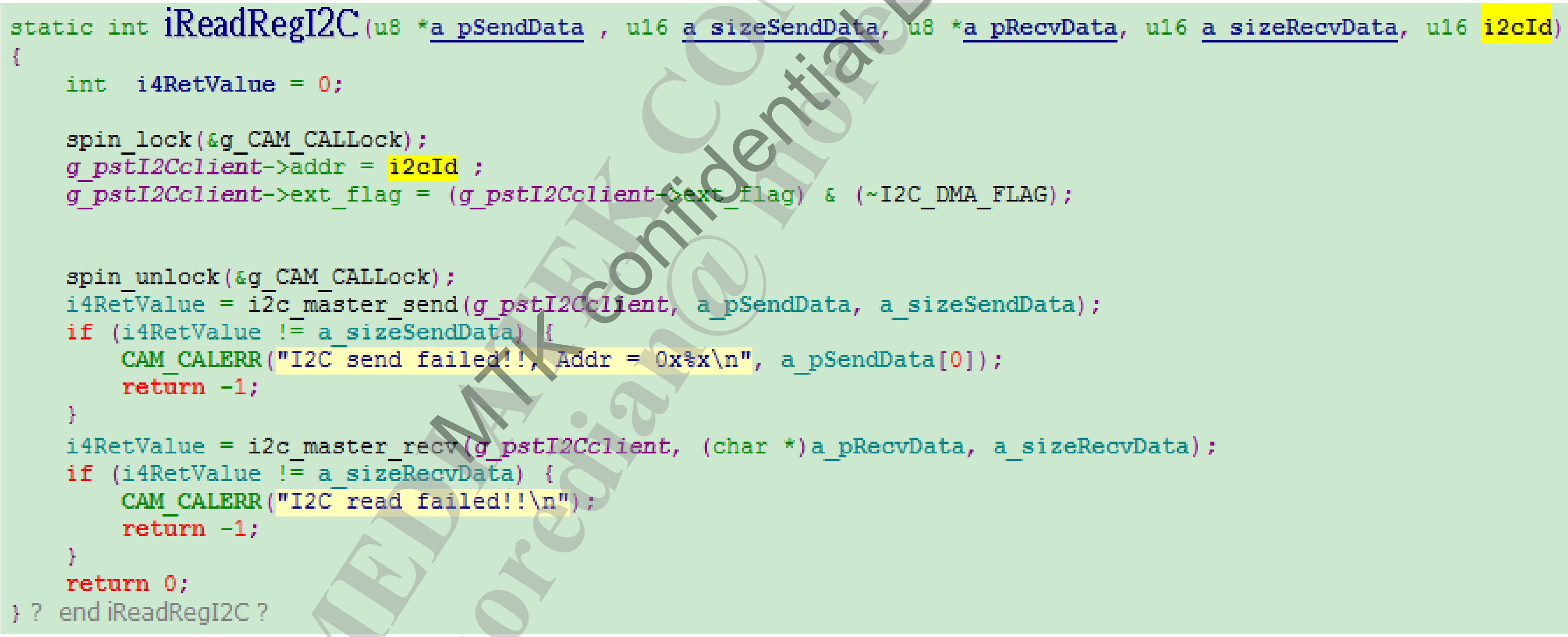


```
static bool selective_read_byte(u32 addr, u8 *data, u16 i2c_id)
{
	/* CAM_CALDB("selective_read_byte\n"); */

	u8 page = addr / PAGE_SIZE_; /* size of page was 256 */
	u8 offset = addr % PAGE_SIZE_;
	/*kdSetI2CSpeed(EEPROM_I2C_SPEED);*/

	if (iReadRegI2C(&offset, 1, (u8 *)data, 1, i2c_id + (page << 1)) < 0) {
		CAM_CALERR("fail selective_read_byte addr =0x%x data = 0x%x,page %d, offset 0x%x",
		 addr, *data, page, offset);
		return false;
	}
	/* CAM_CALDB("selective_read_byte addr =0x%x data = 0x%x,page %d, offset 0x%x", addr,
	*data,page,offset); */
	return true;
}
```

Step2-2
implement OTP driver

- Type2(based on page read)

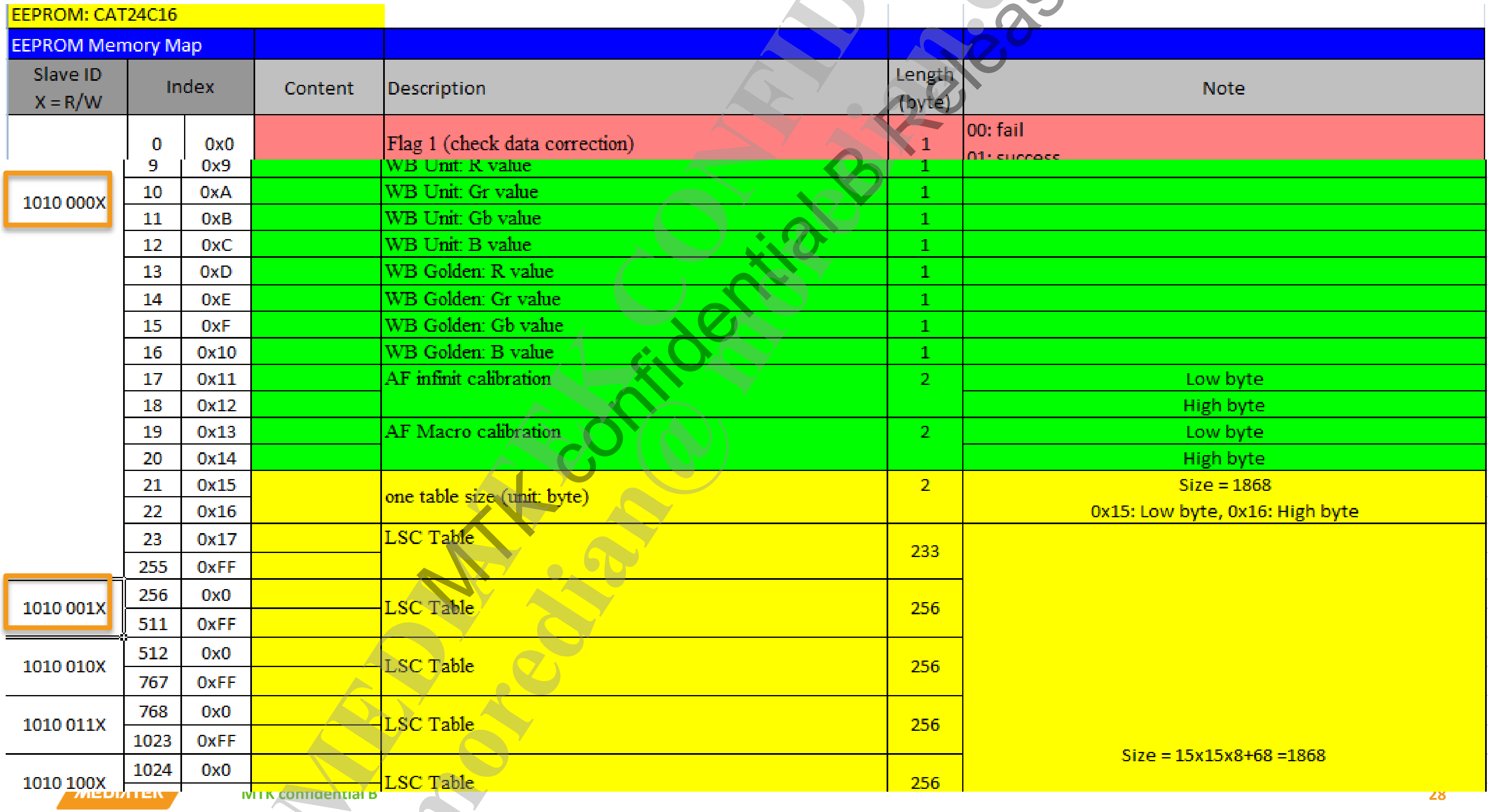
kernel-3.18\drivers\misc\mediatek\cam_cal\src\common\cat24c16\cat24c16.c



```
static int iReadRegI2C(u8 *a_pSendData , u16 a_sizeSendData, u8 *a_pRecvData, u16 a_sizeRecvData, u16 i2cId)
{
    int  i4RetValue = 0;

    spin_lock(&g_CAM_CALLock);
    g_pstI2Cclient->addr = i2cId ;
    g_pstI2Cclient->ext_flag = (g_pstI2Cclient->ext_flag) & (~I2C_DMA_FLAG);


    spin_unlock(&g_CAM_CALLock);
    i4RetValue = i2c_master_send(g_pstI2Cclient, a_pSendData, a_sizeSendData);
    if (i4RetValue != a_sizeSendData) {
        CAM_CALERR("I2C send failed!!, Addr = 0x%x\n", a_pSendData[0]);
        return -1;
    }
    i4RetValue = i2c_master_recv(g_pstI2Cclient, (char *)a_pRecvData, a_sizeRecvData);
    if (i4RetValue != a_sizeRecvData) {
        CAM_CALERR("I2C read failed!!\n");
        return -1;
    }
    return 0;
} ?  end iReadRegI2C ?
```

Step2-2
implement OTP driver

- Type2 layout



EEPROM: CAT24C16

EEPROM Memory Map

Slave ID X = R/W	Index		Content	Description	Length (byte)	Note
	0	0x0		Flag 1 (check data correction)	1	00: fail 01: success
1010 000X	9	0x9		WB Unit: R value	1	
	10	0xA		WB Unit: Gr value	1	
	11	0xB		WB Unit: Gb value	1	
	12	0xC		WB Unit: B value	1	
	13	0xD		WB Golden: R value	1	
	14	0xE		WB Golden: Gr value	1	
	15	0xF		WB Golden: Gb value	1	
	16	0x10		WB Golden: B value	1	
	17	0x11		AF infinit calibration	2	Low byte
	18	0x12				High byte
	19	0x13		AF Macro calibration	2	Low byte
	20	0x14				High byte
	21	0x15		one table size (unit: byte)	2	Size = 1868 0x15: Low byte, 0x16: High byte
	22	0x16				
	23	0x17		LSC Table	233	Size = 15x15x8+68 =1868
	255	0xFF				
1010 001X	256	0x0		LSC Table	256	
	511	0xFF				
1010 010X	512	0x0		LSC Table	256	
	767	0xFF				
1010 011X	768	0x0		LSC Table	256	
	1023	0xFF				
1010 100X	1024	0x0		LSC Table	256	

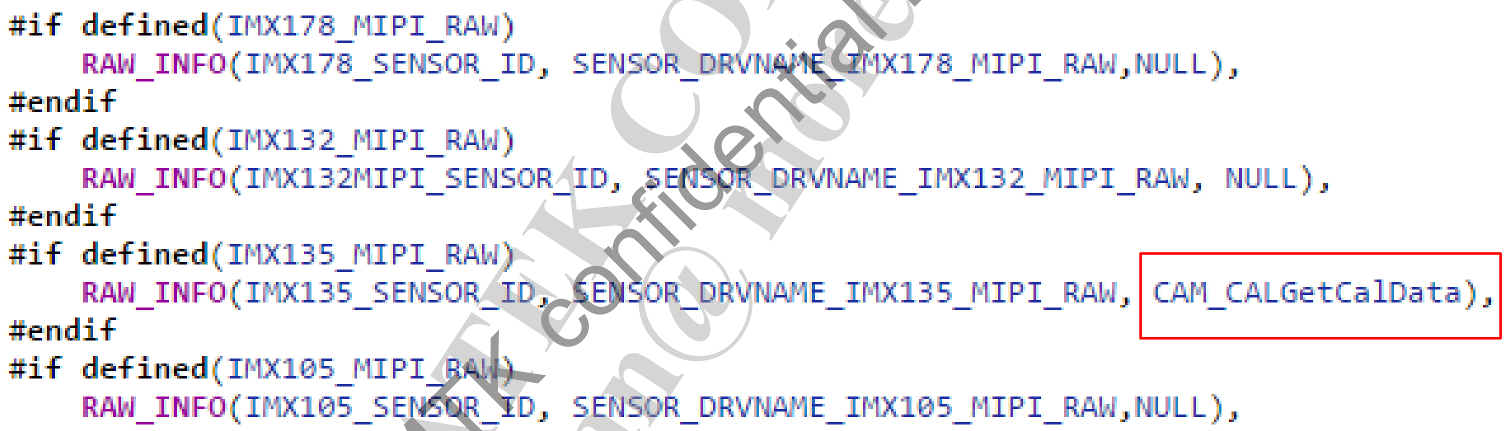
Step3
OTP Hal modification

- Config Modify
- Kernel Modify
- Hal Modify
- Customizaton

- vendor/mediatek/proprietary/custom/mt6757/hal/imgsensor_src/sensorlist.cpp

  a) NULL means don’t support OTP

  b) use CAM_CALGetCalData to support OTP

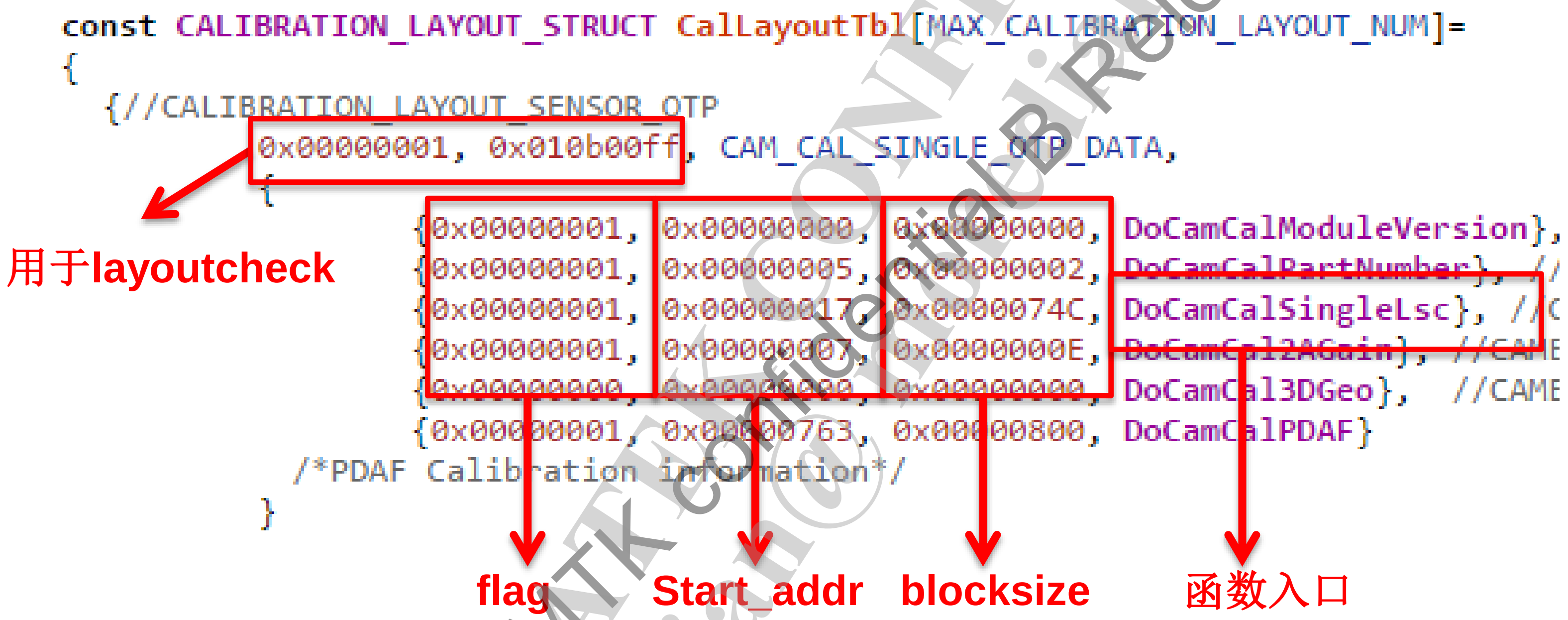


```
#if defined(IMX178_MIPI_RAW)
    RAW_INFO(IMX178_SENSOR_ID, SENSOR_DRVNAME_IMX178_MIPI_RAW,NULL),
#endif
#if defined(IMX132_MIPI_RAW)
    RAW_INFO(IMX132MIPI_SENSOR_ID, SENSOR_DRVNAME_IMX132_MIPI_RAW, NULL),
#endif
#if defined(IMX135_MIPI_RAW)
    RAW_INFO(IMX135_SENSOR_ID, SENSOR_DRVNAME_IMX135_MIPI_RAW, CAM_CALGetCalData),
#endif
#if defined(IMX105_MIPI_RAW)
    RAW_INFO(IMX105_SENSOR_ID, SENSOR_DRVNAME_IMX105_MIPI_RAW,NULL),
```

Step3
OTP Hal modification

根据eeprom客制化data起始地址和大小

- vendor/mediatek/proprietary/custom/mt6757/hal/imgsensor_src/camera_calibration_cam_cal.cpp:



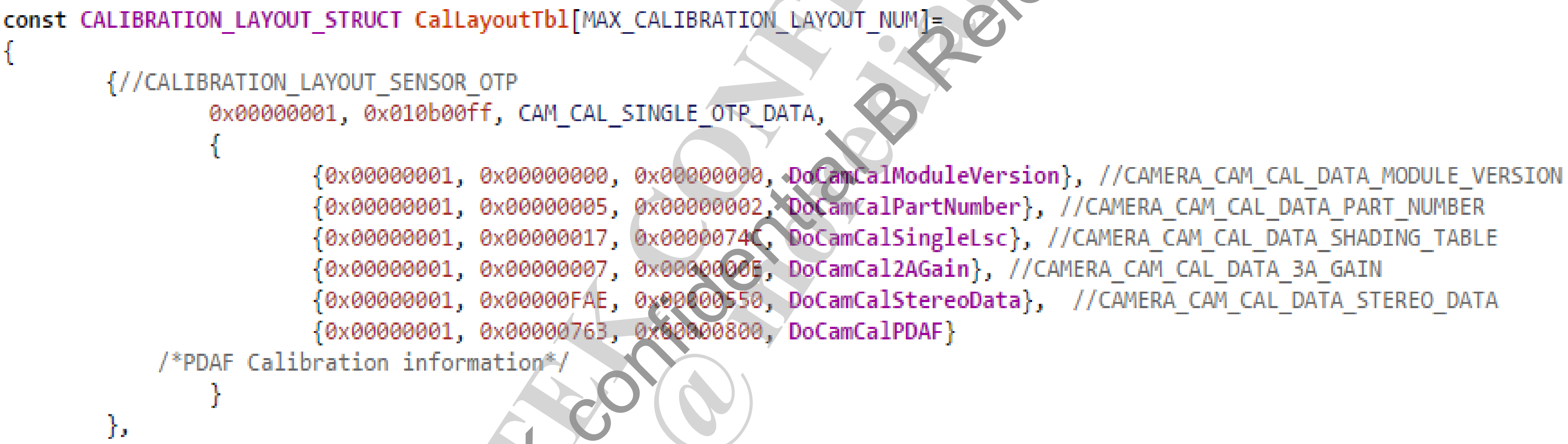
```
const CALIBRATION_LAYOUT_STRUCT CalLayoutTbl[MAX_CALIBRATION_LAYOUT_NUM]=
{
   {//CALIBRATION_LAYOUT_SENSOR_OTP
         0x00000001, 0x010b00ff, CAM_CAL_SINGLE_OTP_DATA,
         {
                  {0x00000001, 0x00000000, 0x00000000, DoCamCalModuleVersion},
                  {0x00000001, 0x00000005, 0x00000002, DoCamCalPartNumber}, //
                  {0x00000001, 0x00000017, 0x0000074C, DoCamCalSingleLsc}, //C
                  {0x00000001, 0x00000007, 0x0000000E, DoCamCal2AGain}, //CAME
                  {0x00000000, 0x00000000, 0x00000000, DoCamCal3DGeo},  //CAME
                  {0x00000001, 0x00000763, 0x00000800, DoCamCalPDAF}
         /*PDAF Calibration information*/
         }
```

根据实际情况，不需要用到的入口函数，其相应flag可置为0.

Step3
OTP Hal modification

Explanation of API in CalLayoutTbl

- vendor/mediatek/proprietary/custom/mt6763/hal/imgsensor_src/camera_calibration_cam_cal.cpp:



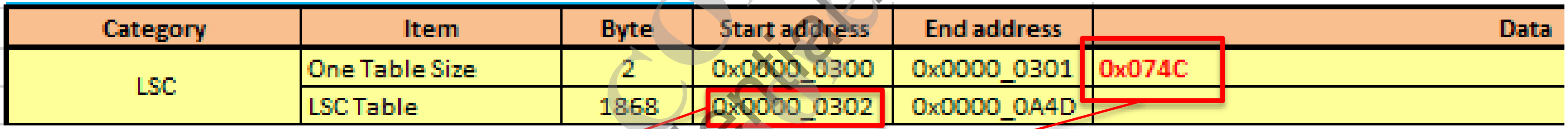
```
const CALIBRATION_LAYOUT_STRUCT CalLayoutTbl[MAX_CALIBRATION_LAYOUT_NUM]=
{
        {//CALIBRATION_LAYOUT_SENSOR_OTP
                0x00000001, 0x010b00ff, CAM_CAL_SINGLE_OTP_DATA,
                {
                        {0x00000001, 0x00000000, 0x00000000, DoCamCalModuleVersion}, //CAMERA_CAM_CAL_DATA_MODULE_VERSION
                        {0x00000001, 0x00000005, 0x00000002, DoCamCalPartNumber}, //CAMERA_CAM_CAL_DATA_PART_NUMBER
                        {0x00000001, 0x00000017, 0x0000074C, DoCamCalSingleLsc}, //CAMERA_CAM_CAL_DATA_SHADING_TABLE
                        {0x00000001, 0x00000007, 0x0000000E, DoCamCal2AGain}, //CAMERA_CAM_CAL_DATA_3A_GAIN
                        {0x00000001, 0x00000FAE, 0x00000550, DoCamCalStereoData},  //CAMERA_CAM_CAL_DATA_STEREO_DATA
                        {0x00000001, 0x00000763, 0x00000800, DoCamCalPDAF}
            /*PDAF Calibration information*/
                }
        },
```

- DoCamCalSingleLsc：读取lens shading相关的calibration data
- DoCamCal2AGain：读取AF/AWB相关的calibration data
- DoCamCalStereoData：读取双摄depth map相关的calibration data
- DoCamCalPDAF：读取PDAF相关的calibration data (from MT6757)

Step3
OTP Hal modification

根据eeprom客制化data起始地址和大小

例如eeprom中lsc的烧录为:

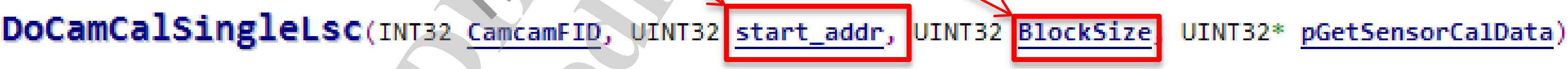


Category	Item	Byte	Start address	End address	Data
LSC	One Table Size	2	0x0000_0300	0x0000_0301	0x074C
	LSC Table	1868	0x0000_0302	0x0000_0A4D	

那么应该配置为

{0x00000001, 0x00000302, 0x0000074C, DoCamCalSingleLsc},

最终这些参数会作为 DoCamCalSingleLsc( )的形参,通过ioctl传递给kernel space:

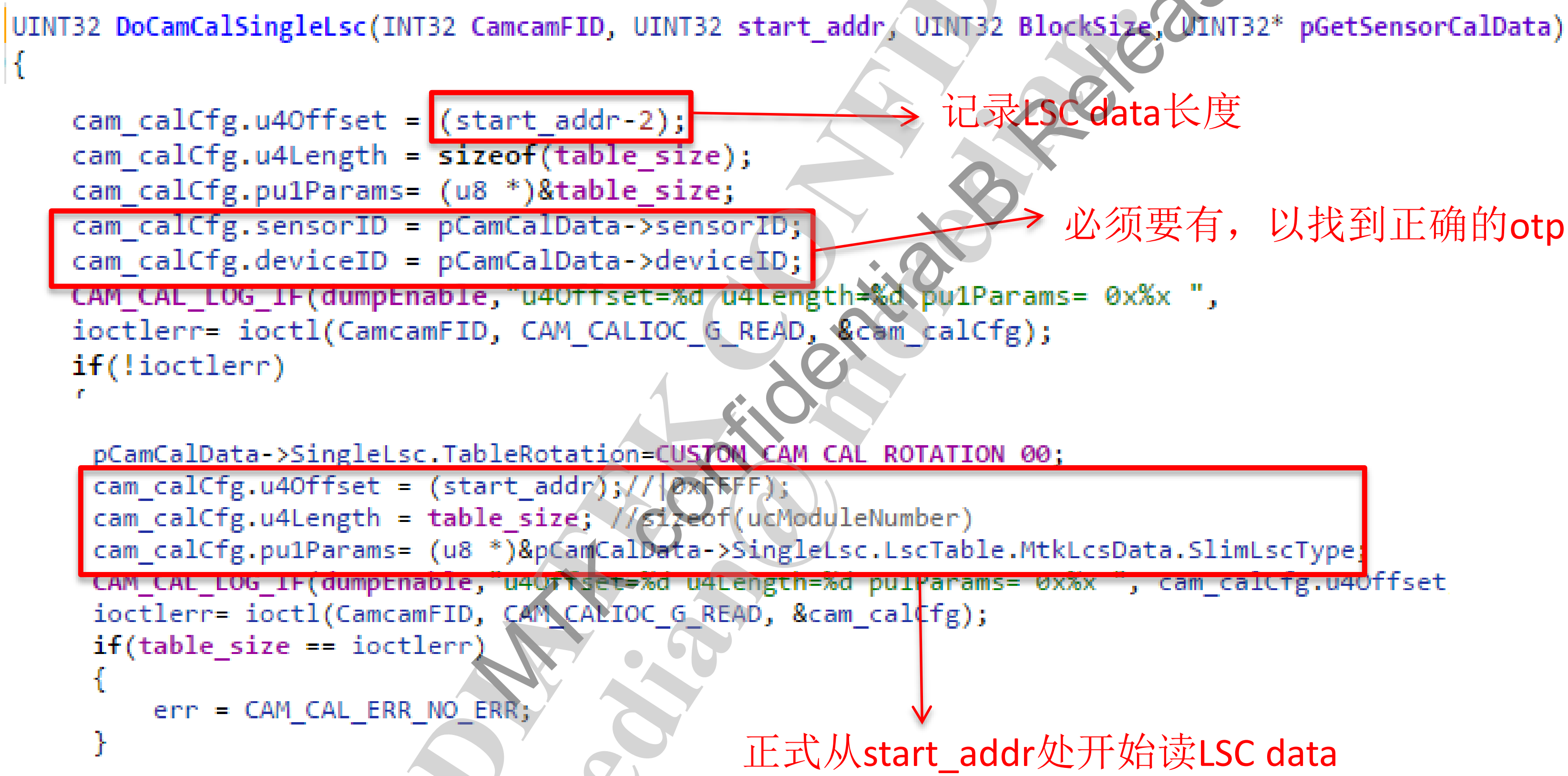


```
DoCamCalSingleLsc(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize, UINT32* pGetSensorCalData)
```

DoCamCal2AGain()等函数同理配置.

Step3
OTP Hal modification

- Function DoCamCalSingleLsc



```
UINT32 DoCamCalSingleLsc(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize, UINT32* pGetSensorCalData)
{

    cam_calCfg.u4Offset = (start_addr-2);
    cam_calCfg.u4Length = sizeof(table_size);
    cam_calCfg.pu1Params= (u8 *)&table_size;
    cam_calCfg.sensorID = pCamCalData->sensorID;
    cam_calCfg.deviceID = pCamCalData->deviceID;
    CAM_CAL_LOG_IF(dumpEnable,"u4Offset=%d u4Length=%d pu1Params= 0x%x ",
    ioctlerr= ioctl(CamcamFID, CAM_CALIOC_G_READ, &cam_calCfg);
    if(!ioctlerr)
    {

     pCamCalData->SingleLsc.TableRotation=CUSTOM_CAM_CAL_ROTATION_00;
     cam_calCfg.u4Offset = (start_addr);//|0xFFFF);
     cam_calCfg.u4Length = table_size; //sizeof(ucModuleNumber)
     cam_calCfg.pu1Params= (u8 *)&pCamCalData->SingleLsc.LscTable.MtkLcsData.SlimLscType;
     CAM_CAL_LOG_IF(dumpEnable,"u4Offset=%d u4Length=%d pu1Params= 0x%x ", cam_calCfg.u4Offset
     ioctlerr= ioctl(CamcamFID, CAM_CALIOC_G_READ, &cam_calCfg);
     if(table_size == ioctlerr)
     {
         err = CAM_CAL_ERR_NO_ERR;
     }
```

记录LSC data长度

必须要有，以找到正确的otp read func

正式从start_addr处开始读LSC data

DoCamCal2AGain()等函数同理.

Step3
OTP Hal modification

配置pixel ID:

如果要进行lsc calibration需要在camera_isp_lsc_xxxmipiraw.h中配置pixel id.通过和module厂商确定烧录的OTP的first pixel,然后修改:

vendor\mediatek\proprietary\custom\[platform]\hal\imgsensor\xxx_mipi_raw\camera_isp_lsc_xxxmipiraw.h

```
SensorGoldenCalTable:{   // SensorGoldenCalTable
        PixId:    3,       //0,1,2,3: B,Gb,Gr,R
        SlimLscType:    0,
        Width:    0,
```

Step4
OTP customization

- Config Modify
- Kernel Modify
- Hal Modify
- Customizaton

Check EEPROM LAYOUT

1) Make sure eeprom was burned based on MTK specification

2) The should be four bytes burned in eeprom: FF 00 0B 01

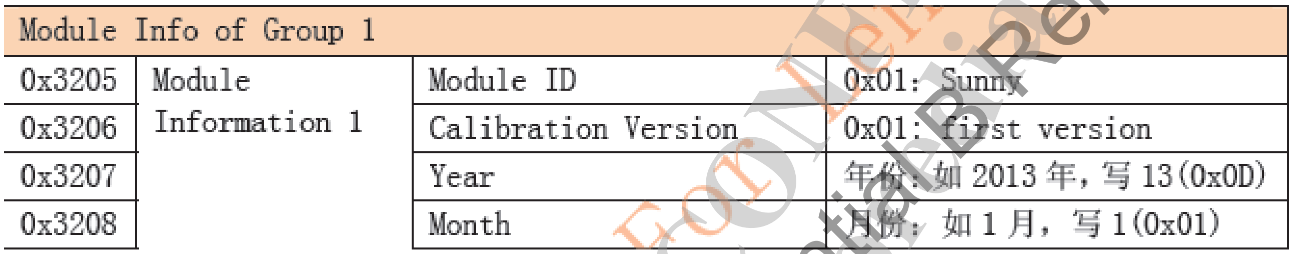
MTK Format	
0000	Flag 1 (check data correction)
0001	Byte [0,1,2,3] = FF 00 0B 01 : calibration version
0003	
0005	Serial number (include vendor or project name, module number)
. .	....

请注意，烧录的data格式需要按照MTK格式烧录，否则MTK将不予support

Step4
OTP customization

- calibrationID customization

如果layout不是按照MTK规范烧录的,也可通过客制化来修改:



Module Info of Group 1			
0x3205	Module Information 1	Module ID	0x01: Sunny
0x3206		Calibration Version	0x01: first version
0x3207		Year	年份: 如 2013 年, 写 13(0x0D)
0x3208		Month	月份: 如 1 月, 写 1(0x01)

(1)从layout中找一个固定的值, 例如0x3205,里面烧录的值一直为0x01.

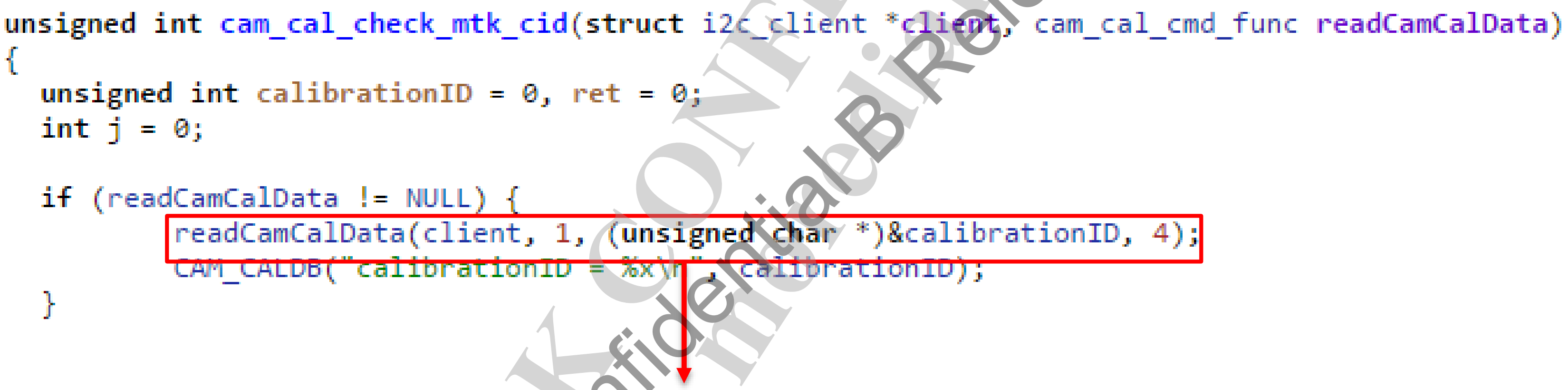
(2)需要修改cam_cal_list.cpp中的:

```
#define MTK_MAX_CID_NUM  4
unsigned int mtkCidList[MTK_MAX_CID_NUM] = {
        0x00000001,
        0x010b00ff,
         0x020b00ff,
         0x030b00ff
};
```

Step4
OTP customization

- calibrationID customization

cam_cal_check_mtk_cid需要修改到相应的地址



```
unsigned int cam_cal_check_mtk_cid(struct i2c_client *client, cam_cal_cmd_func readCamCalData)
{
    unsigned int calibrationID = 0, ret = 0;
    int j = 0;

    if (readCamCalData != NULL) {
            readCamCalData(client, 1, (unsigned char *)&calibrationID, 4);
            CAM_CALDB("calibrationID = %x\n", calibrationID);
    }
```

```
readCamCalData(client, 0x3205, (unsigned char *)&calibrationID, 1);
```

Step4
OTP customization

- calibrationID customization

(3)修改camera_calibration_cam_cal.cpp中的

```
        {//CALIBRATION_LAYOUT_SENSOR_OTP
                0x00000001, 0x010b00ff, CAM_CAL_SINGLE_OTP_DATA,
                {
{0x00000000, 0x00000000, 0x00000000, DoCamCalModuleVersion}, //CAMERA_CAM_CAL_DATA_MODULE_VERSION
{0x00000000, 0x00000005, 0x00000002, DoCamCalPartNumber}, //CAMERA_CAM_CAL_DATA_PART_NUMBER
{0x00000001, 0x00000017, 0x0000074C, DoCamCalSingleLsc}, //CAMERA_CAM_CAL_DATA_SHADING_TABLE
{0x00000001, 0x00000007, 0x0000000E, DoCamCal2AGain}, //CAMERA_CAM_CAL_DATA_3A_GAIN
                ......
                }
```

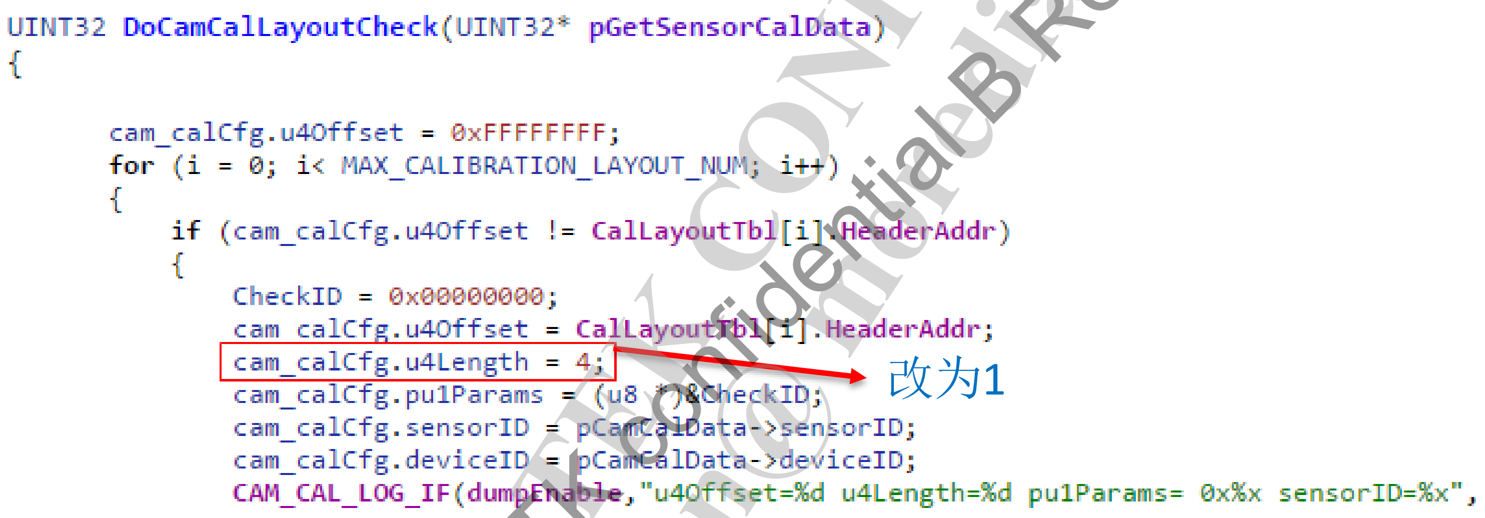
0x00000001, 0x010b00ff修改为: 0x00003205, 0x000000001

确保check layout时候read(0x3205)=0x0000 0001

Step4
OTP customization

- calibrationID customization

因为只需要读0x3205一个寄存器,所以需要修改length为1:



```
UINT32 DoCamCalLayoutCheck(UINT32* pGetSensorCalData)
{

        cam_calCfg.u4Offset = 0xFFFFFFFF;
        for (i = 0; i< MAX_CALIBRATION_LAYOUT_NUM; i++)
        {
            if (cam_calCfg.u4Offset != CalLayoutTbl[i].HeaderAddr)
            {
                CheckID = 0x00000000;
                cam_calCfg.u4Offset = CalLayoutTbl[i].HeaderAddr;
                cam_calCfg.u4Length = 4;
                cam_calCfg.pu1Params = (u8 *)&CheckID;
                cam_calCfg.sensorID = pCamCalData->sensorID;
                cam_calCfg.deviceID = pCamCalData->deviceID;
                CAM_CAL_LOG_IF(dumpEnable,"u4Offset=%d u4Length=%d pu1Params= 0x%x sensorID=%x",
```

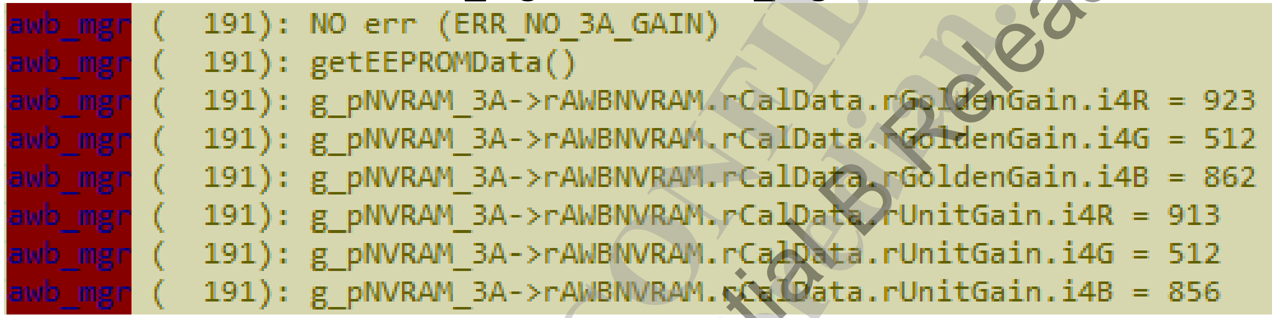
改为1

Outline

- Terminologies
- Sensor OTP porting
- Platform OTP porting
  - OTP old/new architecture
  - Old architecture OTP porting
  - New architecture OTP porting
- Case study

AWB验证

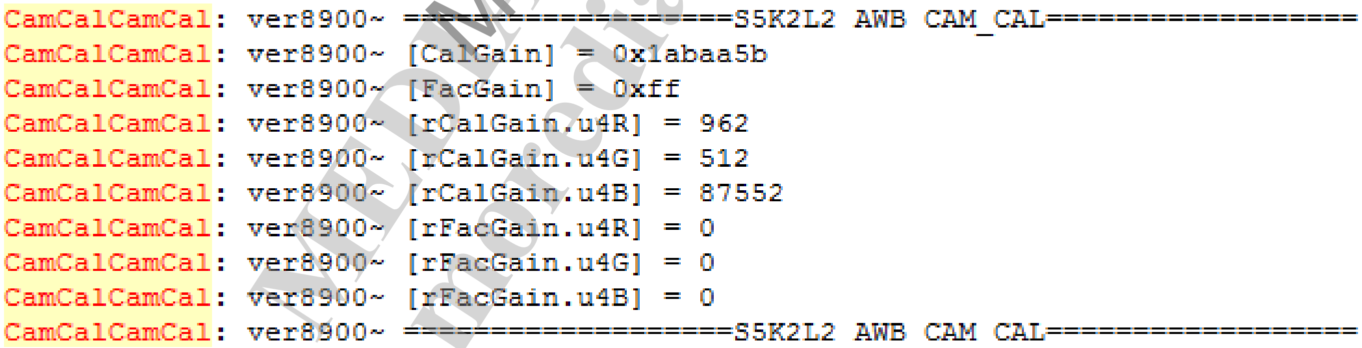
- 用MTKlogger 抓取开机和open camera log
  - Awb_mgr.cpp中log默认是关闭的.需要修改为 int bAwbVerboseEn=1;
  - Awb 的验证需要在main_log中搜索”awb_mgr”:



```
awb_mgr (  191): NO err (ERR_NO_3A_GAIN)
awb_mgr (  191): getEEPROMData()
awb_mgr (  191): g_pNVRAM_3A->rAWBNVRAM.rCalData.rGoldenGain.i4R = 923
awb_mgr (  191): g_pNVRAM_3A->rAWBNVRAM.rCalData.rGoldenGain.i4G = 512
awb_mgr (  191): g_pNVRAM_3A->rAWBNVRAM.rCalData.rGoldenGain.i4B = 862
awb_mgr (  191): g_pNVRAM_3A->rAWBNVRAM.rCalData.rUnitGain.i4R = 913
awb_mgr (  191): g_pNVRAM_3A->rAWBNVRAM.rCalData.rUnitGain.i4G = 512
awb_mgr (  191): g_pNVRAM_3A->rAWBNVRAM.rCalData.rUnitGain.i4B = 856
```

  - Golden/Unit代表读出的AWB值.
  - **注意: R和B值一定大于G, 否则烧录的AWB data有误,或者 AWB计算公式有误.**

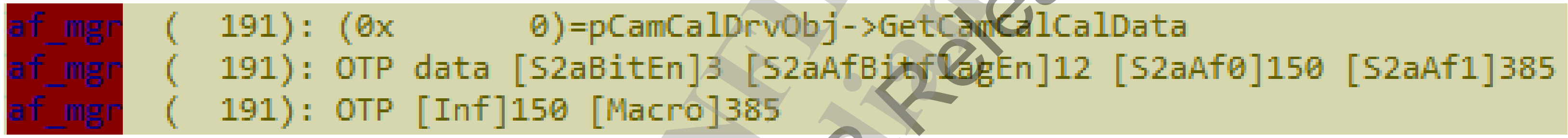
- MT6757 AWB验证方法
  - 打开mtklogger，并打开camera
  - 在main_log中搜”camcalcamcal”



```
CamCalCamCal: ver8900~ ===================S5K2L2 AWB CAM_CAL===================
CamCalCamCal: ver8900~ [CalGain] = 0x1abaa5b
CamCalCamCal: ver8900~ [FacGain] = 0xff
CamCalCamCal: ver8900~ [rCalGain.u4R] = 962
CamCalCamCal: ver8900~ [rCalGain.u4G] = 512
CamCalCamCal: ver8900~ [rCalGain.u4B] = 87552
CamCalCamCal: ver8900~ [rFacGain.u4R] = 0
CamCalCamCal: ver8900~ [rFacGain.u4G] = 0
CamCalCamCal: ver8900~ [rFacGain.u4B] = 0
CamCalCamCal: ver8900~ ===================S5K2L2 AWB CAM CAL===================
```

AF验证

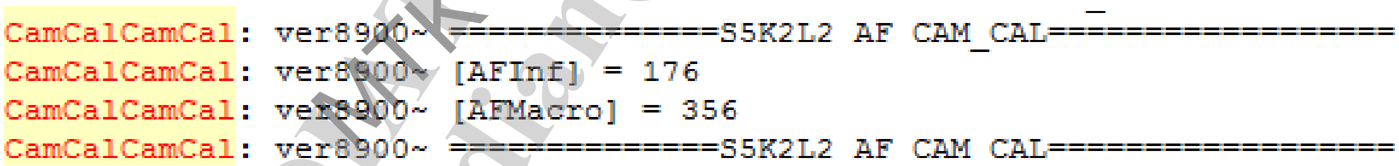
- 用MTKlogger 抓取开机和open camera log
  - AF 的验证需要在main_log中搜索”af_mgr”:



```
af_mgr  (  191): (0x       0)=pCamCalDrvObj->GetCamCalCalData
af_mgr  (  191): OTP data [S2aBitEn]3 [S2aAfBitflagEn]12 [S2aAf0]150 [S2aAf1]385
af_mgr  (  191): OTP [Inf]150 [Macro]385
```

  - [S2aBitEn]=3, 且Macro>Inf. 说明read AF OTP data 成功.
  - Inf/Macro代表读出的AF值.

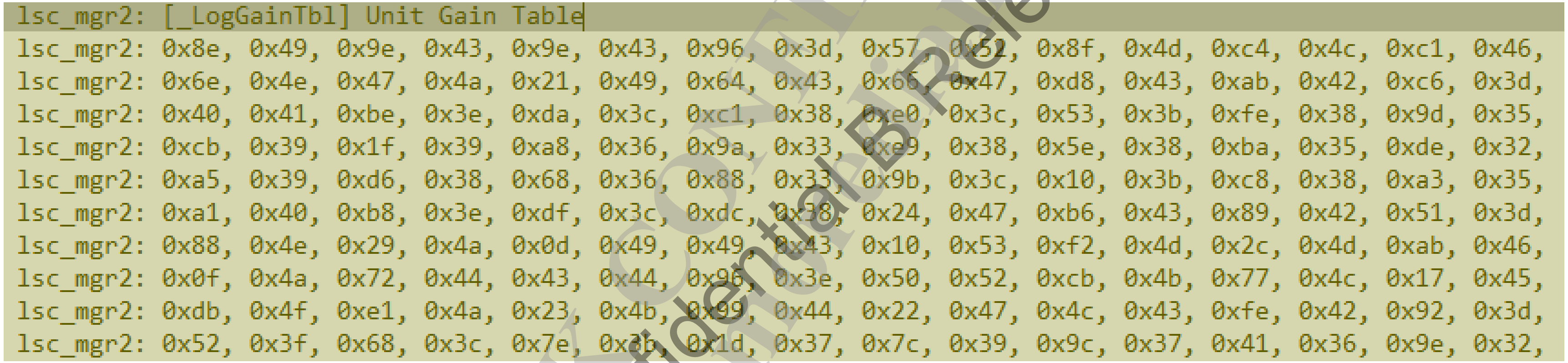
- MT6757 AF验证方法
  - 打开mtklogger，并打开camera
  - 在main_log中搜”camcalcamcal”



```
CamCalCamCal: ver8900~ ===============S5K2L2 AF CAM_CAL==================
CamCalCamCal: ver8900~ [AFInf] = 176
CamCalCamCal: ver8900~ [AFMacro] = 356
CamCalCamCal: ver8900~ ===============S5K2L2 AF CAM_CAL==================
```

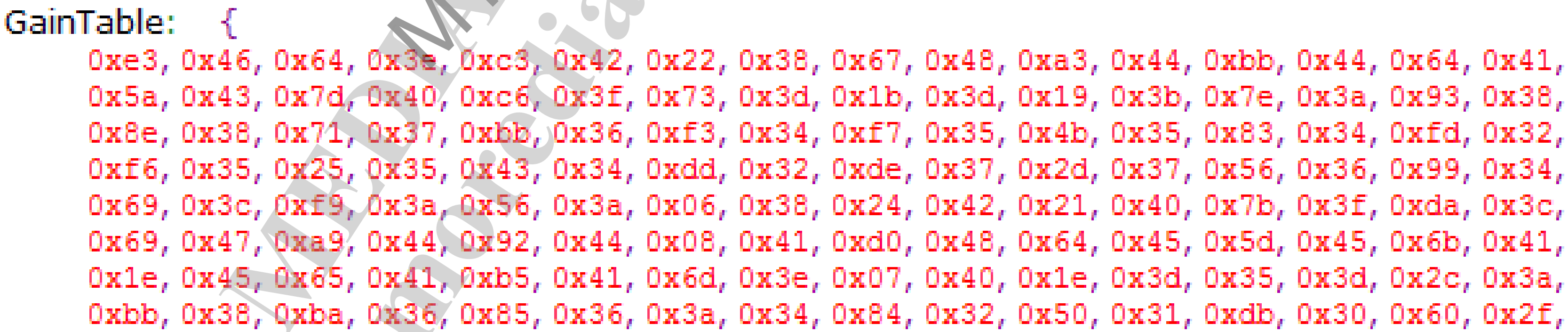
LSC验证(MT6797/6755 以前的版本)

- 用MTKlogger 抓取开机和open camera log
  - Lsc的log要在format+download bin第一次开机log中.
  - 在main_log.boot中搜索”lsc_mgr”:



```
lsc_mgr2: [_LogGainTbl] Unit Gain Table
lsc_mgr2: 0x8e, 0x49, 0x9e, 0x43, 0x9e, 0x43, 0x96, 0x3d, 0x57, 0x52, 0x8f, 0x4d, 0xc4, 0x4c, 0xc1, 0x46,
lsc_mgr2: 0x6e, 0x4e, 0x47, 0x4a, 0x21, 0x49, 0x64, 0x43, 0x66, 0x47, 0xd8, 0x43, 0xab, 0x42, 0xc6, 0x3d,
lsc_mgr2: 0x40, 0x41, 0xbe, 0x3e, 0xda, 0x3c, 0xc1, 0x38, 0xe0, 0x3c, 0x53, 0x3b, 0xfe, 0x38, 0x9d, 0x35,
lsc_mgr2: 0xcb, 0x39, 0x1f, 0x39, 0xa8, 0x36, 0x9a, 0x33, 0xe9, 0x38, 0x5e, 0x38, 0xba, 0x35, 0xde, 0x32,
lsc_mgr2: 0xa5, 0x39, 0xd6, 0x38, 0x68, 0x36, 0x88, 0x33, 0x9b, 0x3c, 0x10, 0x3b, 0xc8, 0x38, 0xa3, 0x35,
lsc_mgr2: 0xa1, 0x40, 0xb8, 0x3e, 0xdf, 0x3c, 0xdc, 0x38, 0x24, 0x47, 0xb6, 0x43, 0x89, 0x42, 0x51, 0x3d,
lsc_mgr2: 0x88, 0x4e, 0x29, 0x4a, 0x0d, 0x49, 0x49, 0x43, 0x10, 0x53, 0xf2, 0x4d, 0x2c, 0x4d, 0xab, 0x46,
lsc_mgr2: 0x0f, 0x4a, 0x72, 0x44, 0x43, 0x44, 0x96, 0x3e, 0x50, 0x52, 0xcb, 0x4b, 0x77, 0x4c, 0x17, 0x45,
lsc_mgr2: 0xdb, 0x4f, 0xe1, 0x4a, 0x23, 0x4b, 0x99, 0x44, 0x22, 0x47, 0x4c, 0x43, 0xfe, 0x42, 0x92, 0x3d,
lsc_mgr2: 0x52, 0x3f, 0x68, 0x3c, 0x7e, 0x3b, 0x1d, 0x37, 0x7c, 0x39, 0x9c, 0x37, 0x41, 0x36, 0x9e, 0x32,
```

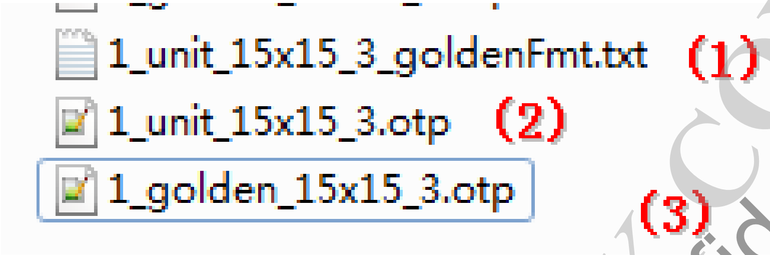
  - Unit Gain Table代表从当前模组中读出来lsc data.
  - **Gold Gain Table代表从Golden模组中读取出来的. 更换golden模组,那log打出的Uint table就是Golden table. 需要把这个table粘贴到camera_isp_lsc_xxxmipiraw.h:**



```
GainTable:    {
        0xe3, 0x46, 0x64, 0x3e, 0xc3, 0x42, 0x22, 0x38, 0x67, 0x48, 0xa3, 0x44, 0xbb, 0x44, 0x64, 0x41,
        0x5a, 0x43, 0x7d, 0x40, 0xc6, 0x3f, 0x73, 0x3d, 0x1b, 0x3d, 0x19, 0x3b, 0x7e, 0x3a, 0x93, 0x38,
        0x8e, 0x38, 0x71, 0x37, 0xbb, 0x36, 0xf3, 0x34, 0xf7, 0x35, 0x4b, 0x35, 0x83, 0x34, 0xfd, 0x32,
        0xf6, 0x35, 0x25, 0x35, 0x43, 0x34, 0xdd, 0x32, 0xde, 0x37, 0x2d, 0x37, 0x56, 0x36, 0x99, 0x34,
        0x69, 0x3c, 0xf9, 0x3a, 0x56, 0x3a, 0x06, 0x38, 0x24, 0x42, 0x21, 0x40, 0x7b, 0x3f, 0xda, 0x3c,
        0x69, 0x47, 0xa9, 0x44, 0x92, 0x44, 0x08, 0x41, 0xd0, 0x48, 0x64, 0x45, 0x5d, 0x45, 0x6b, 0x41,
        0x1e, 0x45, 0x65, 0x41, 0xb5, 0x41, 0x6d, 0x3e, 0x07, 0x40, 0x1e, 0x3d, 0x35, 0x3d, 0x2c, 0x3a,
        0xbb, 0x38, 0xba, 0x36, 0x85, 0x36, 0x3a, 0x34, 0x84, 0x32, 0x50, 0x31, 0xdb, 0x30, 0x60, 0x2f,
```

LSC验证（MT6797/MT6755版本)

- MT6797不会在log中直接将LSC data打印出来，会在根目录下生成shading_otp这个文件夹，下面一共会生成三只与LSC data相关的文件,MT6755 log中有data，根目录也会生成相应文件。



1_unit_15x15_3_goldenFmt.txt (1)

1_unit_15x15_3.otp (2)

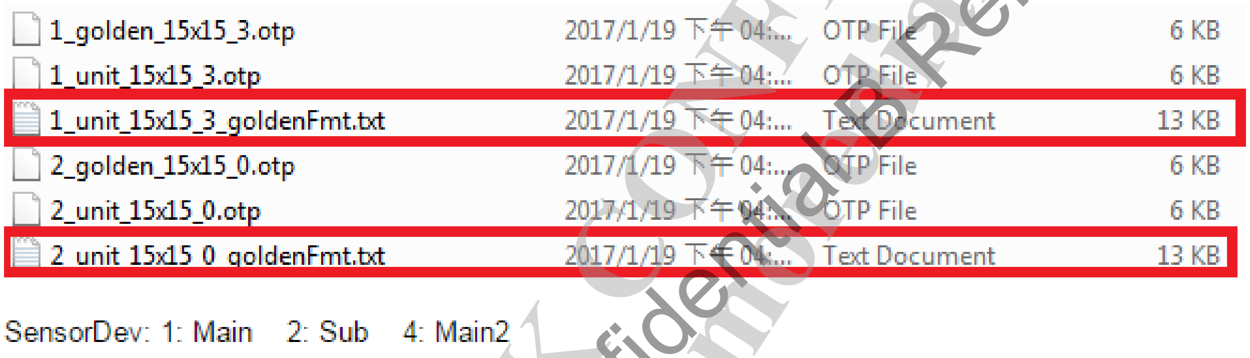
1_golden_15x15_3.otp (3)

shading_otp.rar

- (1)中读出来的是当前模组烧录的LSC data,若使用的是golden模组，那么读取出来的值请直接填入camera_isp_lsc_xxxmipiraw.h中
- (2)与(1)的值一样，只是按照4byte一个保存起来
- (3)是当前camera_isp_lsc_xxx.h 中的SensorGoldenCalTable的Gaintable值

LSC验证（MT6757版本)

- LSC data不会直接打印在log中， 需要从/sdcard/shading_otp目录中提取
- LSC data的使用方法与MT6755/MT6797相同



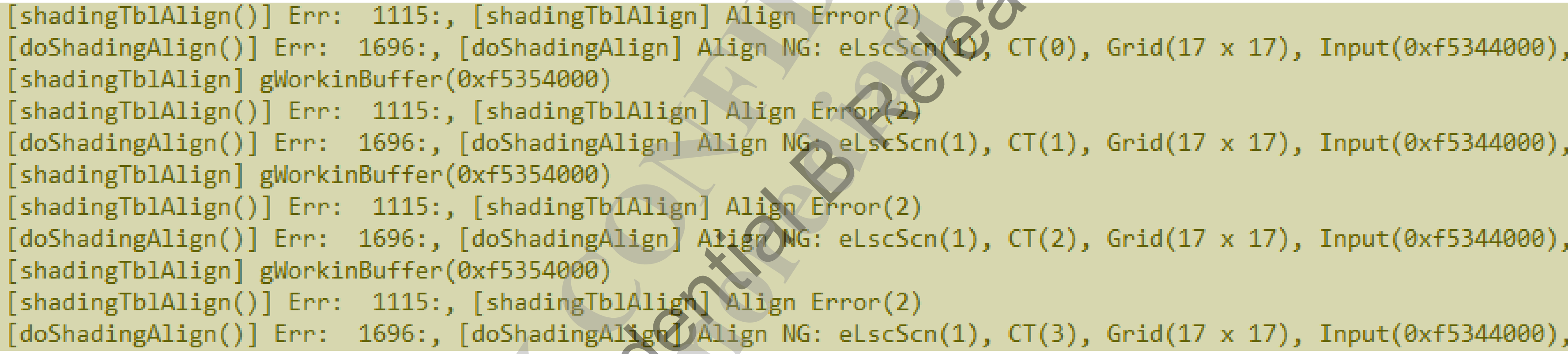
1_golden_15x15_3.otp	2017/1/19 下午 04:...	OTP File	6 KB
1_unit_15x15_3.otp	2017/1/19 下午 04:...	OTP File	6 KB
1_unit_15x15_3_goldenFmt.txt	2017/1/19 下午 04:...	Text Document	13 KB
2_golden_15x15_0.otp	2017/1/19 下午 04:...	OTP File	6 KB
2_unit_15x15_0.otp	2017/1/19 下午 04:...	OTP File	6 KB
2_unit_15x15_0_goldenFmt.txt	2017/1/19 下午 04:...	Text Document	13 KB

SensorDev: 1: Main 2: Sub 4: Main2

- 如果/sdcard/shading_otp中没有dump相关数据
- 请使用以下命令：
  - adb shell setprop debug.lsc_mgr.log 1
  - kill 2396(相应cameraserver的进程id， 可用ps|grep cameraserver查询)
  - sync && sync
  - 打开MTKlogger， 并开启camera
  - adb pull /sdcard/shading_otp就可以得到相应数据

LSC验证

- Error分析
  - 如果出现以下error:

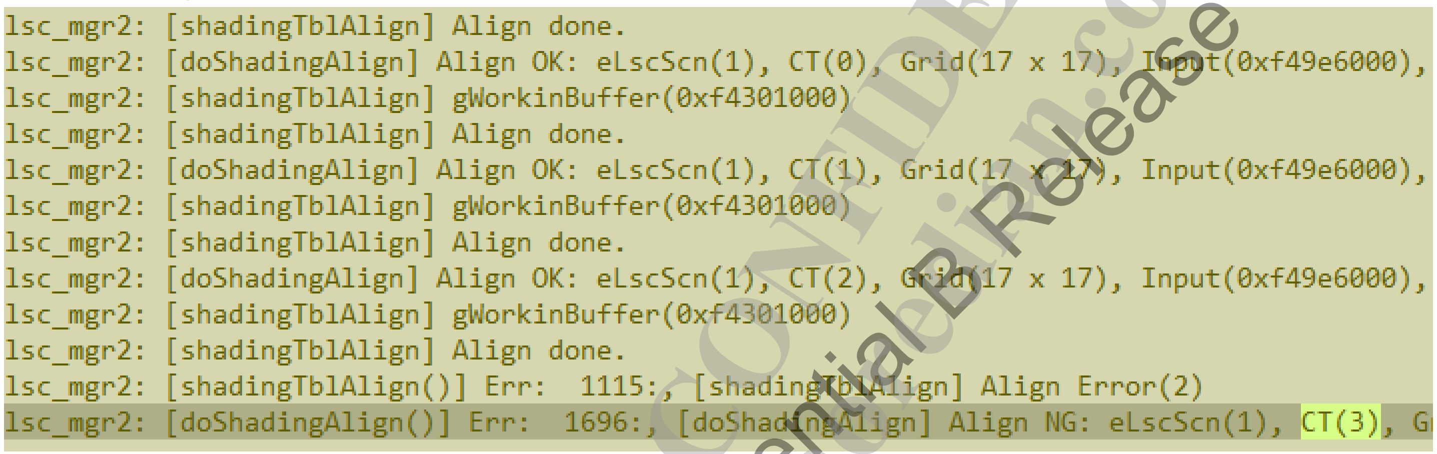


```
[shadingTblAlign()] Err:  1115:, [shadingTblAlign] Align Error(2)
[doShadingAlign()] Err:  1696:, [doShadingAlign] Align NG: eLscScn(1), CT(0), Grid(17 x 17), Input(0xf5344000),
[shadingTblAlign] gWorkinBuffer(0xf5354000)
[shadingTblAlign()] Err:  1115:, [shadingTblAlign] Align Error(2)
[doShadingAlign()] Err:  1696:, [doShadingAlign] Align NG: eLscScn(1), CT(1), Grid(17 x 17), Input(0xf5344000),
[shadingTblAlign] gWorkinBuffer(0xf5354000)
[shadingTblAlign()] Err:  1115:, [shadingTblAlign] Align Error(2)
[doShadingAlign()] Err:  1696:, [doShadingAlign] Align NG: eLscScn(1), CT(2), Grid(17 x 17), Input(0xf5344000),
[shadingTblAlign] gWorkinBuffer(0xf5354000)
[shadingTblAlign()] Err:  1115:, [shadingTblAlign] Align Error(2)
[doShadingAlign()] Err:  1696:, [doShadingAlign] Align NG: eLscScn(1), CT(3), Grid(17 x 17), Input(0xf5344000),
```

说明golden table和uint table 个数 有差异,导致无法calibration. 请确保camera_isp_lsc_xxxmipiraw.h中的GainTable是从Golden 模组读出的,并且保证个数与Uint table相同.

LSC验证

- 正常log如下:

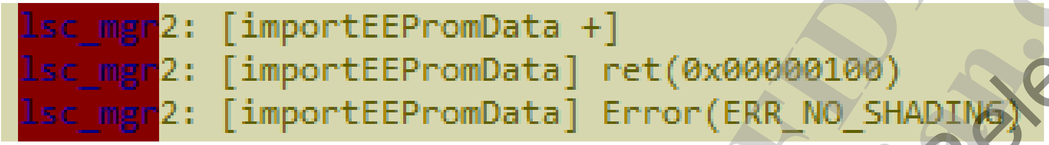


```
lsc_mgr2: [shadingTblAlign] Align done.
lsc_mgr2: [doShadingAlign] Align OK: eLscScn(1), CT(0), Grid(17 x 17), Input(0xf49e6000),
lsc_mgr2: [shadingTblAlign] gWorkinBuffer(0xf4301000)
lsc_mgr2: [shadingTblAlign] Align done.
lsc_mgr2: [doShadingAlign] Align OK: eLscScn(1), CT(1), Grid(17 x 17), Input(0xf49e6000),
lsc_mgr2: [shadingTblAlign] gWorkinBuffer(0xf4301000)
lsc_mgr2: [shadingTblAlign] Align done.
lsc_mgr2: [doShadingAlign] Align OK: eLscScn(1), CT(2), Grid(17 x 17), Input(0xf49e6000),
lsc_mgr2: [shadingTblAlign] gWorkinBuffer(0xf4301000)
lsc_mgr2: [shadingTblAlign] Align done.
lsc_mgr2: [shadingTblAlign()] Err:  1115:, [shadingTblAlign] Align Error(2)
lsc_mgr2: [doShadingAlign()] Err:  1696:, [doShadingAlign] Align NG: eLscScn(1), CT(3), G
```

注意: 如果CT(3)有Align Error请忽略, CT(3)目前没有用到.（建议直接copy CT2到CT3）

debug

- 如果AWB/AF/LSC OTP有error:



```
lsc_mgr2: [importEEPromData +]
lsc_mgr2: [importEEPromData] ret(0x00000100)
lsc_mgr2: [importEEPromData] Error(ERR_NO_SHADING)
```

在开机过程中下adb命令:

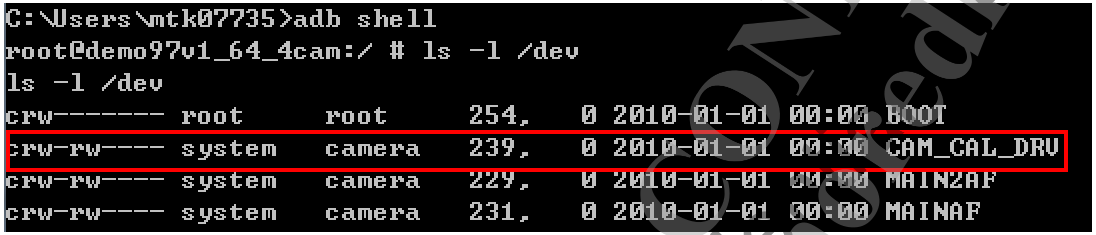
```
adb shell setprop camcalcamcal.log 1
adb shell setprop camcaldrv.log 1
```

并进入camera,抓取的mtklog会打开camera_calibration_cam_cal.cpp中的debug log: **CamCalCamCal**便于分析.

Case Study 1

- I2C bus写错导致Device注册不上

  adb shell连手机，查看设备,若设备没注册上，就无法找到cam_cal_drv这个设备



```
C:\Users\mtk07735>adb shell
root@demo97v1_64_4cam:/ # ls -l /dev
ls -l /dev
crw------- root     root     254,   0 2010-01-01 00:00 BOOT
crw-rw---- system   camera   239,   0 2010-01-01 00:00 CAM_CAL_DRV
crw-rw---- system   camera   229,   0 2010-01-01 00:00 MAIN2AF
crw-rw---- system   camera   231,   0 2010-01-01 00:00 MAINAF
```

Kernellog中搜cam_cal_drv,会有获取I2C client的信息打出来，从这里可以看cam_cal_drv get到的I2C BUS NUM.

```
10211 <7>[  173.249029] .(2)[3205:Binder_2][name:cam_cal_drv&]CAM_CAL_DRV[cam_cal_get_i2c_client]
i2c_info->addr ==50, register i2c g_busNum[0]=0
10212 <7>[  173.249043] .(2)[3205:Binder_2][name:cam_cal_drv&]CAM_CAL_DRV[cam_cal_get_i2c_client]
g_adapt!=NULL, register i2c 0 start !
10213 <3>[  173.249276] .(4)[3205:Binder_2][name:i2c&]ERROR,530: id=0,addr: 50, transfer error
10214 <3>[  173.249282] .(4)[3205:Binder_2][name:i2c&]ERROR,536: I2C_ACKERR
10218 <7>[  173.249293] [I2C]Trans_stop=1,Trans_comp=0,Trans_error=2
10229 <7>[  173.249344] .(4)[3205:Binder_2][name:cam_cal_drv&]CAM_CAL_DRV[cam_cal_get_i2c_client]
failed to get client i2c busID=0
```

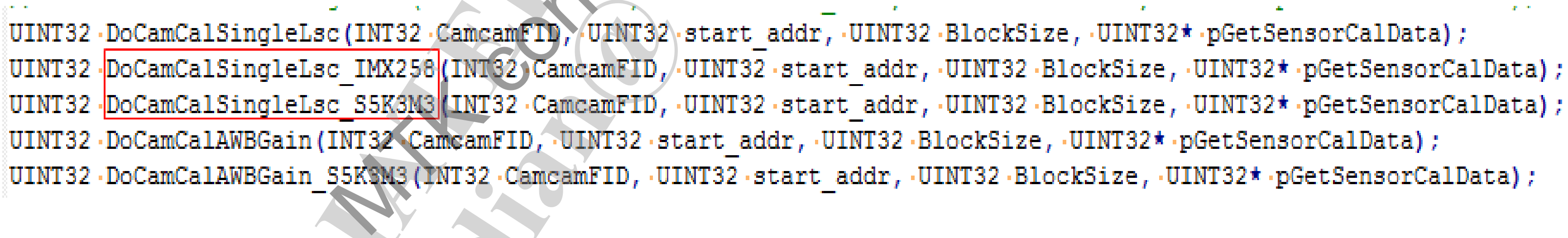
Case Study 2

- 兼容多颗sensor

a) 同一项目需要兼容不同sensor，或有二供模组

b) mian/sub OTP的CheckID的地址和值相同

1. 首先by sensor name定义不同的入口函数

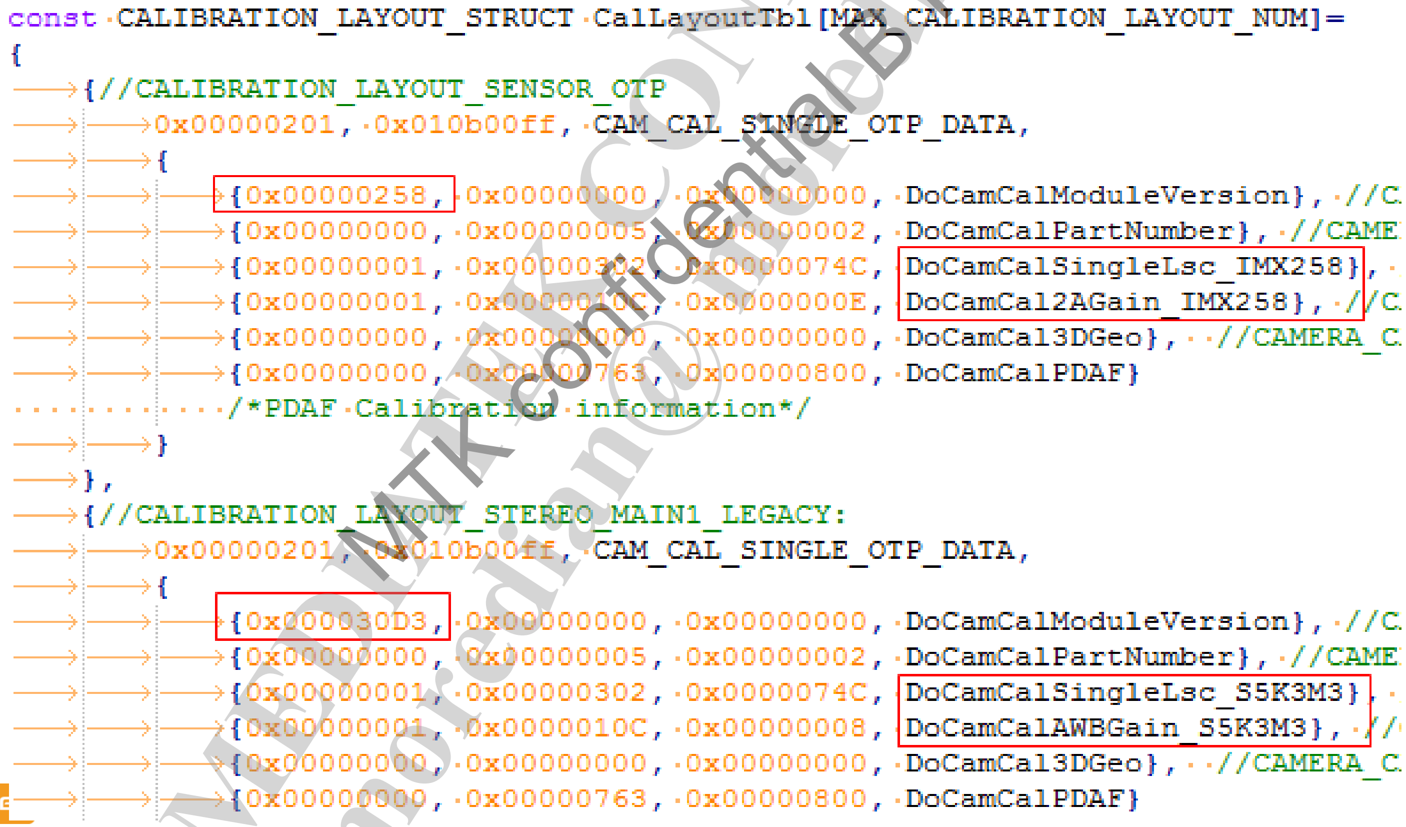


```
UINT32 DoCamCalSingleLsc(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize, UINT32* pGetSensorCalData);
UINT32 DoCamCalSingleLsc_IMX258(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize, UINT32* pGetSensorCalData);
UINT32 DoCamCalSingleLsc_S5K3M3(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize, UINT32* pGetSensorCalData);
UINT32 DoCamCalAWBGain(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize, UINT32* pGetSensorCalData);
UINT32 DoCamCalAWBGain_S5K3M3(INT32 CamcamFID, UINT32 start_addr, UINT32 BlockSize, UINT32* pGetSensorCalData);
```

Case Study 2

- 兼容多颗sensor

2. DoCamCalModuleVersion该行第一个元素设为sensor id
   用来判断应该走哪一个table

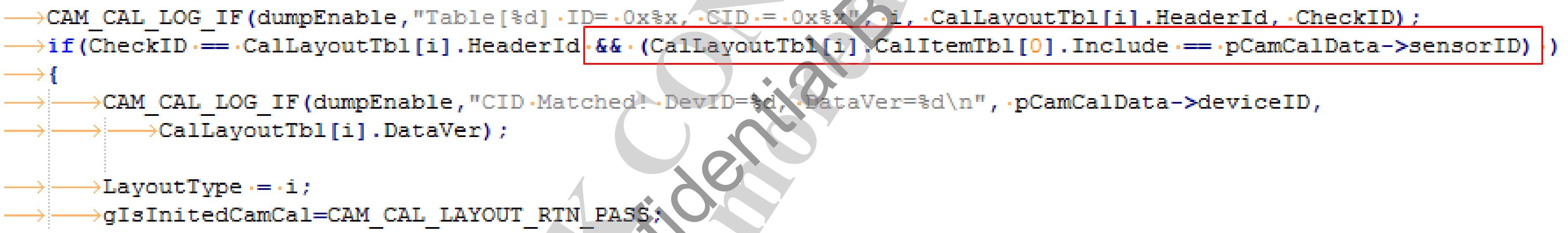


```
const CALIBRATION_LAYOUT_STRUCT CalLayoutTbl[MAX_CALIBRATION_LAYOUT_NUM]=
{
    {//CALIBRATION_LAYOUT_SENSOR_OTP
        0x00000201, 0x010b00ff, CAM_CAL_SINGLE_OTP_DATA,
        {
            {0x00000258, 0x00000000, 0x00000000, DoCamCalModuleVersion}, //C
            {0x00000000, 0x00000005, 0x00000002, DoCamCalPartNumber}, //CAME
            {0x00000001, 0x00000302, 0x0000074C, DoCamCalSingleLsc_IMX258},
            {0x00000001, 0x0000010C, 0x0000000E, DoCamCal2AGain_IMX258}, //C
            {0x00000000, 0x00000000, 0x00000000, DoCamCal3DGeo}, //CAMERA_C
            {0x00000000, 0x00000763, 0x00000800, DoCamCalPDAF}
            /*PDAF Calibration information*/
        }
    },
    {//CALIBRATION_LAYOUT_STEREO_MAIN1_LEGACY:
        0x00000201, 0x010b00ff, CAM_CAL_SINGLE_OTP_DATA,
        {
            {0x000030D3, 0x00000000, 0x00000000, DoCamCalModuleVersion}, //C
            {0x00000000, 0x00000005, 0x00000002, DoCamCalPartNumber}, //CAME
            {0x00000001, 0x00000302, 0x0000074C, DoCamCalSingleLsc_S5K3M3},
            {0x00000001, 0x0000010C, 0x00000008, DoCamCalAWBGain_S5K3M3}, //
            {0x00000000, 0x00000000, 0x00000000, DoCamCal3DGeo}, //CAMERA_C
            {0x00000000, 0x00000763, 0x00000800, DoCamCalPDAF}
```

Case Study 2

- 兼容多颗sensor

  3. DoCamCalLayoutCheck增加sensor id判断来决定走哪一个table



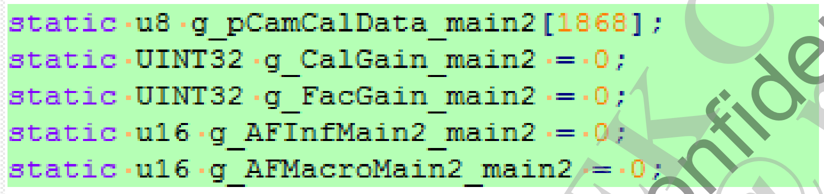
```
	CAM_CAL_LOG_IF(dumpEnable,"Table[%d] ID= 0x%x, CID = 0x%x", i, CalLayoutTbl[i].HeaderId, CheckID);
	if(CheckID == CalLayoutTbl[i].HeaderId && (CalLayoutTbl[i].CalItemTbl[0].Include == pCamCalData->sensorID) )
	{
		CAM_CAL_LOG_IF(dumpEnable,"CID Matched! DevID=%d, DataVer=%d\n", pCamCalData->deviceID,
			CalLayoutTbl[i].DataVer);

		LayoutType = i;
		gIsInitedCamCal=CAM_CAL_LAYOUT_RTN_PASS;
```

Case Study 3

- 预读OTP data

main2 OTP data烧在main eeprom； sub2 OTP data烧在sub eeprom。防止读main2/sub2 data而main/sub未上电不能读到，就需要预读data。

1. 首先定义需要的全局变量来保存预读的data，如LSC/AF/AWB等



```
static u8 g_pCamCalData_main2[1868];
static UINT32 g_CalGain_main2 = 0;
static UINT32 g_FacGain_main2 = 0;
static u16 g_AFInfMain2_main2 = 0;
static u16 g_AFMacroMain2_main2 = 0;
```

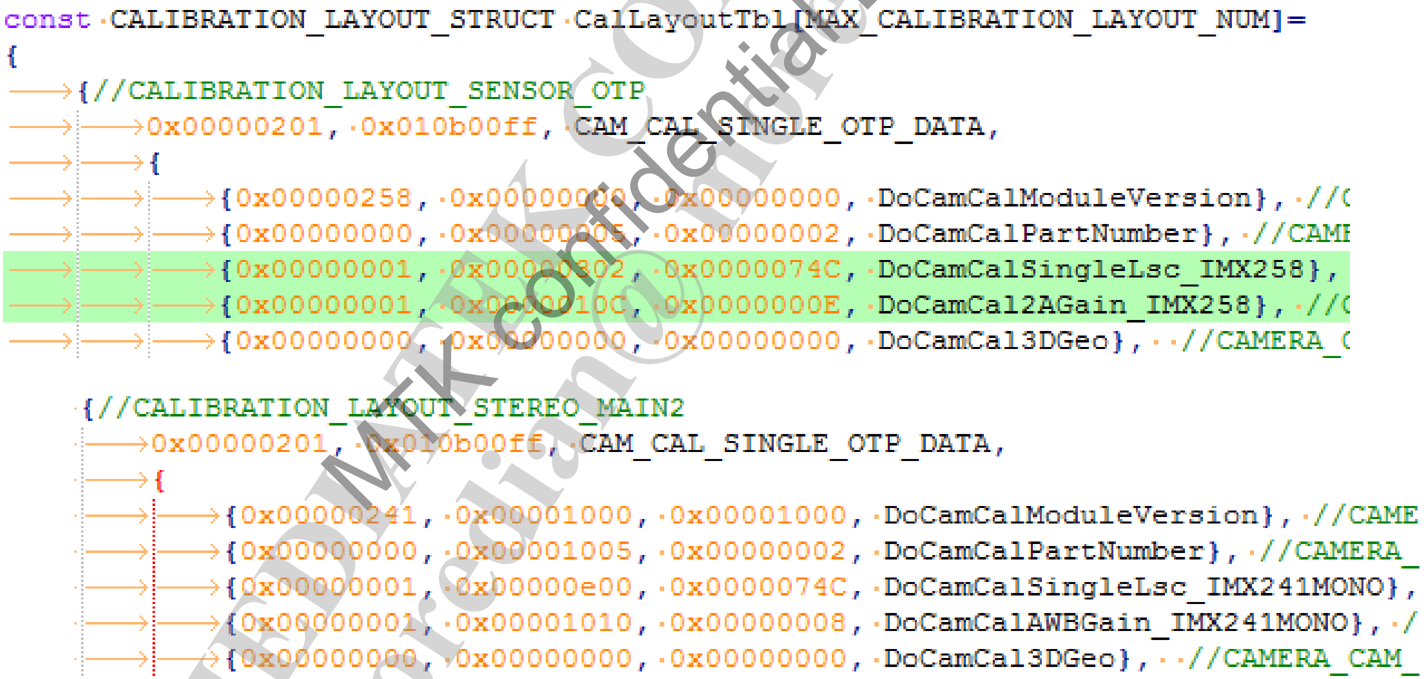
Case Study 3

- 预读OTP data

  2. 为main/main2定义不同的入口函数

     使用sensor id来区分，参考case study 2

     main→imx258; main2→imx241mono



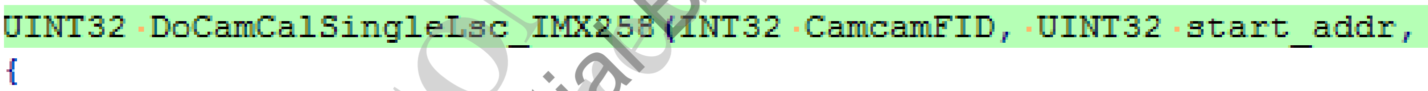
```
const CALIBRATION_LAYOUT_STRUCT CalLayoutTbl[MAX_CALIBRATION_LAYOUT_NUM]=
{
    {//CALIBRATION_LAYOUT_SENSOR_OTP
        0x00000201, 0x010b00ff, CAM_CAL_SINGLE_OTP_DATA,
        {
            {0x00000258, 0x00000000, 0x00000000, DoCamCalModuleVersion}, //C
            {0x00000000, 0x00000005, 0x00000002, DoCamCalPartNumber}, //CAM
            {0x00000001, 0x00000802, 0x0000074C, DoCamCalSingleLsc_IMX258},
            {0x00000001, 0x0000010C, 0x0000000E, DoCamCal2AGain_IMX258}, //C
            {0x00000000, 0x00000000, 0x00000000, DoCamCal3DGeo},  //CAMERA_C

    {//CALIBRATION_LAYOUT_STEREO_MAIN2
        0x00000201, 0x010b00ff, CAM_CAL_SINGLE_OTP_DATA,
        {
            {0x00000241, 0x00001000, 0x00001000, DoCamCalModuleVersion}, //CAME
            {0x00000000, 0x00001005, 0x00000002, DoCamCalPartNumber}, //CAMERA_
            {0x00000001, 0x00000e00, 0x0000074C, DoCamCalSingleLsc_IMX241MONO},
            {0x00000001, 0x00001010, 0x00000008, DoCamCalAWBGain_IMX241MONO}, /
            {0x00000000, 0x00000000, 0x00000000, DoCamCal3DGeo},  //CAMERA_CAM_
```

Case Study 3

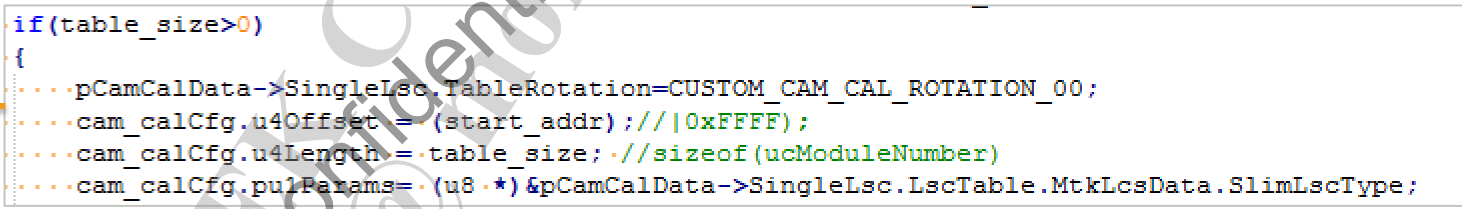
- 预读OTP data

3. 读main OTP data时，将main2的也读出来

在读main LSC data时读出来，因为lsc只在第一次开机去读。

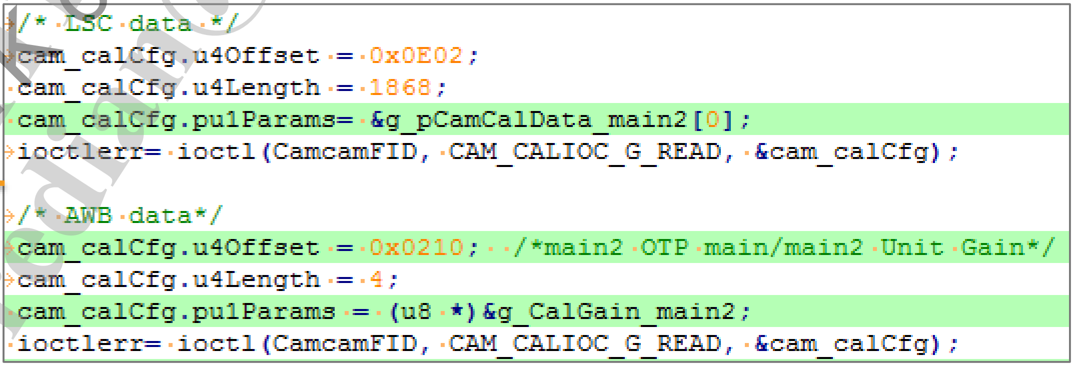


```
UINT32 DoCamCalSingleLsc_IMX258(INT32 CamcamFID, UINT32 start_addr,
{
```



```
if(table_size>0)
{
    pCamCalData->SingleLsc.TableRotation=CUSTOM_CAM_CAL_ROTATION_00;
    cam_calCfg.u4Offset = (start_addr);//|0xFFFF);
    cam_calCfg.u4Length = table_size; //sizeof(ucModuleNumber)
    cam_calCfg.pu1Params= (u8 *)&pCamCalData->SingleLsc.LscTable.MtkLcsData.SlimLscType;
```

读取main的lsc data



```
/* LSC data */
cam_calCfg.u4Offset = 0x0E02;
cam_calCfg.u4Length = 1868;
cam_calCfg.pu1Params= &g_pCamCalData_main2[0];
ioctlerr= ioctl(CamcamFID, CAM_CALIOC_G_READ, &cam_calCfg);

/* AWB data*/
cam_calCfg.u4Offset = 0x0210;  /*main2 OTP main/main2 Unit Gain*/
cam_calCfg.u4Length = 4;
cam_calCfg.pu1Params = (u8 *)&g_CalGain_main2;
ioctlerr= ioctl(CamcamFID, CAM_CALIOC_G_READ, &cam_calCfg);
```

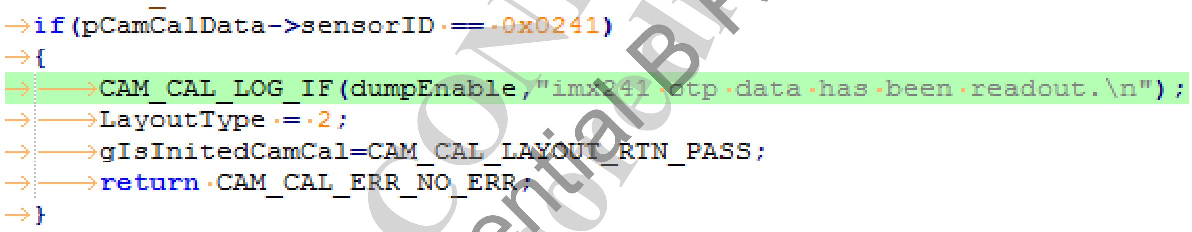
读取main2的lsc/af/awb等

Main2可能只有lsc，根据实际情况定夺

Case Study 3

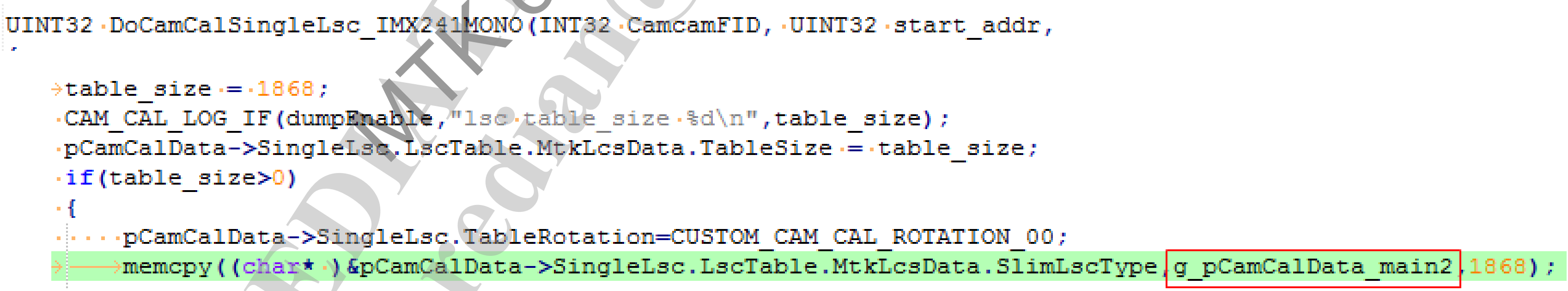
- 预读OTP data

4. 由于main2不会通过i2c去读OTP，需在layoutcheck中直接指定table跳过check ID过程，以获取入口函数



```
	if(pCamCalData->sensorID == 0x0241)
	{
		CAM_CAL_LOG_IF(dumpEnable,"imx241 otp data has been readout.\n");
		LayoutType = 2;
		gIsInitedCamCal=CAM_CAL_LAYOUT_RTN_PASS;
		return CAM_CAL_ERR_NO_ERR;
	}
```

5.在main2的入口函数中直接将独到的数据copy到相应的函数参数中

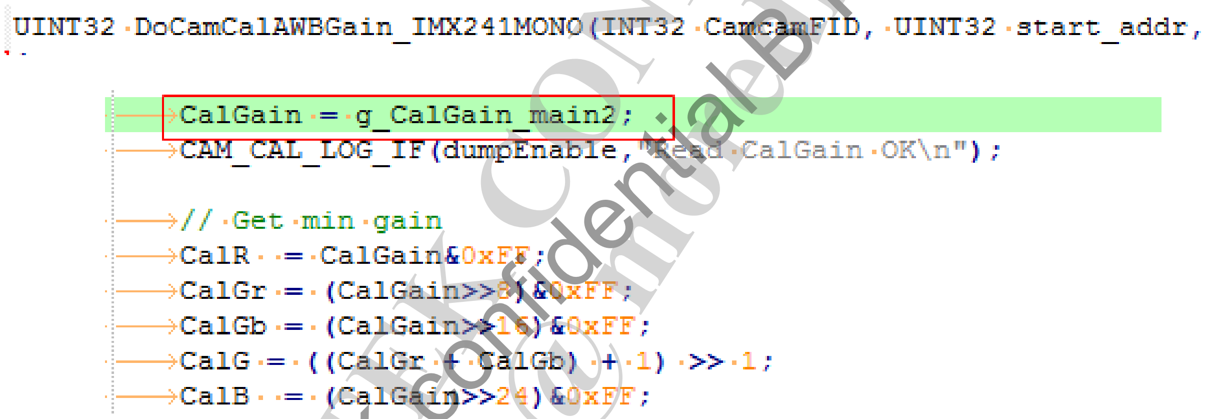


```
UINT32 DoCamCalSingleLsc_IMX241MONO(INT32 CamcamFID, UINT32 start_addr,

	table_size = 1868;
	CAM_CAL_LOG_IF(dumpEnable,"lsc table_size %d\n",table_size);
	pCamCalData->SingleLsc.LscTable.MtkLcsData.TableSize = table_size;
	if(table_size>0)
	{
	    pCamCalData->SingleLsc.TableRotation=CUSTOM_CAM_CAL_ROTATION_00;
		memcpy((char* )&pCamCalData->SingleLsc.LscTable.MtkLcsData.SlimLscType,g_pCamCalData_main2,1868);
```

Case Study 3

- 预读OTP data

6. 如果是AWB data， 则直接赋给CalGain/FacGain做计算即可



```
UINT32 DoCamCalAWBGain_IMX241MONO(INT32 CamcamFID, UINT32 start_addr,

        CalGain = g_CalGain_main2;
        CAM_CAL_LOG_IF(dumpEnable,"Read CalGain OK\n");

        // Get min gain
        CalR  = CalGain&0xFF;
        CalGr = (CalGain>>8)&0xFF;
        CalGb = (CalGain>>16)&0xFF;
        CalG = ((CalGr + CalGb) + 1) >> 1;
        CalB  = (CalGain>>24)&0xFF;
```

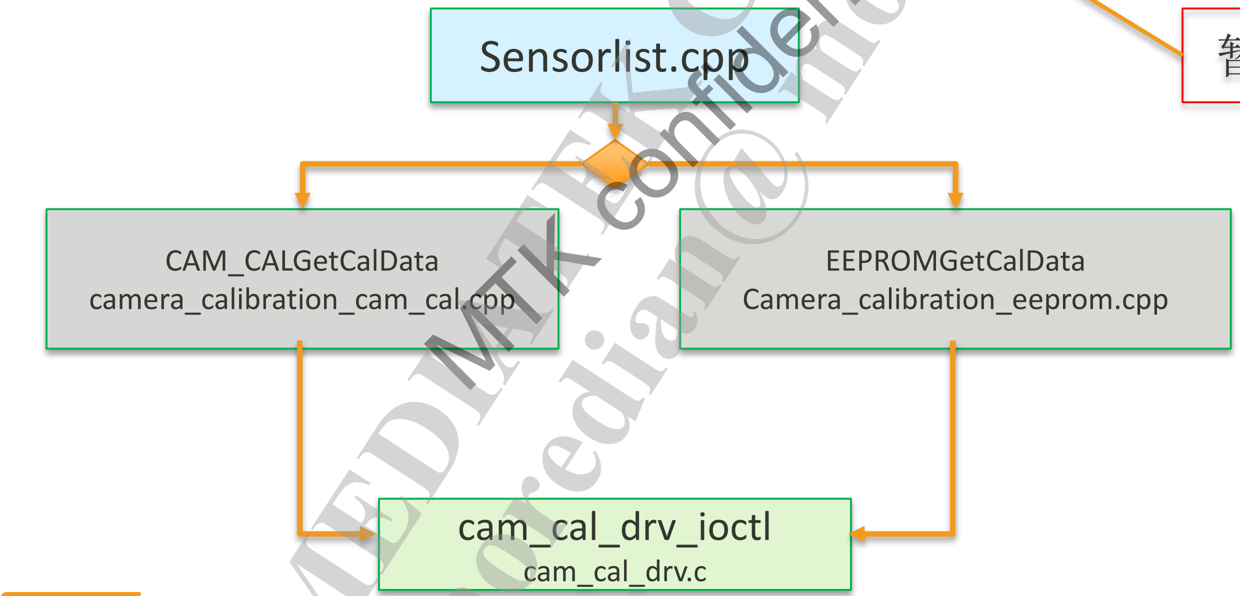
Appendix

Here is another way to read OTP data

/vendor/mediatek/proprietary/custom/mt6757/hal/imgsensor_src/camera_calibration_eeprom.cpp

- Can read OTP all data at once
- Can avoid to read AF/AWB OTP data when opening camera every time
- When main/main2 OTP data are burned in main's eeprom together, main2's otp data cannot be read. Thus, you can use this method.

暂不建议使用该方法



Sensorlist.cpp

CAM_CALGetCalData
camera_calibration_cam_cal.cpp

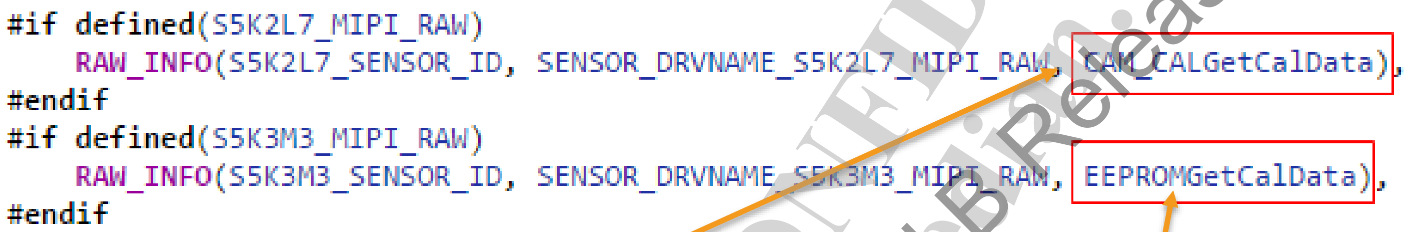
EEPROMGetCalData
Camera_calibration_eeprom.cpp

cam_cal_drv_ioctl
cam_cal_drv.c

Appendix

How to use camera_calibration_eeprom.cpp

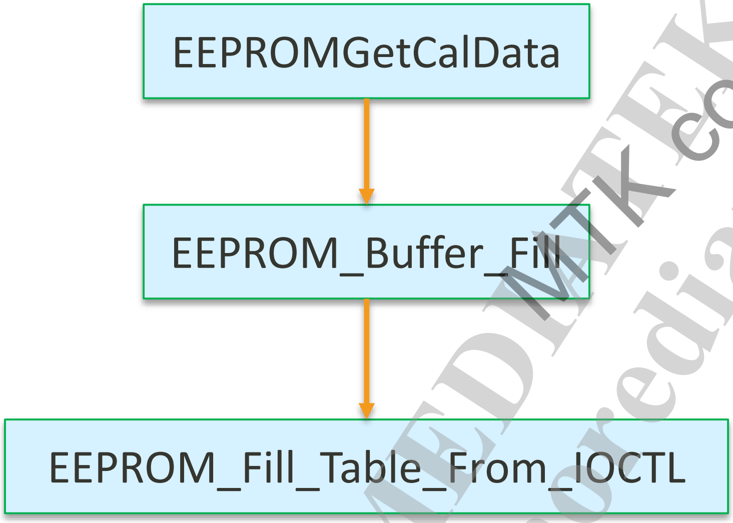
- Modify sensorlist.cpp



```
#if defined(S5K2L7_MIPI_RAW)
    RAW_INFO(S5K2L7_SENSOR_ID, SENSOR_DRVNAME_S5K2L7_MIPI_RAW, CAM_CALGetCalData),
#endif
#if defined(S5K3M3_MIPI_RAW)
    RAW_INFO(S5K3M3_SENSOR_ID, SENSOR_DRVNAME_S5K3M3_MIPI_RAW, EEPROMGetCalData),
#endif
```

Use camea_calibration_cam_cal.cpp

Use camea_calibration_eeprom.cpp



EEPROMGetCalData

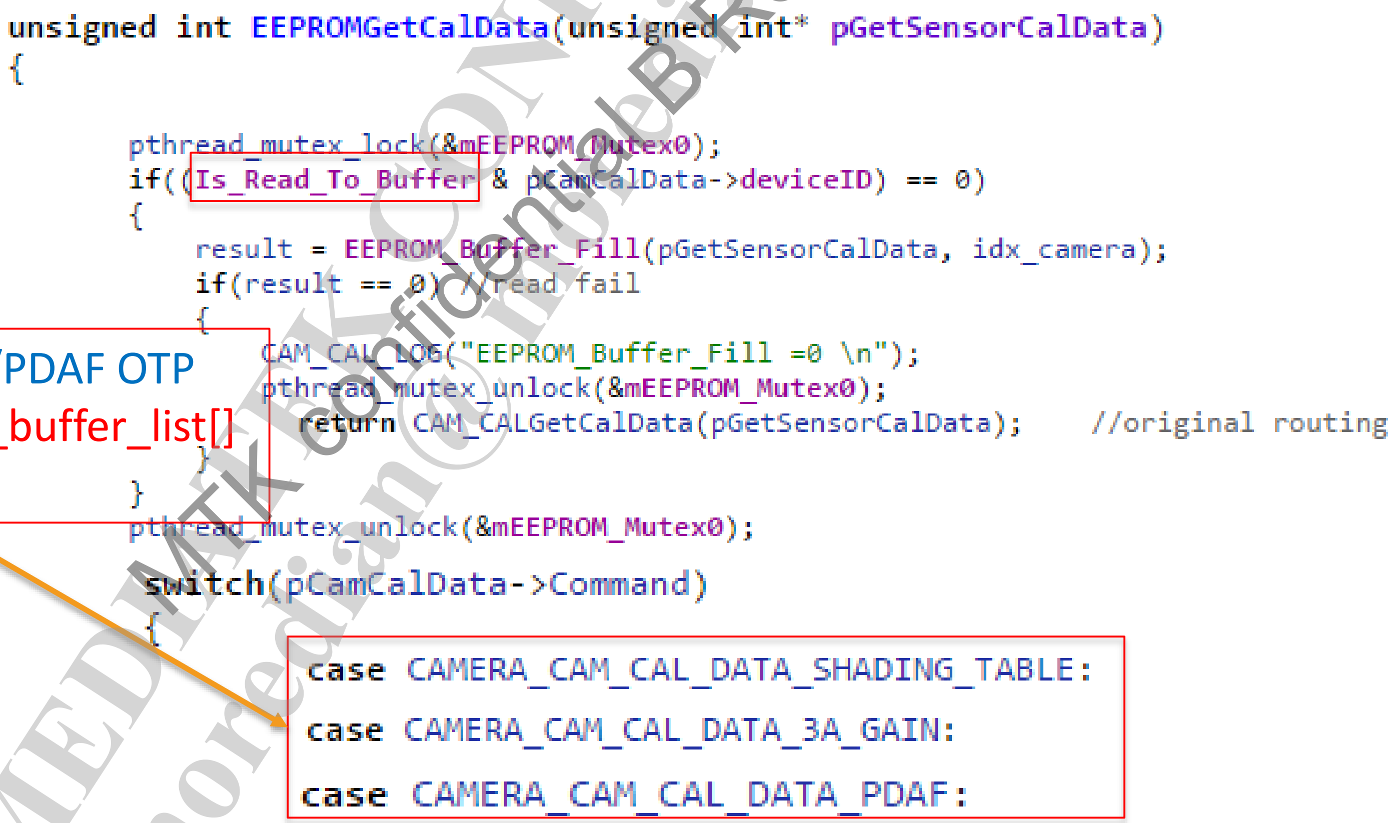
EEPROM_Buffer_Fill

EEPROM_Fill_Table_From_IOCTL

Appendix

How to use camera_calibration_eeprom.cpp

- Hal3A will call EEPROMGetCalData to get the OTP data
  - Is_Read_To_Buffer >0, means the data was read out and will not read again.
  - If EEPROM_Buffer_Fill read failed, will call the old API CAM_CALGetCalData to read the data.



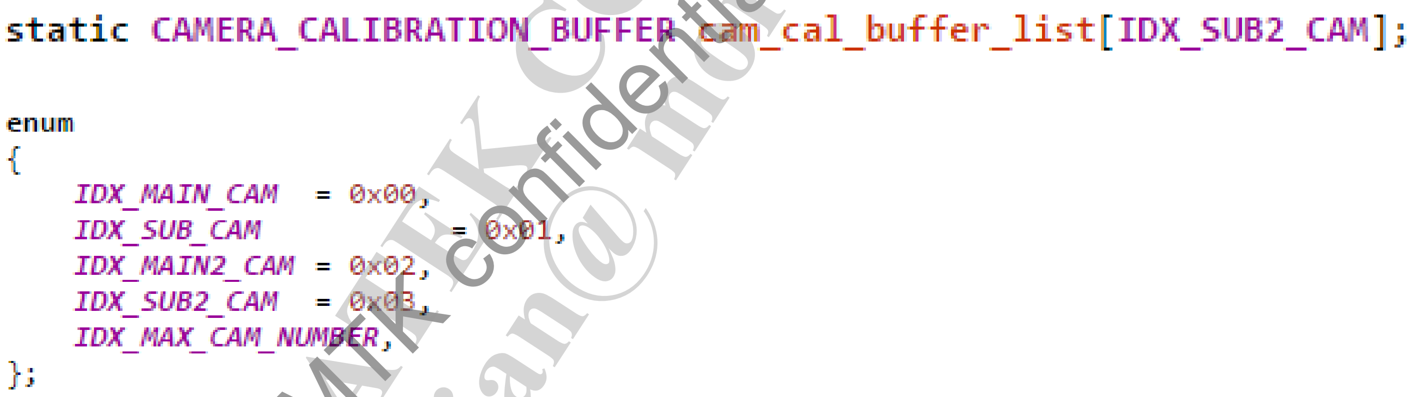
```
unsigned int EEPROMGetCalData(unsigned int* pGetSensorCalData)
{

    pthread_mutex_lock(&mEEPROM_Mutex0);
    if((Is_Read_To_Buffer & pCamCalData->deviceID) == 0)
    {
        result = EEPROM_Buffer_Fill(pGetSensorCalData, idx_camera);
        if(result == 0) //read fail
        {
          CAM_CAL_LOG("EEPROM_Buffer_Fill =0 \n");
          pthread_mutex_unlock(&mEEPROM_Mutex0);
           return CAM_CALGetCalData(pGetSensorCalData);     //original routing
        }
    }
    pthread_mutex_unlock(&mEEPROM_Mutex0);
     switch(pCamCalData->Command)
     {
          case CAMERA_CAM_CAL_DATA_SHADING_TABLE:

          case CAMERA_CAM_CAL_DATA_3A_GAIN:

          case CAMERA_CAM_CAL_DATA_PDAF:
```

Read LSC/AF_AWB/PDAF OTP data from cam_cal_buffer_list[] by Command type

Appendix

How to use camera_calibration_eeprom.cpp

- Cam_cal_buffer_list[]
  - cam_cal_buffer_list[] is a global array for storing the OTP data read by function EEPROM_Fill_Table_From_IOCTL()
  - cam_cal_buffer_list[0] stores main camera’s OTP data, cam_cal_buffer_list[1] stores sub camera’s OTP data, and so on.

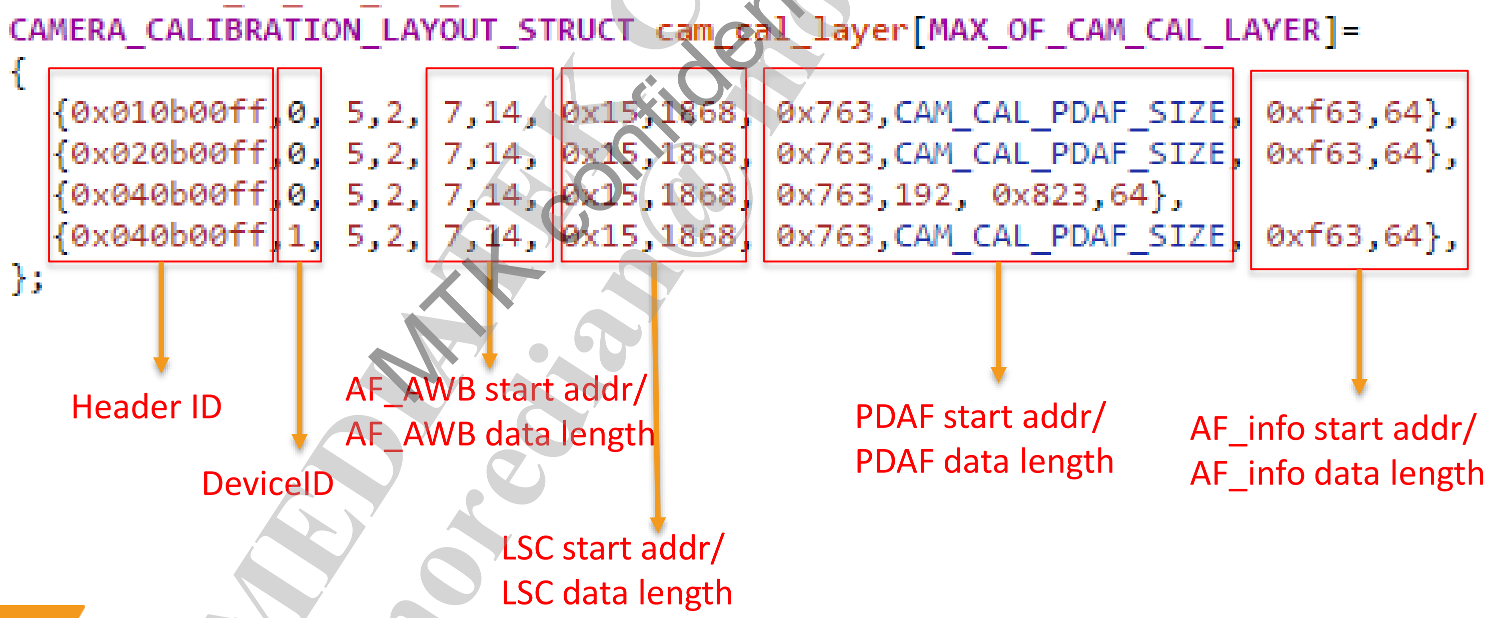


```
static CAMERA_CALIBRATION_BUFFER cam_cal_buffer_list[IDX_SUB2_CAM];

enum
{
    IDX_MAIN_CAM   = 0x00,
    IDX_SUB_CAM            = 0x01,
    IDX_MAIN2_CAM = 0x02,
    IDX_SUB2_CAM   = 0x03,
    IDX_MAX_CAM_NUMBER,
};
```

Appendix

How to use camera_calibration_eeprom.cpp

- EEPROM_Fill_Table_From_IOCTL is the real API to read data via ioctl
  - — You should carefully customize this function to meet your OTP layout document.
  - — Customize the start address and data length in the cam_cal_layer array below.
  - — DeviceID: 0 for main, 1 for sub, 2 for main2, 3 for sub2.
  - — If AF/AWB data was not burned continuously, you need to customize the



```
CAMERA_CALIBRATION_LAYOUT_STRUCT cam_cal_layer[MAX_OF_CAM_CAL_LAYER]=
{
    {0x010b00ff,0, 5,2, 7,14, 0x15,1868, 0x763,CAM_CAL_PDAF_SIZE, 0xf63,64},
    {0x020b00ff,0, 5,2, 7,14, 0x15,1868, 0x763,CAM_CAL_PDAF_SIZE, 0xf63,64},
    {0x040b00ff,0, 5,2, 7,14, 0x15,1868, 0x763,192, 0x823,64},
    {0x040b00ff,1, 5,2, 7,14, 0x15,1868, 0x763,CAM_CAL_PDAF_SIZE, 0xf63,64},
};
```

Header ID

DeviceID

AF_AWB start addr/
AF_AWB data length

LSC start addr/
LSC data length

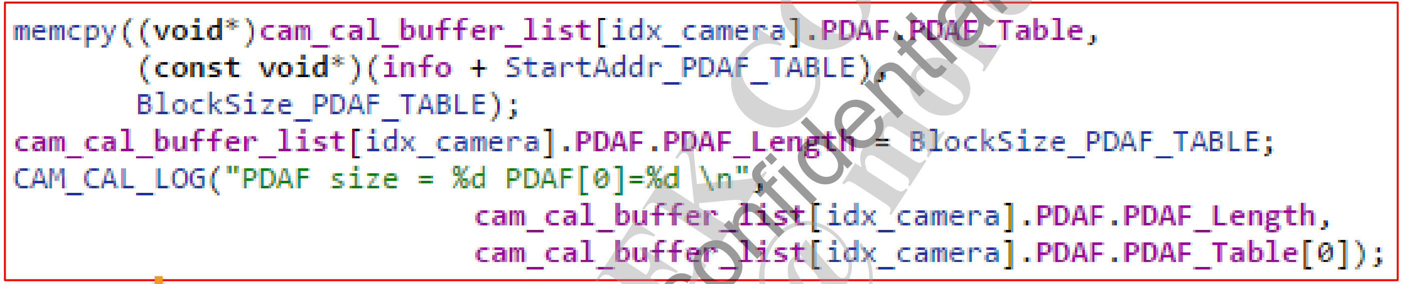
PDAF start addr/
PDAF data length

AF_info start addr/
AF_info data length

Appendix

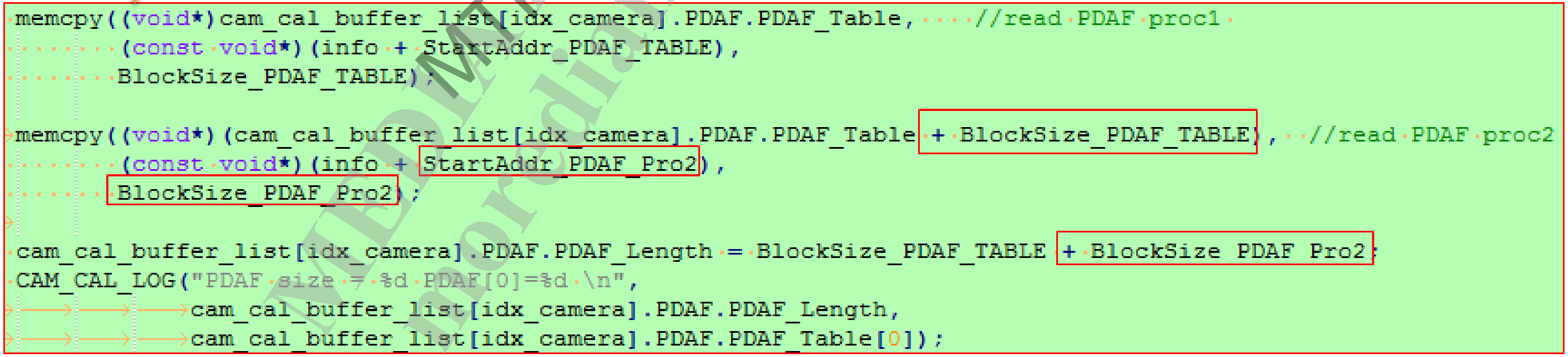
How to use camera_calibration_eeprom.cpp

- Some data may not be burned continuously
  - — AF/AWB data may not be burned continuously
  - — PDAF data has two parts and may not be burned continuously either.
- Thus the EEPROM_Fill_Table_From_IOCTL should be customized
  - — Take PDAF data as an example, its proc1/proc2 data was burned separately.



```
memcpy((void*)cam_cal_buffer_list[idx_camera].PDAF.PDAF_Table,
       (const void*)(info + StartAddr_PDAF_TABLE),
       BlockSize_PDAF_TABLE);
cam_cal_buffer_list[idx_camera].PDAF.PDAF_Length = BlockSize_PDAF_TABLE;
CAM_CAL_LOG("PDAF size = %d PDAF[0]=%d \n",
                          cam_cal_buffer_list[idx_camera].PDAF.PDAF_Length,
                          cam_cal_buffer_list[idx_camera].PDAF.PDAF_Table[0]);
```

Read PDAF proc1 first, then read PDAF proc2



```
memcpy((void*)cam_cal_buffer_list[idx_camera].PDAF.PDAF_Table,    //read PDAF proc1
       (const void*)(info + StartAddr_PDAF_TABLE),
       BlockSize_PDAF_TABLE);

memcpy((void*)(cam_cal_buffer_list[idx_camera].PDAF.PDAF_Table + BlockSize_PDAF_TABLE),   //read PDAF proc2
       (const void*)(info + StartAddr_PDAF_Pro2),
       BlockSize_PDAF_Pro2);

cam_cal_buffer_list[idx_camera].PDAF.PDAF_Length = BlockSize_PDAF_TABLE + BlockSize_PDAF_Pro2;
CAM_CAL_LOG("PDAF size = %d PDAF[0]=%d \n",
            cam_cal_buffer_list[idx_camera].PDAF.PDAF_Length,
            cam_cal_buffer_list[idx_camera].PDAF.PDAF_Table[0]);
```

everyday genius