Confidential B

MEDIATEK

AF Driver porting guide

Agenda

- AF Porting
- AF Debug SOP & Case Study

AF(O) Driver porting guide

生效版本

AF Porting 修改点说明:

- N Before Kibo+ 无效.
- N For Kibo+ 后有效

ANDROID O AF Porting Guide

- File list

device\mediatek\<ProjectName>\ProjectConfig.mk

ProjectConfig

~~kernel-4.4\arch\arm64\configs\<$Project>_debug_config~~
~~kernel-4.4\arch\arm64\configs\<$Project>_config~~
~~kernel-4.4\drivers\misc\mediatek\lens :~~
kernel-4.4\drivers\misc\mediatek\lens\main:
kernel-4.4\drivers\misc\mediatek\lens\main2:
kernel-4.4\drivers\misc\mediatek\lens\sub:

kernel-4.4\drivers\misc\mediatek\lens\main\inc\lens_info.h
kernel-4.4\drivers\misc\mediatek\lens\main\inc\lens_list.h
kernel-4.4\drivers\misc\mediatek\lens\main\main_lens.c
kernel-4.4\drivers\misc\mediatek\lens\main\common\dw9714af\dw9714af.c

kernel

不用再配置了

有微调

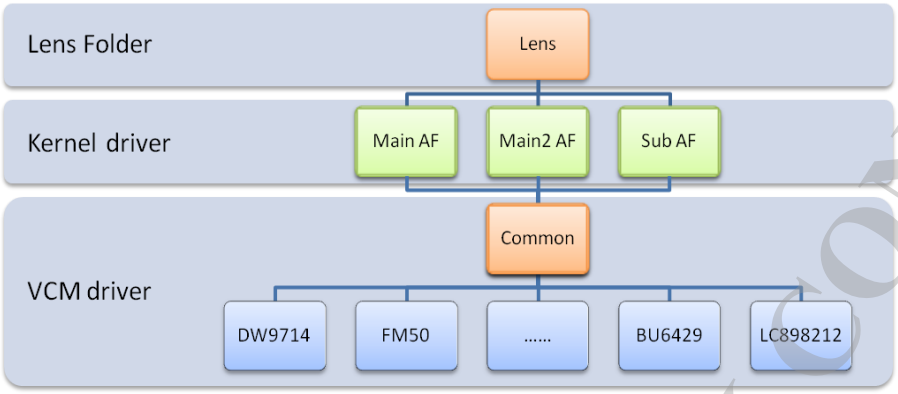
Api有修改,不能直接拿N版本用

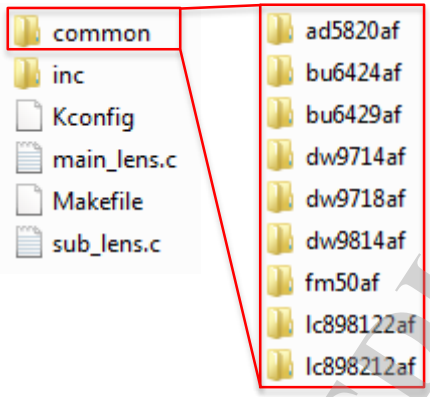
vendor\mediatek\proprietary\custom\mt6797\hal\inc\camera_custom_lens.h
vendor\mediatek\proprietary\custom\mt6797\hal\lens\src\lenslist.cpp
vendor\mediatek\proprietary\custom\[$Platform]\hal\lens\xxxxxxaf\lens_para_XXxxxxAF.cpp

hal

Architecture

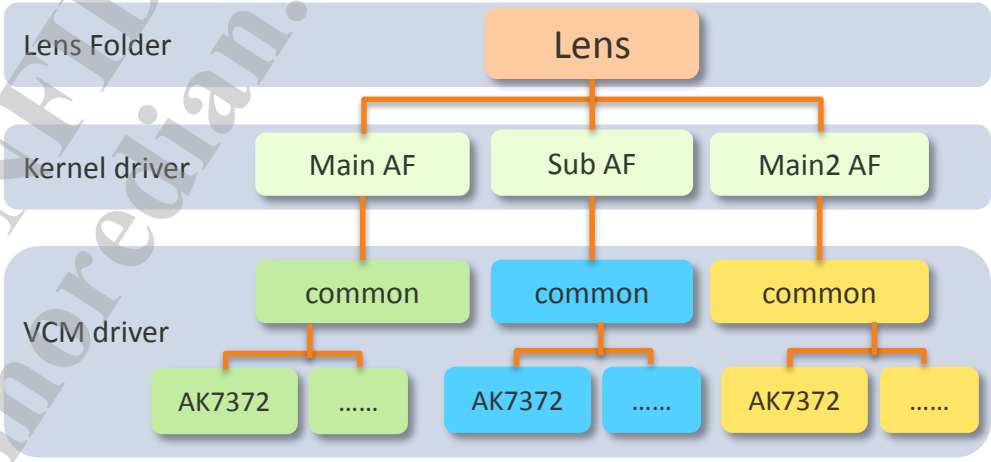
- Kernel 3.18

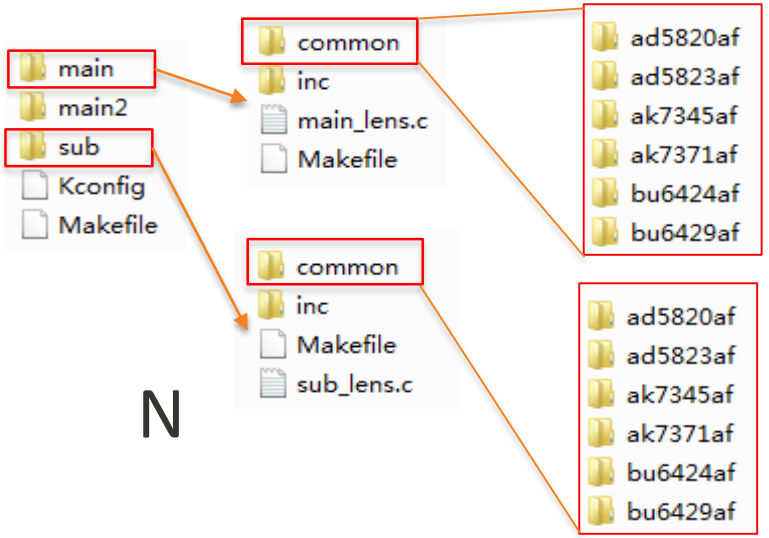




M

- Kernel 4.4





N

Step 1
Lens configuration

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

只需要修改ProjectConfig文件了.

Step2-1
Kernel driver modificaton

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

- 请参考ANDROID M AF Porting Guide修改
  - kernel-4.4\drivers\misc\mediatek\lens\main\inc\lens_list.h
  - kernel-4.4\drivers\misc\mediatek\lens\main2\inc\lens_list.h
  - kernel-4.4\drivers\misc\mediatek\lens\sub\inc\lens_list.h

```
#ifdef CONFIG_MTK_LENS_AK7371AF_SUPPORT
extern void AK7371AF_SetI2Cclient(struct i2c_client *pstAF_I2Cclient, spinlock_t *pAF_SpinLock, int *pAF_Opened);
extern long AK7371AF_Ioctl(struct file *a_pstFile, unsigned int a_u4Command, unsigned long a_u4Param);
extern int AK7371AF_Release(struct inode *a_pstInode, struct file *a_pstFile);
#endif
```

```
#ifdef CONFIG_MTK_LENS_AK7371AF_SUPPORT
#define AK7371AF_SetI2Cclient AK7371AF_SetI2Cclient_Main
#define AK7371AF_Ioctl AK7371AF_Ioctl_Main
#define AK7371AF_Release AK7371AF_Release_Main
extern void AK7371AF_SetI2Cclient(struct i2c_client *pstAF_I2Cclient, spinlock_t *pAF_SpinLock, int *pAF_Opened);
extern long AK7371AF_Ioctl(struct file *a_pstFile, unsigned int a_u4Command, unsigned long a_u4Param);
extern int AK7371AF_Release(struct inode *a_pstInode, struct file *a_pstFile);
extern int AK7371AF_PowerDown(void);
#endif
```

Step2-2

Configure **VCM IC power** in main_lens.c/sub_lens.c/main2_lens.c

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

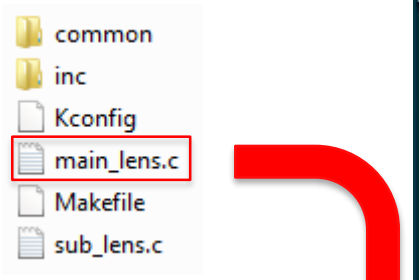
AF power will be controlled by AFRegulatorCtrl()

```
void AFRegulatorCtrl(int Stage)
{

    if (Stage == 0) {
            if (regVCAMAF == NULL) {

            node = of_find_compatible_node(NULL, NULL, "mediatek,CAMERA_MAIN_AF");

            if (node) {
                    kd_node = lens_device->of_node;
                    lens_device->of_node = node;

                    if (strncmp(CONFIG_ARCH_MTK_PROJECT, "k71v1_64_bsp_fhdp", 17) == 0)
                            regVCAMAF = regulator_get(lens_device, "vldo28");
                    else
                            regVCAMAF = regulator_get(lens_device, "vcamaf");

            } else if (Stage == 1) {
                    if (regVCAMAF != NULL && g_regVCAMAFEn == 0) {
                            int Status = regulator_is_enabled(regVCAMAF);

                            LOG_INF("regulator_is_enabled %d\n", Status);

                            if (!Status) {
                                    Status = regulator_set_voltage(regVCAMAF, 2800000, 2800000);
```

Step2-3

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

- Add Driver to DrvList




```
void AF_PowerDown(void)
{
    if (g_pstAF_I2Cclient != NULL) {
        LOG_INF("CONFIG_MTK_PLATFORM : %s\n", CONFIG_MTK_PLATFORM);

        #if defined(CONFIG_MACH_MT6739) || defined(CONFIG_MACH_MT6771) || defined(CONFIG_MACH_MT6775)
        LC898217AF_SetI2Cclient(g_pstAF_I2Cclient, &g_AF_SpinLock, &g_s4AF_Opened);
        LC898217AF_PowerDown();
        #endif

        #ifdef CONFIG_MTK_LENS_AK7371AF_SUPPORT
        AK7371AF_SetI2Cclient(g_pstAF_I2Cclient, &g_AF_SpinLock, &g_s4AF_Opened);
        AK7371AF_PowerDown();
        #endif

        #ifdef CONFIG_MACH_MT6758
        AK7371AF_SetI2Cclient(g_pstAF_I2Cclient, &g_AF_SpinLock, &g_s4AF_Opened);
        AK7371AF_PowerDown();

        BU63169AF_SetI2Cclient(g_pstAF_I2Cclient, &g_AF_SpinLock, &g_s4AF_Opened);
        BU63169AF_PowerDown();
        #endif
    }
```

```
static struct stAF_DrvList g_stAF_DrvList[MAX_NUM_OF_LENS] = {
    {1, AFDRV_AK7371AF, AK7371AF_SetI2Cclient, AK7371AF_Ioctl, AK7371AF_Release, NULL},
    {1, AFDRV_BU6424AF, BU6424AF_SetI2Cclient, BU6424AF_Ioctl, BU6424AF_Release, NULL},
    {1, AFDRV_BU6429AF, BU6429AF_SetI2Cclient, BU6429AF_Ioctl, BU6429AF_Release, NULL},
    {1, AFDRV_BU64748AF, bu64748af_SetI2Cclient_Main, bu64748af_Ioctl_Main, bu64748af_Release_Main, NULL},
    {1,
        #ifdef CONFIG_MTK_LENS_BU63165AF_SUPPORT
```

Step 3
Lens HAL modification

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

- 请参考ANDROID M AF Porting Guide修改

Step 4
Permission modification

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

默认codebase里面已经配置了main sub main2的权限, 无需再care了

AF Driver Porting

Android N branch

ANDROID N AF Porting Guide

- File list

device\mediatek\<ProjectName>\ProjectConfig.mk — ProjectConfig

kernel-4.4\arch\arm64\configs\<$Project>_debug_config
kernel-4.4\arch\arm64\configs\<$Project>_config
kernel-4.4\drivers\misc\mediatek\lens :
kernel-4.4\drivers\misc\mediatek\lens\main:
kernel-4.4\drivers\misc\mediatek\lens\main2:
kernel-4.4\drivers\misc\mediatek\lens\sub:

kernel-4.4\drivers\misc\mediatek\lens\main\inc\lens_info.h
kernel-4.4\drivers\misc\mediatek\lens\main\inc\lens_list.h
kernel-4.4\drivers\misc\mediatek\lens\main\main_lens.c
kernel-4.4\drivers\misc\mediatek\lens\main\common\dw9714af\dw9714af.c

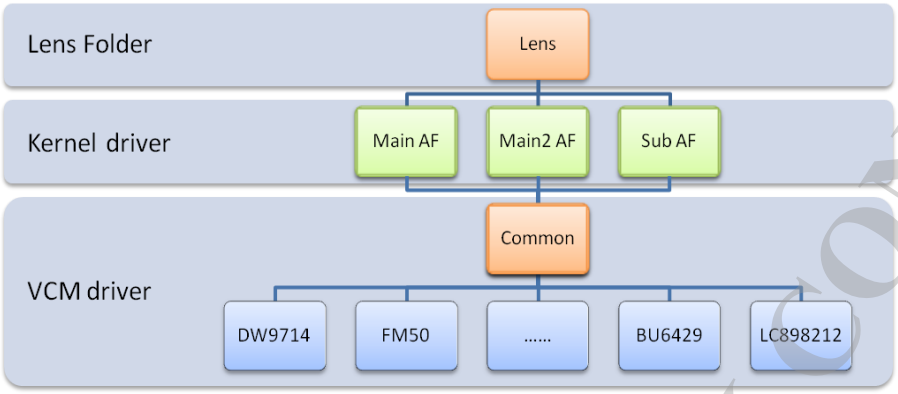
— kernel

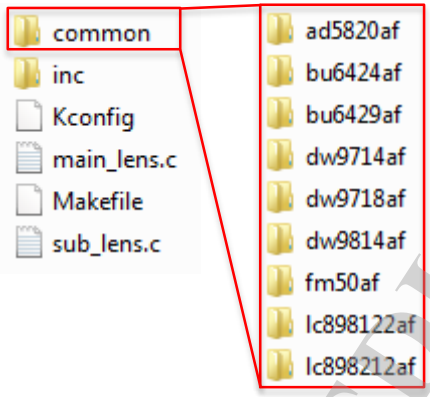
vendor\mediatek\proprietary\custom\mt6797\hal\inc\camera_custom_lens.h
vendor\mediatek\proprietary\custom\mt6797\hal\lens\src\lenslist.cpp
vendor\mediatek\proprietary\custom\[$Platform]\hal\lens\xxxxxxaf\lens_para_XXxxxxAF.cpp

— hal

Architecture

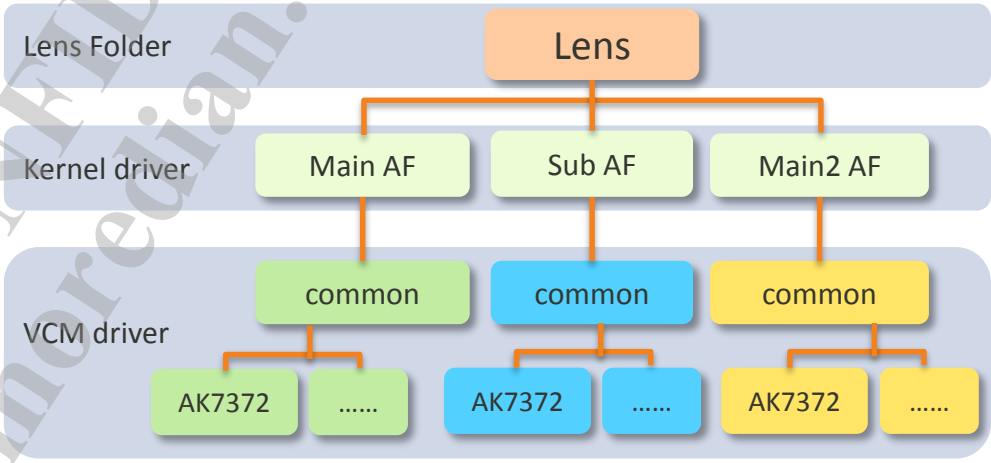
- Kernel 3.18

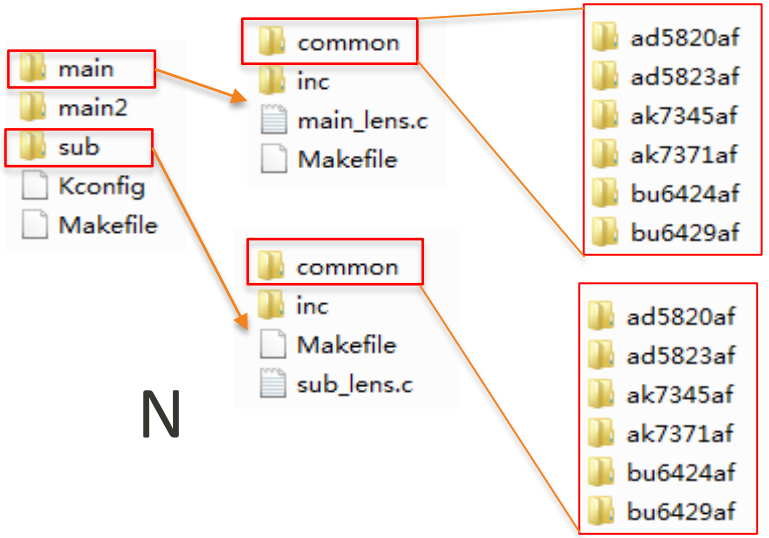




M

- Kernel 4.4





N

Step 1
Lens configuration

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

- 请参考ANDROID M AF Porting Guide修改

Step2
Kernel driver modificaton

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

- 请参考ANDROID M AF Porting Guide修改
  - kernel-4.4\drivers\misc\mediatek\lens\main\inc\lens_list.h
  - kernel-4.4\drivers\misc\mediatek\lens\main2\inc\lens_list.h
  - kernel-4.4\drivers\misc\mediatek\lens\sub\inc\lens_list.h

```
#ifdef CONFIG_MTK_LENS_AK7371AF_SUPPORT
extern void AK7371AF_SetI2Cclient(struct i2c_client *pstAF_I2Cclient, spinlock_t *pAF_SpinLock, int *pAF_Opened);
extern long AK7371AF_Ioctl(struct file *a_pstFile, unsigned int a_u4Command, unsigned long a_u4Param);
extern int AK7371AF_Release(struct inode *a_pstInode, struct file *a_pstFile);
#endif
```

```
#ifdef CONFIG_MTK_LENS_AK7371AF_SUPPORT
#define AK7371AF_SetI2Cclient AK7371AF_SetI2Cclient_Main
#define AK7371AF_Ioctl AK7371AF_Ioctl_Main
#define AK7371AF_Release AK7371AF_Release_Main
extern void AK7371AF_SetI2Cclient(struct i2c_client *pstAF_I2Cclient, spinlock_t *pAF_SpinLock, int *pAF_Opened);
extern long AK7371AF_Ioctl(struct file *a_pstFile, unsigned int a_u4Command, unsigned long a_u4Param);
extern int AK7371AF_Release(struct inode *a_pstInode, struct file *a_pstFile);
extern int AK7371AF_PowerDown(void);
#endif
```

Step 3
Lens HAL modification

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

- 请参考ANDROID M AF Porting Guide修改

Step 4
Permission modification

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

- 请参考ANDROID M AF Porting Guide修改

AF Driver Porting

Android M branch

ANDROID M AF Porting Guide

- File list

Config

1. kernel-3.18\arch\arm64\configs\<ProjectName>_debug_defconfig
2. kernel-3.18\arch\arm64\configs\<ProjectName>_defconfig
3. device\mediatek\<ProjectName>\ProjectConfig.mk

Kernel

1. kernel-3.18\drivers\misc\mediatek\lens\inc\lens_list.h lens_info.h
2. kernel-3.18\drivers\misc\mediatek\lens\main_lens.c sub_lens.c
3. kernel-3.18\drivers\misc\mediatek\lens\common\xxxxxxaf\XXxxxxAF.c
4. kernel-3.18\drivers\misc\mediatek\lens\Kconfig
5. kernel-3.18\drivers\misc\mediatek\lens\Makefile

HAL

1. vendor\mediatek\proprietary\custom\[$Platform]\hal\inc\camera_custom_lens.h
2. vendor\mediatek\proprietary\custom\[$Platform]\hal\lens\src\lenslist.cpp
3. vendor\mediatek\proprietary\custom\[$Platform]\hal\lens\xxxxxxaf\lens_para_XXxxxxAF.cpp

Permision

1. device\mediatek\common\sepolicy\file.contexts mediaserver.te meta_tst.te
2. device\mediatek\$Project\init.project.rc

Step 1-1
Add Lens Driver in Project Configure

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

- Check 『 ***_defconfig*** 』 to enable lens driver.
  - alps\kernel-3.18\arch\arm64\configs

① Add lens driver in supporting list

```
# CONFIG_MTK_LENS_DW9718AF_SUPPORT is not set
CONFIG_MTK_LENS_DW9714AF_SUPPORT=y
CONFIG_MTK_LENS_LC898122AF_SUPPORT=y
CONFIG_MTK_LENS_LC898212AF_SUPPORT=y
# CONFIG_MTK_LENS_FM50AF_SUPPORT is not set
# CONFIG_MTK_LENS_MT9P017AF_SUPPORT is not set
# CONFIG_MTK_LENS_OV8825AF_SUPPORT is not set
# CONFIG_MTK_LENS_SENSORDRIVE_SUPPORT is not set
# CONFIG_MTK_LENS_GAF001AF_SUPPORT is not set
# CONFIG_MTK_LENS_GAF002AF_SUPPORT is not set
# CONFIG_MTK_LENS_GAF008AF_SUPPORT is not set
# CONFIG_MTK_CAM_CAL_GT24C32A_SUPPORT is not set
# CONFIG_MTK_CAM_CAL_BRCC064GWZ_3_SUPPORT is not set
CONFIG_MTK_CPU_STRESS=y
CONFIG_MTK_LASTPC=y
CONFIG_MTK_FMRADIO=y
CONFIG_MTK_HWMON=y
CONFIG_MTK_CMDQ=y
CONFIG_MTK_VIDEOX=y
CONFIG_MTK_MT_LOGGER=y
CONFIG_MTK_CONN_MD=y
CONFIG_MTK_LENS=y
```

② Add lens driver in supporting list

```
# CONFIG_ARCH_MT6582 is not set
# CONFIG_ARCH_MT6592 is not set
CONFIG_ARCH_MT6752=y
# CONFIG_ARCH_MT6795 is not set
# CONFIG_ARCH_MT8127 is not set
```

③Make sure target platform is defined and enabled.

Step 1-2
Enable Lens Driver Define in HAL

- Add lens driver in 『 ***ProjectConfig.mk*** 』 for HAL.
  - alps\device\mediatek\[***Project***]

② Add variable in global define list

```
AUTO_ADD_GLOBAL_DEFINE_BY_NAME_VALUE = MTK_HAC_SUPPORT SIM_ME_LOCK_MODE CUSTOM_CONFIG_MAX_DRAM_SIZE MTK_MAGICONFERENCE_SUPPORT
AUTO_ADD_GLOBAL_DEFINE_BY_VALUE = BOOT_LOGO MTK_AUDIO_BLOUD_CUSTOMPARAMETER_REV MTK_PLATFORM CUSTOM_HAL_LENS CUSTOM_KERNEL_LENS
BOOT_LOGO = fhd
BUILD_GMS = no
BUILD_KERNEL = yes
BUILD_LK = yes
BUILD_MD32 = yes
BUILD_MTK_SDK =
BUILD_PRELOADER = yes
BUILD_UBOOT = no
CUSTOM_BUILD_VERNO =
CUSTOM_CONFIG_MAX_DRAM_SIZE = 0x100000000
CUSTOM_HAL_ANT = mt6752_ant_m1
CUSTOM_HAL_AUDIOFLINGER = audio
CUSTOM_HAL_BLUETOOTH = bluetooth
CUSTOM_HAL_CAMERA = camera
CUSTOM_HAL_CAM_CAL = dummy_eeprom
CUSTOM_HAL_EEPROM = dummy_eeprom
CUSTOM_HAL_FLASHLIGHT = dummy_flashlight
CUSTOM_HAL_IMGSENSOR = imx214_mipi_raw imx135_mipi_raw ov5648_mipi_raw s5k2p8_mipi_raw
CUSTOM_HAL_LENS = lc898122af ad5820af dw9714af dummy_lens
```

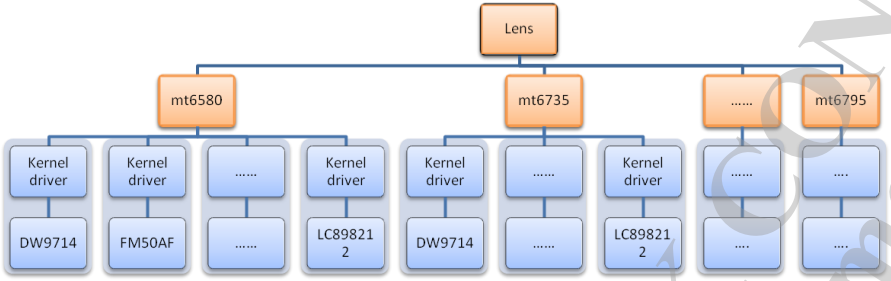
① Add lens driver CUSTOM_KERNEL_LENS
Add lens driver CUSTOM_HAL_LENS

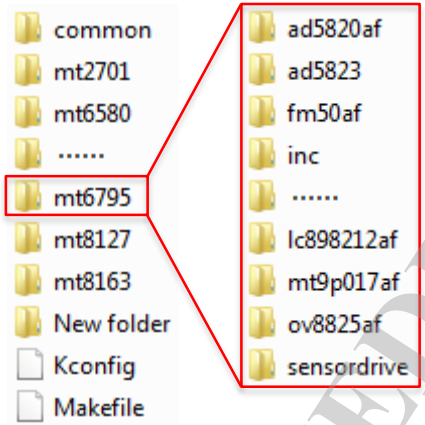
ProjectConfig.mk

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

Architecture

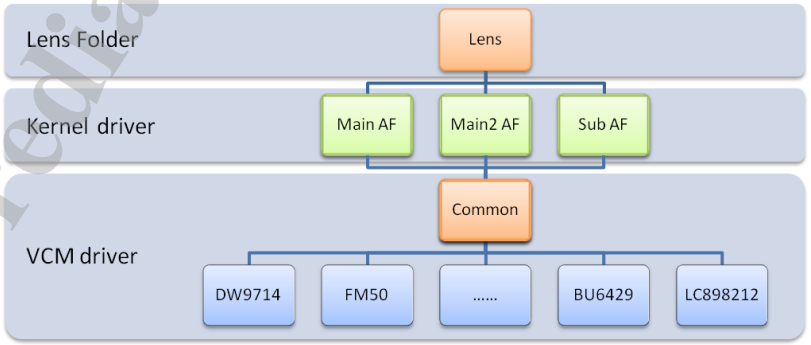
- Kernel 3.10

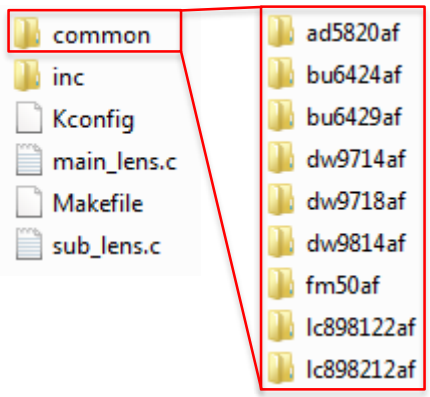




L

- Kernel 3.18

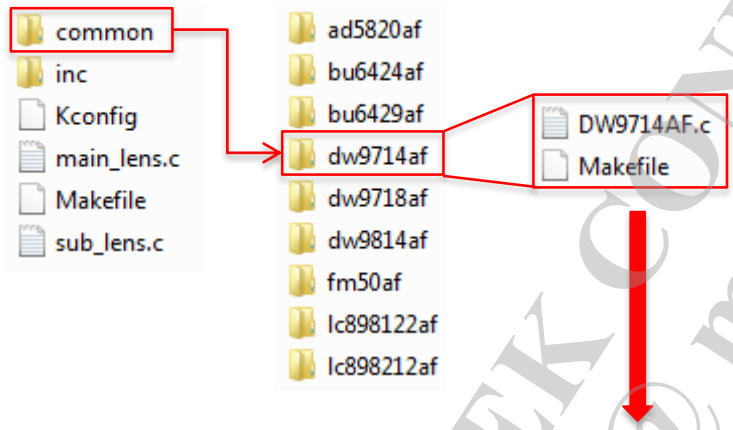




M

Step2-1

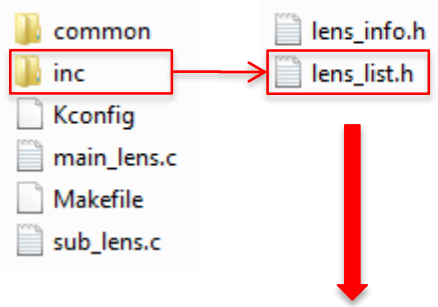
- Implement VCM driver




```
long DW9714AF_Ioctl(struct file *a_pstFile, unsigned int a_u4Command, unsigned long a_u4Param)
{
    ......
}
int DW9714AF_Release(struct inode *a_pstInode, struct file *a_pstFile)
{
    .......
}
void DW9714AF_SetI2Cclient(struct i2c_client *pstAF_I2Cclient, spinlock_t *pAF_SpinLock, int *pAF_Opened)
{
    g_pstAF_I2Cclient = pstAF_I2Cclient;
    g_pAF_SpinLock = pAF_SpinLock;
    g_pAF_Opened = pAF_Opened;
}
```

Step2-2

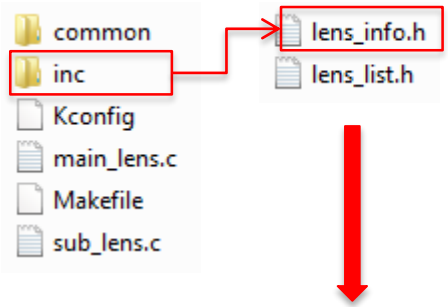
- Add driver's extern function




```
#ifdef CONFIG_MTK_LENS_BU6424AF_SUPPORT
extern void BU6424AF_SetI2Cclient(struct i2c_client *pstAF_I2Cclient, spinlock_t *pAF_SpinLock, int *pAF_Opened);
extern long BU6424AF_Ioctl(struct file *a_pstFile, unsigned int a_u4Command, unsigned long a_u4Param);
extern int BU6424AF_Release(struct inode *a_pstInode, struct file *a_pstFile);
#endif

#ifdef CONFIG_MTK_LENS_BU6429AF_SUPPORT
extern void BU6429AF_SetI2Cclient(struct i2c_client *pstAF_I2Cclient, spinlock_t *pAF_SpinLock, int *pAF_Opened);
extern long BU6429AF_Ioctl(struct file *a_pstFile, unsigned int a_u4Command, unsigned long a_u4Param);
extern int BU6429AF_Release(struct inode *a_pstInode, struct file *a_pstFile);
#endif

#ifdef CONFIG_MTK_LENS_DW9714AF_SUPPORT
extern void DW9714AF_SetI2Cclient(struct i2c_client *pstAF_I2Cclient, spinlock_t *pAF_SpinLock, int *pAF_Opened);
extern long DW9714AF_Ioctl(struct file *a_pstFile, unsigned int a_u4Command, unsigned long a_u4Param);
extern int DW9714AF_Release(struct inode *a_pstInode, struct file *a_pstFile);
#endif
```

Step2-3

- Add driver’s extern function



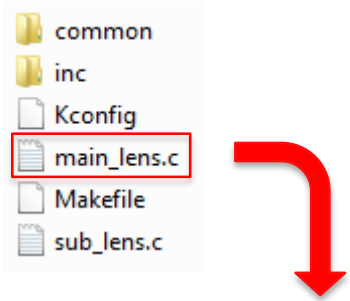
1. kernel-3.18\drivers\misc\mediatek\lens\inc\ lens_info.h

```
#define AFDRV_DW9714AF "DW9714AF"
#define AFDRV_DW9800WAF "DW9800WAF"  <---添加
#define AFDRV_DW9718AF "DW9718AF"
```

否则 main_lens.c 中编译报错,提示 AFDRV_DW9800WAF 未被定义.

Step2-4

- Add Driver to DrvList




```
static stAF_DrvList g_stAF_DrvList[MAX_NUM_OF_LENS] = {
    #ifdef CONFIG_MTK_LENS_BU6424AF_SUPPORT
    {1, AFDRV_BU6424AF, BU6424AF_SetI2Cclient, BU6424AF_Ioctl, BU6424AF_Release},
    #endif
    #ifdef CONFIG_MTK_LENS_BU6429AF_SUPPORT
    {1, AFDRV_BU6429AF, BU6429AF_SetI2Cclient, BU6429AF_Ioctl, BU6429AF_Release},
    #endif
    #ifdef CONFIG_MTK_LENS_DW9714AF_SUPPORT
    {1, AFDRV_DW9714AF, DW9714AF_SetI2Cclient, DW9714AF_Ioctl, DW9714AF_Release},
    #endif
    #ifdef CONFIG_MTK_LENS_DW9718AF_SUPPORT
    {1, AFDRV_DW9718AF, DW9718AF_SetI2Cclient, DW9718AF_Ioctl, DW9718AF_Release},
    #endif
    .........
};
```

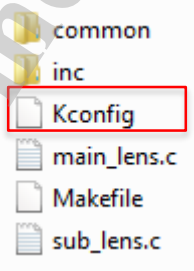
Step2-5

Add the config & Makefile setting of the AF,

For Example:DW9800WAF

Kconfig Modify

path:kernel-3.18\drivers\misc\mediatek\lens\Kconfig




```
config MTK_LENS_DW9800WAF_SUPPORT
 bool "DW9800WAF Lens Driver"
 default n
 help
  DW9800WAF Lens Driver
  This config is used to enable the corresponding
  lens driver for the camera sensor module
  Set as y if the driver is used in this project
```

Makefile Modify

path:kernel-3.18\drivers\misc\mediatek\lens\Makefile

obj-$(CONFIG_MTK_LENS_DW9800WAF_SUPPORT) += common/dw9800waf/

Step2-6

Configure **VCM IC power** in main_lens.c/sub_lens.c/main2_lens.c

AF power will be controlled by AFRegulatorCtrl()

```
void AFRegulatorCtrl(int Stage)
{

    if (Stage == 0) {
            if (regVCAMAF == NULL) {

            node = of_find_compatible_node(NULL, NULL, "mediatek,CAMERA_MAIN_AF");

            if (node) {
                    kd_node = lens_device->of_node;
                    lens_device->of_node = node;

                    if (strncmp(CONFIG_ARCH_MTK_PROJECT, "k71v1_64_bsp_fhdp", 17) == 0)
                            regVCAMAF = regulator_get(lens_device, "vldo28");
                    else
                            regVCAMAF = regulator_get(lens_device, "vcamaf");

            } else if (Stage == 1) {
                    if (regVCAMAF != NULL && g_regVCAMAFEn == 0) {
                            int Status = regulator_is_enabled(regVCAMAF);

                            LOG_INF("regulator_is_enabled %d\n", Status);

                            if (!Status) {
                                    Status = regulator_set_voltage(regVCAMAF, 2800000, 2800000);
```

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

Step 3-1
Add Lens Driver in List for HAL

- Add lens driver in 『 ***lenslist.cpp*** 』 for HAL.
  - alps\vendor\mediatek\proprietary\custom\[$***Platform***]\hal\lens\src

② extern variable

```
#if defined(LC898122AF)
extern PFUNC_GETLENSDEFAULT pLC898122AF_getDefaultData;
#endif

MSDK_LENS_INIT_FUNCTION_STRUCT LensList_main[MAX_NUM_OF_SUPPORT_LENS] =
{
  {DUMMY_SENSOR_ID, DUMMY_LENS_ID, "Dummy", pDummy_getDefaultData},
  #if defined(SENSORDRIVE)
    //{OV3640_SENSOR_ID, SENSOR_DRIVE_LENS_ID, "kd_camera_hw", pSensorDrive_getDefaultData},
  #endif
  #if defined(FM50AF)
    //{DUMMY_SENSOR_ID, FM50AF_LENS_ID, "FM50AF", pFM50AF_getDefaultData},
  #endif
  #if defined(DW9714AF)
    {IMX135_SENSOR_ID, DW9714AF_LENS_ID, "DW9714AF", pDW9714AF_getDefaultData},
  #endif
    #if defined(DW9718AF)
      {IMX135_SENSOR_ID, DW9718AF_LENS_ID, "DW9718AF", pDW9718AF_getDefaultData},
    #endif
  #if defined(AD5820AF)
    {OV5648MIPI_SENSOR_ID, AD5820AF_LENS_ID, "AD5820AF", pAD5820AF_getDefaultData},
  #endif
  #if defined(BU64745GWZAF)
    {S5K2P8_SENSOR_ID, BU64745GWZAF_LENS_ID, "BU64745GWZAF", pBU64745GWZAF_getDefaultData},
  #endif
  #if defined(LC898122AF)
    {IMX214_SENSOR_ID, LC898122AF_LENS_ID, "LC898122AF", pLC898122AF_getDefaultData},
  #endif
```

lenslist.cpp

① Add lens driver in lenslist.cpp

Step 3-2
Check Lens Driver ID is defined

- Check 『 ***camera_custom_lens.h*** 』 to define driver ID.
  - alps\vendor\mediatek\proprietary\custom\[$***Platform***]\hal\inc

camera_custom_lens.h

```
/* LENS ID */
#define DUMMY_LENS_ID                   0xFFFF
#define FM50AF_LENS_ID                  0x0001
#define MT9P017AF_LENS_ID               0x0002

#define SENSOR_DRIVE_LENS_ID            0x1000
#define GAF001AF_LENS_ID                0xFF01
#define GAF002AF_LENS_ID                0xFF02
#define GAF008AF_LENS_ID                0xFF08

#define OV8825AF_LENS_ID                0x0003
#define BU6429AF_LENS_ID                0x0004
#define BU6424AF_LENS_ID                0x0005
#define AD5823AF_LENS_ID                0x5823
#define DW9718AF_LENS_ID                0x9718
#define AD5820AF_LENS_ID                0x5820
#define DW9714AF_LENS_ID                0x9714
#define LC898122AF_LENS_ID              0x9812
#define BU64745GWZAF_LENS_ID            0xFCE9
```

① Check lens driver ID is defined or not

Permission

- Config Modify
- Kernel Modify
- Hal Modify
- Permission Modify

1) alps\device\mediatek\common\sepolicy\device.te

```
type DW9714AF_device, dev_type;
type LC898122AF_device, dev_type;
type LC898212AF_device, dev_type;
type BU6429AF_device, dev_type;
type BU64745GWZAF_device, dev_type;
```

名称为$(AFName)_device
要和设备中注册的设备匹配

2） alps\device\mediatek\common\sepolicy\file_contexts

```
/dev/FM50AF(/.*)? u:object_r:FM50AF_device:s0
/dev/DW9714AF(/.*)? u:object_r:DW9714AF_device:s0
/dev/LC898122AF(/.*)? u:object_r:LC898122AF_device:s0
/dev/LC898212AF(/.*)? u:object_r:LC898212AF_device:s0
/dev/BU6429AF(/.*)? u:object_r:BU6429AF_device:s0
/dev/BU64745GWZAF(/.*)? u:object_r:BU64745GWZAF_device:s0
```

Permission

3) alps\device\mediatek\common\sepolicy\mediaserver.te

```
allow mediaserver block_device:dir search;
allow mediaserver FM50AF_device:chr_file { read write ioctl open };
allow mediaserver AD5820AF_device:chr_file { read write ioctl open };
allow mediaserver DW9714AF_device:chr_file { read write ioctl open };
allow mediaserver LC898122AF_device:chr_file { read write ioctl open };
allow mediaserver LC898212AF_device:chr_file { read write ioctl open };
allow mediaserver BU6429AF_device:chr_file { read write ioctl open };
allow mediaserver BU64745GWZAF_device:chr_file { read write ioctl open };
allow mediaserver DW9718AF_device:chr_file { read write ioctl open };
```

4). device\mediatek\$Project\init.project.rc

#Camera

chmod 0660 /dev/MAINAF

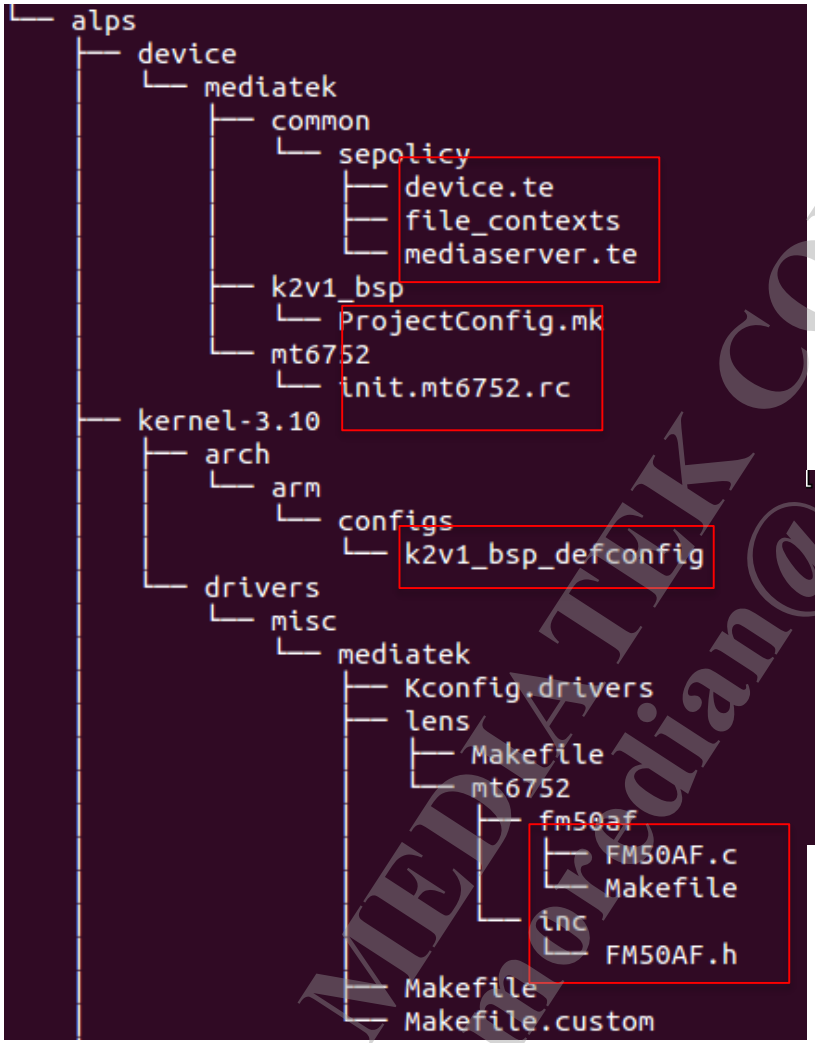
chmod 0660 /dev/FM50AF

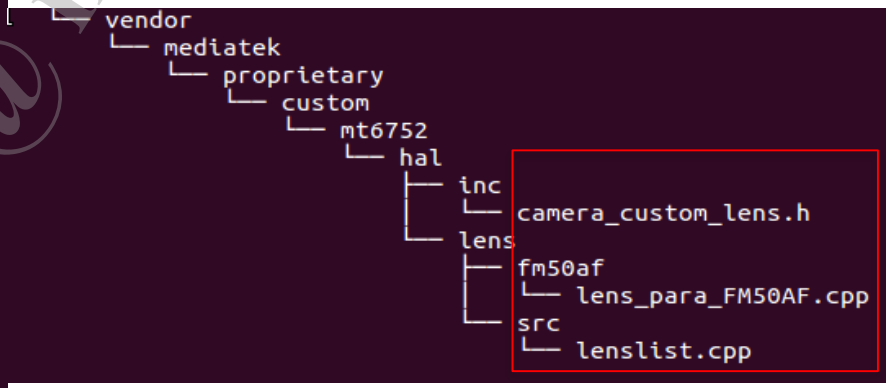
chmod 0660 /dev/XXXAF

chown system camera /dev/MAINAF
chown system camera /dev/FM50AF
chown system camera /dev/XXXAF

ANDROID L AF Porting Guide

L版本以fm50af 为例，涉及到的相关文件如下





CONFIDENTIAL B

MEDIATEK

AF_Driver_Porting_MT6752_MT6732_L.pptx

Config和Permision请参考此文档

AF Driver

FM50AF.h

```
#define FM50AF_MAGIC 'A'
/* IOCTRL(inode * ,file * ,cmd ,arg ) */


/* Structures */
typedef struct {
/* current position */
    u32 u4CurrentPosition;
/* macro position */
    u32 u4MacroPosition;
/* Infiniti position */
    u32 u4InfPosition;
/* Motor Status */
    bool bIsMotorMoving;
/* Motor Open? */
    bool bIsMotorOpen;
/* Support SR? */
    bool bIsSupportSR;
} stFM50AF_MotorInfo;

/* Control commnad */
/* S means "set through a ptr" */
/* T means "tell by a arg value" */
/* G means "get by a ptr" */
/* Q means "get by return a value" */
/* X means "switch G and S atomically" */
/* H means "switch T and Q atomically" */
#define FM50AFIOC_G_MOTORINFO _IOR(FM50AF_MAGIC, 0, stFM50AF_MotorInfo)

#define FM50AFIOC_T_MOVETO _IOW(FM50AF_MAGIC, 1, u32)

#define FM50AFIOC_T_SETINFPOS _IOW(FM50AF_MAGIC, 2, u32)

#define FM50AFIOC_T_SETMACROPOS _IOW(FM50AF_MAGIC, 3, u32)
```

```
#include "MediaTypes.h"
#include "mcu_drv.h"
#include "lens_drv.h"
#include "FM50AF.h"
```

```
err = ioctl(m_fdMCU,FM50AFIOC_T_MOVETO,(unsigned long)a_i4FocusPos);
if (err < 0) {
    DRV_ERR("[moveMCU] ioctl - FM50AFIOC_T_MOVETO, error %s",  strerror(errno));
    return err;
}
```

AF Driver Function List

FM50AF.c

Function Name	作用
FM50AF_i2C_init	初始化i2c、注册平台设备 驱动
FM50AF_i2C_exit	platform_driver_unregister
AF_probe	平台设备注册后，add I2C driver
AF_remove	delete I2C driver
AF_suspend & AF_resume	Reserve
AF_i2c_probe	Register AF driver
Register_AF_CharDrv	注册AF字符设备驱动
AF_Open	Set open flag
AF_Release	Reset open flag
AF_Ioctl	Ioctl- moveAF getAFinfo SetInf SetMacro
moveAF	move lens
s4AF_WriteReg	Write lens position to vcm IC
s4AF_ReadReg	Get Lens position value from vcm ic

AF Driver

Tuning File lens_para_FM50AF.cpp

文件内容含义请参考Tuning 相关文档

```
//Version
NVRAM_CAMERA_LENS_FILE_VERSION,

// Focus Range NVRAM
{0, 1023},

// AF NVRAM
{
    // ------------------------------ sAF_Coef ---------------------------
    {
        {
            200, // i4Offset
            13,  // i4NormalNum
            13,  // i4MacroNum
            0,   // i4InfIdxOffset
            0,   // i4MacroIdxOffset
            {
                0,  20,  45,  70,  95, 120, 150, 180, 220, 260,
                305, 355, 405,   0,   0,   0,   0,   0,   0,   0,
                0,   0,   0,   0,   0,   0,   0,   0,   0,   0
            }
        },
        15, // i4THRES_MAIN;
        10, // i4THRES_SUB;
        1,  // i4AFC_FAIL_CNT;
        0,  // i4FAIL_POS;

        4,  // i4INIT_WAIT;
        {500, 500, 500, 500, 500}, // i4FRAME_WAIT
        0,  // i4DONE_WAIT;
    },
    // ------------------------------ sVAFC_Coef ------------------------
    {
        {
            210, // i4Offset
            21,  // i4NormalNum
```

AF Driver

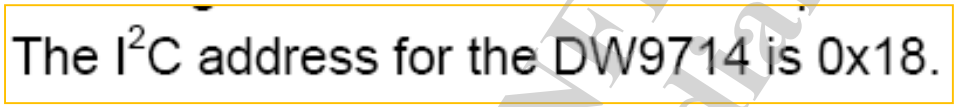
对于不同的VCM IC，AF Driver的不同地方主要体现在

1. AF Driver Name
2. I2C Slave address
3. AF Register ID
4. Platform driver name
5. AF driver class name
6. s4AF_WriteReg 与s4AF_ReadReg
7. AF_Open

AF Driver

以DW9714为例来实现AF Driver dw9714af

1. AF Driver Name，把FM50AF重命名为DW9714AF
2. 从DW9714 DataSheet可知 I2C Slave address 为 0x18

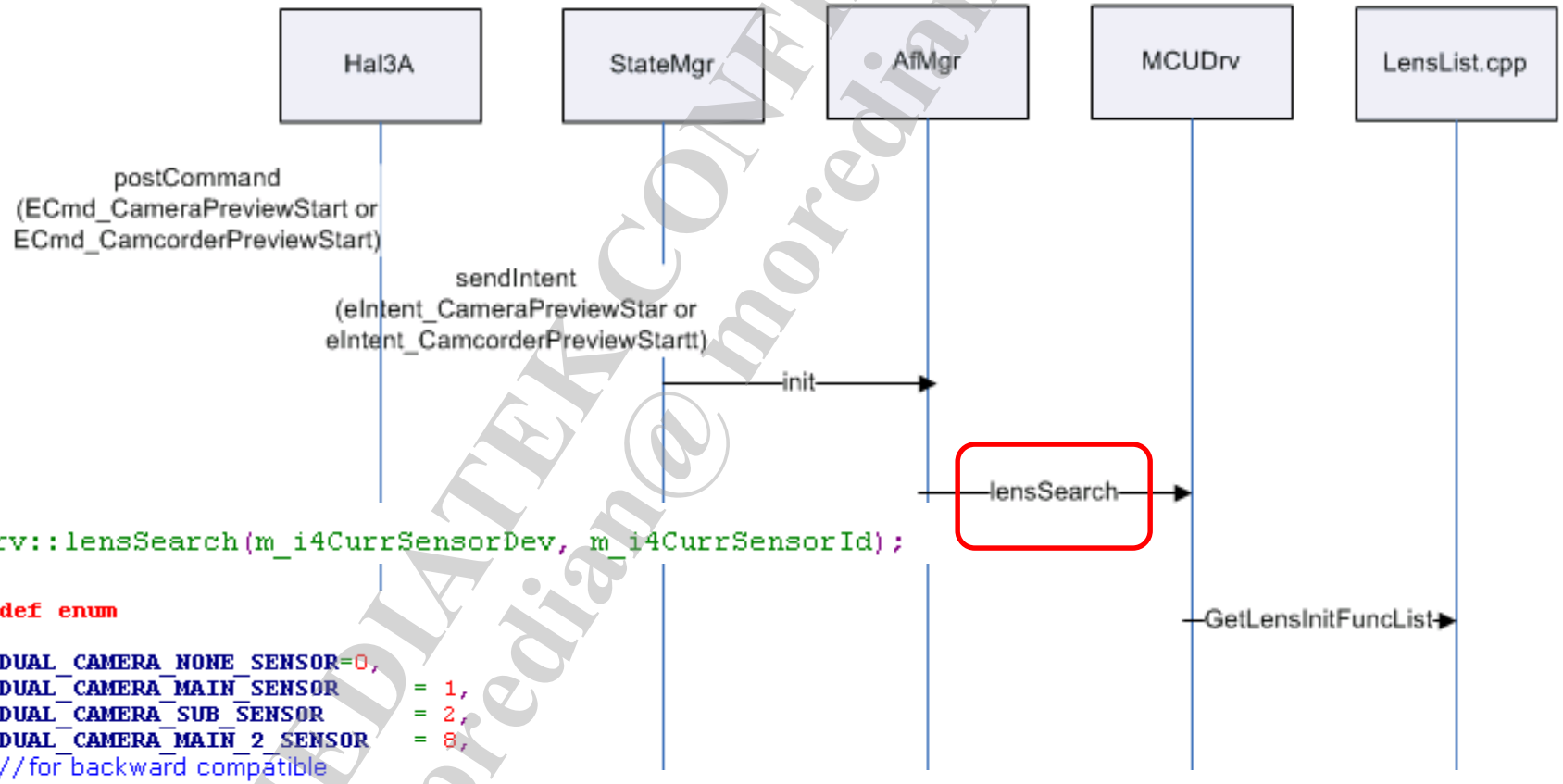


3. AF Register ID，这个ID是i2c driver注册时的唯一标示码，只要保证i2c bus number上唯一就可以，如果不做AF Driver兼容的话，可以和I2C Slave address值相同
4. Platform driver name，建议命名为 "lens_actuator_dw9714af"
5. AF driver class name，建议命名为 "actuatordrv_dw9714af“
6. VCM的init setting等在moveAF()开头进行

Search Lens

This will be called at enter camera

alps\vendor\mediatek\proprietart\platform\$Platform$\hardware\mtkcam\core\featureio\drv\lens\mcu_drv.cpp




```
MCUDrv::lensSearch(m_i4CurrSensorDev, m_i4CurrSensorId);
```

```
typedef enum
{
    DUAL_CAMERA_NONE_SENSOR=0,
    DUAL_CAMERA_MAIN_SENSOR      = 1,
    DUAL_CAMERA_SUB_SENSOR       = 2,
    DUAL_CAMERA_MAIN_2_SENSOR    = 8,
    //for backward compatible
    DUAL_CAMERA_MAIN_SECOND_SENSOR = 8,
    //DUAL_CAMERA_SUB_2_SENSOR  = 16,
    DUAL_CAMERA_SENSOR_MAX
} CAMERA_DUAL_CAMERA_SENSOR_ENUM;
```

Search Lens

lensSearch Function实现了lens 匹配过程，根据当前sensor是main、sub或main2分别从lenslist.cpp 文件中的数组LensList_main[]、LensList_sub或LensList_main2中搜索符合要求的AF driver

```
MSDK_LENS_INIT_FUNCTION_STRUCT LensList_main[MAX_NUM_OF_SUPPORT_LENS] =
{
    {DUMMY_SENSOR_ID, DUMMY_LENS_ID, "Dummy", pDummy_getDefaultData},
    #if defined(SENSORDRIVE)
        {S5K4ECGX_SENSOR_ID, SENSOR_DRIVE_LENS_ID, "kd_camera_hw", pSensorDrive_getDefaultData},
    #endif
    #if defined(FM50AF)
        {OV13850_SENSOR_ID, FM50AF_LENS_ID, "FM50AF", pFM50AF_getDefaultData},
    #endif
    #if defined(LC898122AF)
        {IMX214_SENSOR_ID, LC898122AF_LENS_ID, "LC898122AF", pLC898122AF_getDefaultData},
    #endif
};
MSDK_LENS_INIT_FUNCTION_STRUCT LensList_sub[MAX_NUM_OF_SUPPORT_LENS] =
{
    {DUMMY_SENSOR_ID, DUMMY_LENS_ID, "Dummy", pDummy_getDefaultData},
    #if defined(SENSORDRIVE)
        {S5K4ECGX_SENSOR_ID, SENSOR_DRIVE_LENS_ID, "kd_camera_hw", pSensorDrive_getDefaultData},
    #endif
    #if defined(AD5820AF)
        {OV5648MIPI_SENSOR_ID, AD5820AF_LENS_ID, "AD5820AF", pAD5820AF_getDefaultData},
    #endif

};
```

Search Lens

匹配规则：

匹配结果为m_u4CurrLensIdx，初始值为0，即默认指向第一个元素。

第一遍搜索：m_u4CurrLensIdx指向LensId为 DUMMY_LENS_ID、SENSOR_DRIVE_LENS_ID或FM50AF_LENS_ID 的元素，结果m_u4CurrLensIdx指向数组LensList[]中符合条件的最后一个元素。

第二遍搜索：看数组LensList[]中是否有SensorId 和当前的要找Lens driver的sensor的ID相等的元素，如果有则结果为符合条件的第一个元素，没有的话则结果为第一遍搜索到的结果。

AF Debug SOP & Case Study

Debug SOP

- 1 首先看进入camera应用时的android log，搜索关键字“LensMCUlensSearch”
  - 搜索如下面一段log
  - LensMCU : LensMCUlensSearch() - E
  - LensMCU : LensMCU[CurrSensorDev]0x0001
  - LensMCU : LensMCU[LensInitTable-0][SensorId]0xffff,[LensId]0xffff
  - LensMCU : LensMCU[LensInitTable-1][SensorId]0x4800,[LensId]0x0002
  - LensMCU : LensMCU[LensInitTable-2][SensorId]0x0000,[LensId]0x0000
  - LensMCU : LensMCU[LensInitTable-3][SensorId]0x0000,[LensId]0x0000
  - LensMCU : LensMCU[CurrLensIdx]1
  - LensDrv : init() [m_userCnt]1
  - LensDrv :
  - LensDrv : [Lens Driver]/dev/XXXAF

CurrSensorDev:
0x0001 表示main sensor，
0x0002表示 sub sensor

Lenslist.cpp 中的数组 Lenslist_XXX[] 运行时的前四个Lens Driver信息

CurrLensIdx: 匹配到的Lens Driver在数组中的index

匹配的Lens Driver的LensDrvName加上前缀/dev/，就是对应的设备文件，如果是YUV sensor，这边应该是/dev/kd_camera_hw

  - 通过看这边的log就可以发现很多问题
  - 1.1 看 CurrSensorDev 的值是否为0x0001，如果这边的值为0x0002，即为sub sensor找lens driver，则存在问题
  - 1.2 再看最后一行[Lens Driver]/dev/XXXAF，这行的意思是为当前Sensor匹配到了Lens Driver，并去打开设备驱动文件/dev/XXXAF

    如果 XXXAF不是当前Sensor需要找的Lens Driver，请确认ProjectConfig.mk 和lenslist.cpp文件是否有正确配置

Debug SOP

– 1.3 看log [Lens Driver]/dev/XXXAF 后面是否有打开设备驱动文件 /dev/XXXAF 异常的log

1.3.1 如果显示如下log

LensDrv : Err: 112:, error opening /dev/XXXAF: No such file or directory
IspHal : [init]Err(320):mpMcuDrv->init() fail

表示找不到文件 /dev/XXXAF，即Lens Driver XXXAF 没有配置好或没有注册上

A：检查配置文件 ProjectConfig.mk

配置项CUSTOM_HAL_LENS 和 CUSTOM_KERNEL_LENS 中包含xxxaf，这里应该为小写，为相应AF Driver的文件夹名称

可以通过查看编译out目录 alps\kernel-3.10\mediatek\custom\out\kernel\lens 是否存在对应的AF driver相关文件

Debug SOP

B：配置了多颗 AF Driver ， AF Driver注册存在冲突

同时配置的AF Driver要求在同一个I2C Bus number下i2c register id 唯一， Platform driver name 及 AF driver class name 字符串也不能相同

```
// in K2, main=3, sub=main2=1
#define LENS_I2C_BUSNUM 3

#define AF_DRVNAME "FM50AF"
#define I2C_SLAVE_ADDRESS        0x18
#define I2C_REGISTER_ID          0x18
#define PLATFORM_DRIVER_NAME "lens_actuator_fm50af"
#define AF_DRIVER_CLASS_NAME "actuatordrv_fm50af"

static struct i2c_board_info __initdata kd_lens_dev={ I2C_BOARD_INFO(AF_DRVNAME, I2C_REGISTER_ID
```

C：AF Driver的i2c注册ID 和同i2c bus number 上的其他i2c 设备注册ID 相冲突

Debug SOP

1.3.2 如果显示如下log

LensDrv : Err: 112:, error opening /dev/XXXAF: Permission denied
IspHal : [init]Err(320):mpMcuDrv->init() fail

表示没有权限打开文件 /dev/XXXAF，请检查文件 init.$platform.rc 是否有正确修改

这个问题可以参考 FAQ03447 [Camera Drv]使用 FM50AF 之外的Lens Driver的修改

Debug SOP

- 2 查看包含进入camera应用的kernel log，搜索关键字“XXXAF”，看是否有类似下面的Len Position Trace log
  - [1504:Binder_1]s4XXXAF_WriteReg =0x64
  - [1504:Binder_1]s4XXXAF_WriteReg =0xf2
  - [1504:Binder_1]s4XXXAF_WriteReg =0x123
  - [1504:Binder_1]s4XXXAF_WriteReg =0x158
  - 如果搜索不到Len Position Trace log请按以下步骤处理
  - 2.1 确认 XXXAF.c 文件中的 s4XXXAF_WriteReg 函数中加了Lens Position 的 trace，如下图

```
static int s4XXXAF_WriteReg(u16 a_u2Data)
{
    int  i4RetValue = 0;

    char puSendCmd[2] = {(char)(a_u2Data >> 4) , (char)(((a_u2Data & 0xF) << 4)+0xF)};

    XXXAFDB("[s4XXXAF_WriteReg] a_u2Data: 0x%x \n", a_u2Data);

    i4RetValue = i2c_master_send(g_pstXXXAF_I2Cclient, puSendCmd, 2);

    if (i4RetValue < 0)
    {
        XXXAFDB("[XXXAF] I2C send failed!! \n");
        return -1;
    }

    return 0;
}
```

添加 Lens Position 的Trace

Debug SOP

– 2.2 查看进入camera应用的android log，搜索关键字“focus-mode”
找到如下log

focus-mode=continuous-picture;focus-mode-values=auto,continuous-picture,continuous-video,macro,infinity,manual,fullscan;

如果内容如下，表示有异常

focus-mode=infinity;
focus-mode-values=infinity;

focus-mode是指当前对焦模式，
focus-mode-values指所以支持的对焦模式。
auto：自动对焦
continuous-picture：连续对焦
macro：微距对焦
infinity：无限远，即不对焦
continuous-video：录像时对焦
Manual：手动对焦
Fullscan：工模下full scan使用

请检查Feature配置文件 config.ftbl.xxx_mipi_raw.h，查看AF 相关的配置项是否正确。
下面AF Feature配置就是认为没有AF功能

```
...
FTABLE_SCENE_DEP()
    //=========================================================
#if 1
    //  Focus Mode
    FTABLE_CONFIG_AS_TYPE_OF_DEFAULT_VALUES(
        KEY_AS_(MtkCameraParameters::KEY_FOCUS_MODE),
        SCENE_AS_DEFAULT_SCENE(
            ITEM_AS_DEFAULT_(MtkCameraParameters::FOCUS_MODE_INFINITY),
            ITEM_AS_VALUES_(
                MtkCameraParameters::FOCUS_MODE_INFINITY,
            )
        ),
        //.....................................................
    )
#endif
```

Debug SOP

- 3 进行到这一步表示前面一步中Len Position Trace log已抓取到，然后看写Len Position 寄存器的操作是否存在I2C 相关的Error（ps：需要确认下客户有没有移除掉I2C Log）
  - 3.1 如果存在I2C Error Log，表明AF I2C 不通。
    - 3.1.1 AF是否有正确供电，方便量测的话，可以量测一下
    - 3.1.2 是否有AF Enable Pin，存在的话需要正确拉相应的GPIO
    - 3.1.3 Slave address是否正确
    - 3.1.4 I2C 总线是否正确
    - 3.1.5 如果上面步骤检查了都没有问题的话，联系模组厂确认模组是否存在硬件上的问题或反馈的VCM IC等信息的准确性。
  - 3.2 没有I2C Error log，AF I2C 是通的，但是VCM 马达没有动，继续下一步。

AF Debug SOP & Case Study

一、AF不动

1. （HW）马达的检验、量测AF_VDD
2. 在/dev下看AF驱动是否注册上
3. feature table的配置
4. sensor与lens的匹配
5. VCM driver的实现
6. 权限

二、AF有异响

判断是撞击声或摩擦声，基本处理方法是分段移动马达

1. 在open和release处分步移动马达
2. 请模组厂优化

三、AF不准(参考lens_para demo code)

FAQs

FAQ02642 [Camera Drv]Lens Driver(AF Driver)相关问题

FAQ03008 [Camera Drv]Lens Driver如何兼容

FAQ03447 [Camera Drv]使用 FM50AF 之外的Lens Driver的修改

FAQ04693 [Camera Drv]系统是如何为Sensor匹配Lens Driver

FAQ06464 [Camera Drv]Sub Camera支持AF功能如何修改

FAQ08599 [Camera Drv]如何为RAW Sensor实现Lens Driver

FAQ04468 [Camera Drv]YUV sensor如何添加AF

FAQ13483 L版本上AF Driver open及release函数要求

FAQ11713 [Camera Drv]AF Driver open fail导致进入相机画面很黑

FAQ08616 [Camera Drv]VCM IC Datasheet解读

FAQ14420 无AF时lensSearch匹配问题

FAQ14263 L版本如何开通AF权限

FAQ14363 L版本AF不动常见问题