密级状态：绝密(　　)　　秘密(　　)　　内部资料()　　公开(√)

RK_CIF10_User_Manual

文件状态： ［　］草稿 ［　］正式发布 ［√］正在修改	文件标识：	Company-Project-RD-UR
	当前版本：	2.0
	作　　者：	邓达龙、钟以崇、欧阳亚凤、张云龙
	完成日期：	2017-12-11

福州瑞芯微电子股份有限公司
Fuzhou Rockchips Electronics Co . , Ltd

历 史 版 本

版本	日 期	描 述	作 者	审核
V2.0	2017-12-12	添加适用平台章节并更新文档名称	张云龙	

目 录

1 适用平台

该文档适用于 rk3188/rk3066/rk2918 等 kernel 2.6.32 版本的平台。

2 目录说明

RK CAMERA 相关驱动文件目录，下面简单对文件结构简单说明如下：

```
drivers/media/video:
  |___ rk29_camera_oneframe.c  VIP/CIF Driver,vip/cif 控制器单帧模式
  |___ rk30_camera_oneframe.c
  |___ rk30_camera_pingpong.c
  |___ generic_sensor.c  generic_sensor.h   rockchip Sensor 通用驱动
  |___ ov2655.c ov5642.c ov2659.c ov5640.c   OV 公司 sensor 驱动
  |___ mt9p111.c mt9d112.c mt9m112.c   Micron(Aptina)公司 sensor 驱动
  |___ s5k6aa.c s5k5ca.c  Samsung 公司 sensor 驱动
  |___ soc_camera.c soc_camera.h   soc_camera 设备驱动
  |___ v412-xxxxx.c  v412 设备驱动
  |___ rk29_camera.c  RK29/RK30 camera IO 以及设备注册相关代码
  |___ rk30_camera.c
arch/arm/plat-rk/plat:
  |___ rk_camera.c    IO 操作代码
arch/arm/plat-rk/plat/include:
  |___ rk_camera.h    RK camera 共用定义头文件
arch/arm/mach-rkxx:
       |___ board_rk29sdk.c  板级配置文件
       |___ include/mach/include/rk29_camera.h  各芯片平台 camera 模块头文件
       |___ include/mach/include/rk30_camera.h
```

注意：

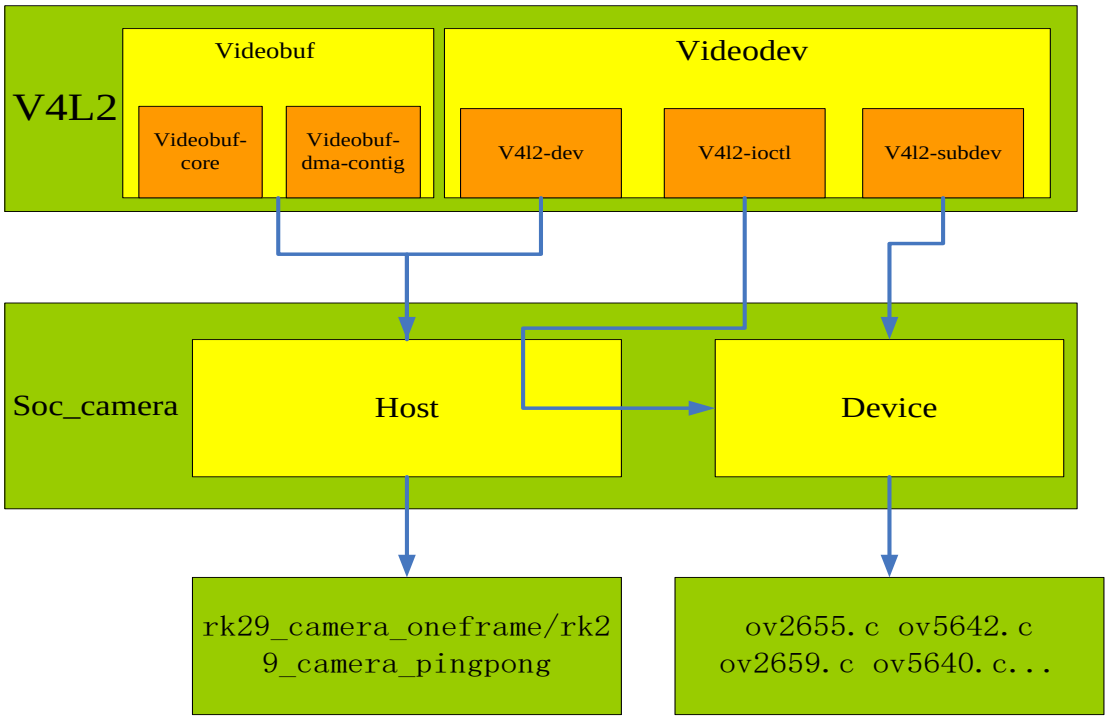
以 SensorName.c 命名的驱动文件，有可能是新版本的也有可能是旧版本的，区分只能是打开驱动文件：

```
#include "generic_sensor.h"
/*
*         Driver Version Note
*v0.0.1: this driver is compatible with generic_sensor
*v0.1.1:
*           add sensor_focus_af_const_pause_usr_cb;
*/
```

static int version = KERNEL_VERSION(0,1,3);

对于具备新版本驱动的 sensor，sdk 同样保留了旧版本的驱动，命名规则为 SensorName_old.c

3 camera 驱动结构框图



4 Camera Host（VIP Controller）驱动简介

Camera Host 驱动主要实现如下：

G:\参考资料\Andriod & Linux\R

1、videobuf 回调以及控制；参考文档

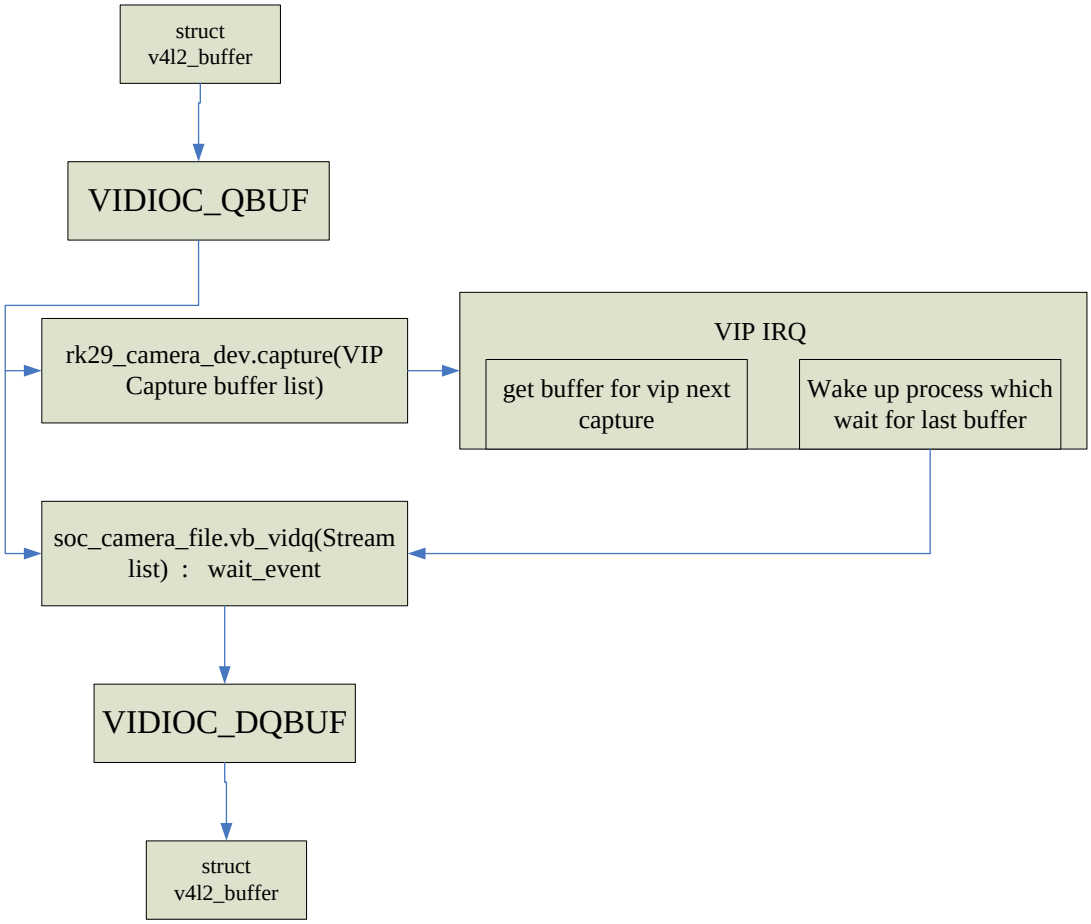
2、VIP Controller 设置；

3、Camera 休眠唤醒；

4、IPP scale 控制；

4.1 vidobuf 回调以及控制

由参考文档得知，使用 videobuf 机制必须实现 videobuf_queue_ops 结构体中的 4 个回调函数。驱动对 videobuf 的控制流程如下：



4.2 VIP Controller 设置

VIP 控制器的设置主要涉及：VIP 各工作时钟控制、 VIP 输出时钟（Sensor 工作时钟）输出、VIP 采集时序极性控制；

以上控制主要集中在以下函数中：

rk29_camera_set_bus_param;

rk29_camera_setup_format;

rk29_camera_set_fmt;

4.3 Camera 休眠唤醒；

VIP 控制器在休眠唤醒中需要寄存器进行备份设置，在唤醒时将备份的寄存器值重新恢复到寄存器中。

4.4 IPP Scale 控制（RK2918/RK30XX）

Sensor 能够输出的分辨率不一定能够完全满足用户的需求，这个时候就必须对 sensor 的输出图像进行 scale。

在使用到 IPP scale 时，vip 采集的 buf 就不能是用户提供的 videobuf，vip 驱动必须获取一段 buf 作为采集用，采集结束后将该段 buf 中的数据利用 IPP 进行 scale 处理，同时将输出到用户指定的 videobuf 中，最后唤醒因获取该 videobuf 而睡眠的进程。IPP 处理必须在内核线程中进行处理。

IPP 操作还相关到数码变焦的实现，参见“Camera Digital Zoom”一节。

5 sensor 驱动简介

原来 Sensor 驱动的配置等信息，麻烦参考该文档的 v3.0.2 及其以前的版本，以下说明是针对统一后的 Sensor 驱动；

注意：

目前的驱动代码兼容以前的 Sensor 驱动，以前的 Sensor 驱动代码可以不作任何修改即可直接在当前版本中运行；

Sensor 驱动功能基本类似，所以在驱动中具备多个配置宏，下面针对这些配置宏说明如下

5.1 如何编写新 sensor 驱动（针对 rockchip 平台）

```
#define SENSOR_NAME                 RK29_CAM_SENSOR_OV5640
// 定义驱动的 Sensor 名字， 该名字与设置注册时用到的名字一致

#define SENSOR_V4L2_IDENT           V4L2_IDENT_OV5640
// 定义 Sensor 在 v4l2 中的编号，参考 v4l2-chip-ident.h

#define SENSOR_ID 0x5640
// 定义 Sensor 的硬件识别号，该识别号用于检测硬件，如果 Sensor 在不同批次存在多个 i d 号，那么将多余的 i d 号码填写在以下数组中
static unsigned int SensorChipID[] = {SENSOR_ID};

#define SENSOR_BUS_PARAM
// 定义 Sensor 输出信号的特性，比如 data 的采集边沿（SOCAM_PCLK_SAMPLE_RISING/SOCAM_PCLK_SAMPLE_FALLING）、hsync 信号的有效电平(SOCAM_HSYNC_ACTIVE_HIGH/SOCAM_HSYNC_ACTIVE_LOW)，vsync 信号的有效电平（SOCAM_VSYNC_ACTIVE_HIGH/ SOCAM_VSYNC_ACTIVE_LOW）等；

#define SENSOR_PREVIEW_W                        800
#define SENSOR_PREVIEW_H                        600
// 定义该 Sensor 驱动预览序列的分辨率

#define SENSOR_PREVIEW_FPS                      15000       // 15fps
#define SENSOR_FULLRES_L_FPS                    7500        // 7.5fps
#define SENSOR_FULLRES_H_FPS                    7500        // 7.5fps
#define SENSOR_720P_FPS                         30000
#define SENSOR_1080P_FPS                        15000
// 定义该 Sensor 驱动预览序列、全分辨率序列、720p、1080p 序列的帧率，注意是帧率*1000

#define SENSOR_REGISTER_LEN         2           // sensor register address bytes
```

```
#define SENSOR_VALUE_LEN        1        // sensor register value bytes
```

// 定义该 Sensor 寄存器地址长度以及寄存器值的长度，这 2 个定义将决定 sensor_write 以及 sensor_read 函数宏的定义，需要注意。如果说 sensor 的寄存器值以及寄存器地址的长度任一不固定的话，那么这两个宏定义需要赋值成 0，驱动中也不能直接使用 sensor_write 以及 sensor_read 函数宏，i2c 的读写只能直接使用

```
00264: extern int generic_sensor_write(struct i2c_client *client,struct rk_sensor_reg* sensor_reg);
00265: extern int generic_sensor_read(struct i2c_client *client, struct rk_sensor_reg* sensor_reg);
```

如何配置以及增加 Sensor 驱动的功能

其中原版本 Sensor 驱动的功能配置宏，例如：CONFIG_SENSOR_WhiteBalance…等，这些宏的配置目前更改由全局变量的赋值来处理：

```
static unsigned int SensorConfiguration = (CFG_WhiteBalance|CFG_Effect
                                           |CFG_Scene|CFG_Focus
                                           |CFG_FocusZone);
```

可配置的宏定义详见 drivers/media/video/generic_sensor.h ：

```
00053: #define CFG_WhiteBalance                (1<<0)
00054: #define CFG_Brightness                  (1<<1)
00055: #define CFG_Contrast                    (1<<2)
00056: #define CFG_Saturation                  (1<<3)
00057: #define CFG_Effect                      (1<<4)
00058: #define CFG_Scene                       (1<<5)
00059: #define CFG_DigitalZoom                 (1<<6)
00060: #define CFG_Focus                       (1<<7)
00061: #define CFG_FocusContinues              (1<<8)
00062: #define CFG_FocusZone                   (1<<9)
00063: #define CFG_FocusRelative               (1<<10)
00064: #define CFG_FocusAbsolute               (1<<11)
00065: #define CFG_FACE_DETECT                 (1<<12)
00066: #define CFG_Exposure                    (1<<13)
00067: #define CFG_Flash                       (1<<14)
00068: #define CFG_Mirror                      (1<<15)
00069: #define CFG_Flip                        (1<<16)
```

其中这些功能打开后，generic_sensor.c 功能对应的代码即生效，相应功能打开后，Sensor 驱动中对应的功能序列必须填写，比如针对 CFG_WhiteBalance 功能，以下数组就必须被填写：

```
sensor_WhiteB_Auto
sensor_WhiteB_Cloudy
sensor_WhiteB_ClearDay
sensor_WhiteB_TungstenLamp1
sensor_WhiteB_TungstenLamp2
```

如果想针对该 Sensor 驱动实现除了以上定义的功能，可以将需要实现的 control 以及 menus 定义在以下结构体中：

```
00704: /*
00705: * User could be add v4l2_querymenu in sensor_controls by new_usr_v4l2menu
00706: */
00707: static struct v4l2_querymenu sensor_menus[] =
00708: {
00709: };
00710: /*
00711: * User could be add v4l2_queryctrl in sensor_controls by new_user_v4l2ctrl
00712: */
00713: static struct sensor_v4l2ctrl_usr_s sensor_controls[] =
00714: {
00715: };
00716:
```

如果定义的 control id 号与 generic_sensor.h 已经定义的功能的 id 号一致，那么以该结构体定义的为准；

关于如何填写 Sensor 驱动各个序列，麻烦注意以下几点：

1）、序列的结尾必须填写一个结束符， 可以直接使用以下宏定义

#define SensorEnd {SEQCMD_END,0x00,0x00,0x00}

2）、序列中需要设置多个寄存器，在某些寄存器写完之后，希望延时一定时间之后在进行序列中后续寄存器的写入，那么可以使用以下宏定义：

#define SensorWaitMs(a) {SEQCMD_WAIT_MS,a,0x00,0x00}

#define SensorWaitUs(a) {SEQCMD_WAIT_US,a,0x00,0x00}

3）、如果该序列是在 Sensor 已经 StreamOn 之后写入，例如 AF 固件等，但是在 StreamOff 时，又希望及时中断输入，那么可以在序列的开始使用以下宏定义：

#define SensorStreamChk {SEQCMD_STREAMCHK,0,0,0}

4）、序列采用的是结构体数组方式，结构体声明如下：

```
00080: struct rk_sensor_reg {
00081:     unsigned int reg;
00082:     unsigned int val;
00083:     unsigned int reg_mask;
00084:     unsigned int val_mask;
00085: };
00086:
```

其中：

Reg： 寄存器地址（低 3 个字节有效） 或是 命令字（最高字节）

Val： 寄存器值；

Reg_mask： 寄存器地址有效字节；

Val _mask: 寄存器值有效字节

可以参考以下几个宏定义：

```
00044: #define SensorReg1Val1(a,b)     {a,b,0xff,0xff}
00045: #define SensorReg2Val1(a,b)     {a,b,0xffff,0xff}
00046: #define SensorReg2Val2(a,b)     {a,b,0xffff,0xffff}
00047:
```

Sensor 驱动文件中，各个回调函数的实现：

1）、

```
00891: /*
00892: *****************************************************************
00893: * Following is callback
00894: * If necessary, you could coding these callback
00895: *****************************************************************
00896: */
00897: /*
00898: * the function is called in open sensor
00899: */
00900: static int sensor_activate_cb(struct i2c_client *client)
```

```
00914: /*
00915: * the function is called in close sensor
00916: */
00917: static int sensor_deactivate_cb(struct i2c_client *client)
```

这 2 个函数分别在打开、关闭 Sensor 时会被调用；

2）、

```
01093: /*
01094: * the function is called before sensor register setting in VIDIOC_S_FMT
01095: */
01096: static int sensor_s_fmt_cb_th(struct i2c_client *client,struct v4l2_mbus_framefmt *mf, bool capture)
01097: {
01098:     return 0;
01099: }
01100: /*
01101: * the function is called after sensor register setting finished in VIDIOC_S_FMT
01102: */
01103: static int sensor_s_fmt_cb_bh (struct i2c_client *client,struct v4l2_mbus_framefmt *mf, bool capture)
01104: {
01105:     return 0;
01106: }
```

以上 2 个回调函数分别在 Sensor 进行分辨率设置的前后分别调用，可以分别在这 2 个函数中实现特殊操作，例如：切换分辨率的序列不能简单的通过填写 sensor_preview_data 和 sensor_fullres_lowfps_data 来实现的话，那么就可以在这个回调中实现自己的切换操作；mt9p111.c 文件中就有类似实现；

3）、

```
00897:
00898: static int sensor_softrest_usr_cb(struct i2c_client *client,struct rk_sensor_reg *series)
00899: {
00900:
00901:     return 0;
00902: }
00903: static int sensor_check_id_usr_cb(struct i2c_client *client,struct rk_sensor_reg *series)
00904: {
```

以上 2 个函数分别是实现 Sensor 的软复位和校验 ID，可以直接填写：

```
static struct rk_sensor_reg sensor_softreset_data[]={
SensorEnd
};

static struct rk_sensor_reg sensor_check_id_data[]={
SensorEnd
};
```

来实现功能，但是如果不能简单通过填写序列实现的话，就将相应功能实现在这两个回调函数中；

4）、

```
01019: /*
01020: * the functions are focus callbacks
01021: */
01022: static int sensor_focus_init_usr_cb(struct i2c_client *client){
01023:     return 0;
01024: }
01025:
01026: static int sensor_focus_af_single_usr_cb(struct i2c_client *client){
01027:     return 0;
01028: }
01029:
01030: static int sensor_focus_af_near_usr_cb(struct i2c_client *client){
01031:     return 0;
01032: }
01033:
01034: static int sensor_focus_af_far_usr_cb(struct i2c_client *client){
01035:     return 0;
01036: }
01037:
01038: static int sensor_focus_af_specialpos_usr_cb(struct i2c_client *client,int pos){
01039:     return 0;
01040: }
01041:
01042: static int sensor_focus_af_const_usr_cb(struct i2c_client *client){
01043:     return 0;
01044: }
01045: static int sensor_focus_af_close_usr_cb(struct i2c_client *client){
01046:     return 0;
01047: }
01048:
01049: static int sensor_focus_af_zoneupdate_usr_cb(struct i2c_client *client){
01050:     return 0;
01051: }
```

以上回调函数都是针对自动对焦的功能实现，依次是：

(1)、初始化回调，可以实现下载固件等操作，由于下载固件比较耗时，麻烦将下载固件在填写序列时标注 SensorStreamChk 属性，参考 ov5640_af_firmware.c;

(2)、单次对焦触发回调；

(3)、微距对焦回调;

(4)、无限远对焦回调；

(5)、特定位置对焦回调，即在微距与无限远之间；

(6)、连续对焦回调；

(7)、暂停对焦回调；

(8)、改变对焦区域回调，实现 Touch focus；

以上回调的函数可以参考 ov5640.c、mt9p111.c；

5）、

```
01060: /*
01061: *   The function can been run in sensor_init_parametres which run in sensor_probe, so user can do some
01062: * initialization in the function.
01063: */
01064: static void sensor_init_parameters_user(struct specific_sensor* spsensor,struct soc_camera_device *icd)
01065: {
01066:     return;
01067: }
01068:
```

该回调函数将在驱动注册的最后被调用，驱动可以在这边实现一些必要的初始化；

编写 Sensor 驱动文件外，其它的必要定义

编写一个新的 Sensor 驱动，以上所述的实现以及配置都是在集中在 Sensor 驱动文件本身，还需要在其它文件中增加一些必要的定义：

1）、arch/arm/plat-rk/include/plat/rk_camera.h:

RK29_CAM_SENSOR_XXXX	Sensor 名字，用于设备注册以及驱动标识
RK29_CAM_SENSOR_NAME_XXX	Sensor 名字对应的字符串
XXXX_FULL_RESOLUTION	Sensor 对应的全分辨率
XXXX_I2C_ADDR	Sensor 对应的 I2C 设备地址
XXXX_PWRDN_ACTIVE	Sensor powerdown 引脚的有效电平

注： bit4： 1 该 Sensor 支持直接从 standby 模式恢复，无需重新初始化

0 该 Sensor 不支持直接从 standby 模式恢复，需重新初始化

XXX_PWRSEQ Sensor 的上电时序，默认采用 sensor_PWRSEQ_DEFAULT

注：上电时序的定义，以 4 位 2 进制位标识上电类别，上电时序顺序依次以 bit0－bit3， bit4－bit7……. 从低到高；

2）、include /media/v4l2-chip-ident.h

定义一个 Sensor 对应的 v4l2 识别号，例如：

V4L2_IDENT_HM2057 = 64150,

5.2 如何注册一个 Camera Sensor 设备

注册一个 Camera 设备，只需要在 board 文件 static struct rkcamera_platform_data new_camera[] =中定义一个 Camera 设备即可。

注意：

原来 board 文件中，前后置摄像头分别可以定义 3 个，这个限制在目前版本中不作限制；需要硬件兼容多个摄像头，只要定义多个即可；

兼容多个摄像头的唯一条件：

Sensor 驱动中必须实现硬件识别号识别功能，即填写 static struct rk_sensor_reg sensor_check_id_data[]序列或是实现回调:sensor_check_id_usr_cb;

```
00897:
00898: static int sensor_softrest_usr_cb(struct i2c_client *client,struct rk_sensor_reg *series)
00899: {
00900:
00901:     return 0;
00902: }
00903: static int sensor_check_id_usr_cb(struct i2c_client *client,struct rk_sensor_reg *series)
00904: {
```

定义一个 Camera 设备可以采用以下格式宏：

1）、简单注册：

new_camera_device(sensor_name,

face,
pwdn_io,
flash_attach,
mir,
i2c_chl,
cif_chl)

sensor_name:

Sensor 设备名字，采用 arch/arm/plat-rk/include/plat/rk_camera.h 中 RK29_CAM_SENSOR_XXXX，与原来 board 文件中的 CONFIG_SENSOR_X 配置一致；

Face:

Sensor 设备作为前置还是作为后置的配置；前置直接填写 front，后置直接填写 back；

Pwdn_io:

Sensor 设备 powerdown（standby）连接 GPIO 的配置，与原 board 文件中 CONFIG_SENSOR_POWERDN_PIN_XX 配置一致；

flash_attach：

该 Sensor 设备是否连接 flash 闪光灯的配置；

Mir:

该 Sensor 设备的镜像配置；其中：

bit0: 0: mirror off
1: mirror on
bit1: 0: flip off
1: flip on

i2c_chl：

该 Sensor 设备连接的 i2c 通道号配置；

cif_chl：

该 Sensor 设备连接的 cif 控制器通道号 ，目前除了 rk3066 具有 2 个 cif 通道外，其余主控芯片都只有 1 个 cif 通道；

2)、完整注册

```
new_camera_device_ex(sensor_name,\
                     face,\
                     ori,\
                     pwr_io,\
                     pwr_active,\
                     rst_io,\
                     rst_active,\
```

```
pwdn_io,\
pwdn_active,\
flash_attach,\
res,\
mir,\
i2c_chl,\
i2c_spd,\
i2c_addr,\
cif_chl,\
mclk)\
```

简单注册只是在完整注册的某些项上采用默认值，如果不采用默认值，可以直接用完整注册的方式来定义一个设备，增加的注册项如下：

Ori：

定义 Sensor 设备的角度，与原 board 文件中 CONFIG_SENSOR_ORIENTATION_X 配置一致，在 new_camera_device 中注册，该值默认采用后置 90，前置 270；

pwr_io：

定义 Sensor 设备的电源控制引脚，与原 board 文件中 CONFIG_SENSOR_POWER_PIN_XX 配置一致；

pwr_active：

定义 Sensor 设备电源控制引脚的有效电平，与原 board 文件中 CONFIG_SENSOR_POWERACTIVE_LEVEL_X 配置一致；

rst_io：

定义 Sensor 设备的硬件复位控制引脚，与原 board 文件中 CONFIG_SENSOR_RESET_PIN_XX 配置一致；

rst_active：

定义 Sensor 设备的硬件复位有效电平，与原 board 文件中 CONFIG_SENSOR_RESETACTIVE_LEVEL_X 配置一致；

Res：

定义 Sensor 的全分辨，在 new_camera_device 中注册，该值默认采用该 sensor 在 rk_camera.h 中定义的真实全分辨率，如果想进行插值，即设备本身真实全分辨为 2Mega，可以在这一项中直接填写 3Mega 或是 5Mega 作为插值后的分辨率；

i2c_spd：

定义 Sensor 设备的 i2c 传输速度，在 new_camera_device 中注册，该值默认采用 100KHz；

mclk：

定义 Sensor 设备的输入时钟，在 new_camera_device 中注册，该值默认采用 24MHz，这边可以定义为 24 或是 48；

3）可扩展注册

除以上介绍的2种注册方式，在rk_camera v0.1.3版本以后增加支持可扩展的注册方式，

new_camera_device_raw

new_camera_device_raw_ex

以上2种注册方式分别对应前面介绍的new_camera_device 和 new_camera_device_ex，区别在于：

1、宏定义的先后必须用｛｝，隔开；

2、在｛｝内部，除了可定义 new_camera_device_raw 以及 new_camera_device_raw_ex 外，还可以增加一些后续版本增加的注册初始化项，例如 FOV 设置；

```
00061:     {
00062:     new_camera_device_raw(RK29_CAM_SENSOR_GC2035,
00063:                           back,
00064:                           RK30_PIN3_PB5,
00065:                           0,
00066:                           0,
00067:                           3,
00068:                           0)
00069:     new_camera_device_fov(51,86)
00070:     },
```

可扩展定义：

new_camera_device_fov(fov_h,fov_v)

Fov_h:

水平方向视场角；

Fov_v:

垂直方向视场角；

针对新旧版本 sensor 驱动与新旧设备注册方式的兼容方式说明如下：

1）、sensor 旧版驱动	+ board 文件旧的设备注册方式	————	yes
2）、sensor 旧版驱动	+ board 文件新的设备注册方式	————	yes
3）、sensor 新版驱动	+ board 文件旧的设备注册方式	————	no
4）、sensor 新版驱动	+ baord 文件新的设备注册方式	————	yes

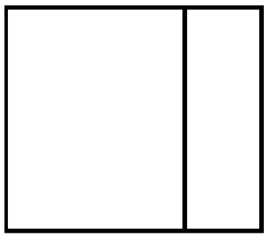
原 board 文件中，尽管采用旧的设备注册方式，也必须定义如下结构体：

```
static struct rkcamera_platform_data new_camera[] = {
    new_camera_device_end
};
```

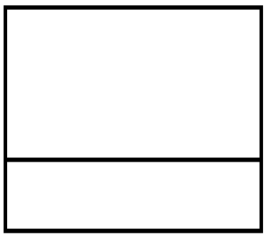
6 sensor 调试注意点：

1、 确认 sensor 的各电源引脚是否供电正常？其中包括 sensor 的 IO 电源与 VIP 控制器的 IO 电源是否相符？
2、 确认 sensor 复位电平是否正常？
3、 确认 sensor power down 控制 IO 是否控制正常，有些板子 sensor ldo 的控制脚也单独引出，该脚也需要确认是否控制正常？
4、 确认 VIP 输出的 mclk 是否满足 sensor 要求？
5、 最后确认 I2C 是否正常？ 如果在 I2C 输出波形正常的情况下，sensor 在 i2c 的第 9 位未产生 ACK 信号，那么在确认以上 4 点后，需要考虑其他器件的影响，在 25 内核版本的 tca6424 驱动中，为了满足其清中断要求，会在 i2c 操作之后紧跟着操作 i2c 的 sck 和 sda 线。这样操作会影响到某些 I2C 器件的读写！！！
6、 图像异常时需要判断如下情况：
   1）、HREF/VSYNC 的有效电平是否与 VIP 设置一致；
   2）、PCLK 输出有效数据的边沿是否与 VIP 设置一致；
   3）、PCLK 最高只能 100MHz；
   4）、数据输出格式是否满足要求：YUV422， 输出顺序： UYVY、YUYV
   5）、sensor IO 的电平是否与 VIP IO 的电平不一致（1.8v/2.8v）；
7、Dual－Sensor 共用 VIP 控制器的 DATA/VSYNC/HREF/PCLK 信号线时，确认各个 Sensor 的 Stanby 动作是否符合要求，同一时刻只能有一个 sensor 处于工作状态。同时由于共用 IO 的负载增大，所以必须关注各个 sensor 的 IO 驱动能力是否需要调整；

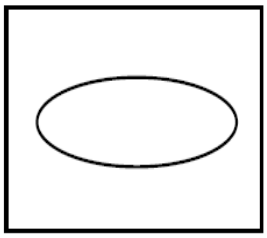
- 图像窗口位置



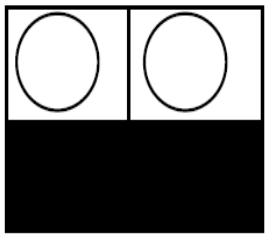
行同步问题



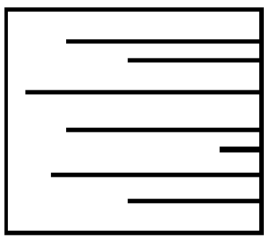
场同步问题



后端采样频率太高

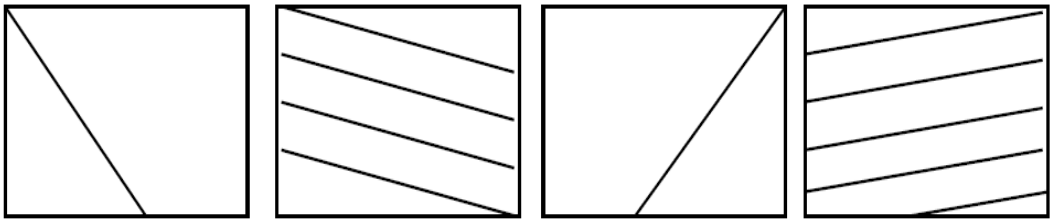


后端采样频率太低

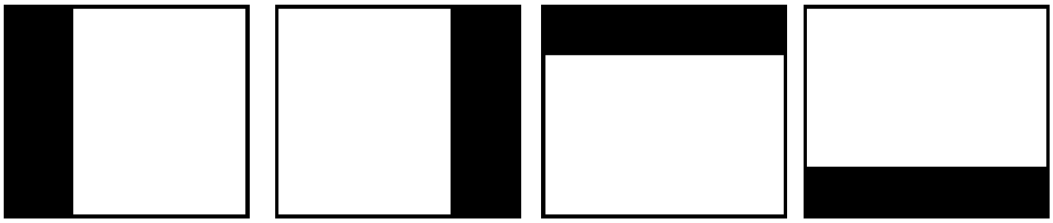


时钟与数据同步有问题

- 图像窗口位置



行数据多　行数据多　行数据少　行数据少



窗口位置不对　窗口位置不对　输出行数不够　输出行数不够

7 Camera Sensor 支持列表

Camera Sensor 型号	Driver 版本	备注
5Mega		
Ov5640	V0.1.0 cts pass	AF、Tocush focus
Mt9p111	V0.1.0 cts pass	AF、Tocush focus
hm5065	V0.1.0 cts pass	AF、Tocush focus
Ov5647	V0.1.0 cts pass	RK SoftISP support
3Mega		
nt99340	V0.1.0 cts pass	AF
2Mega		
ov2659	V0.1.0 cts pass	
Nt99240	V0.1.0 cts pass	
nt99252	V0.1.0 cts pass	
sp2518	V0.1.0 cts pass	
hm2057	V0.1.0 cts pass	
gc2015	V0.0.1	
gc2035	V0.1.0 cts pass	
gc2235	V0.1.0 cts pass	RK SoftISP support
1Mega		
nt99160	V0.1.0 cts pass	
0.3Mega		
gc0307	V0.0.1	
gc0308	V0.0.1	
gc0309	V0.0.1	
gc0329	V0.0.1	
gc0328	V0.0.1	
以下 Sensor 驱动为旧版 （除标注外，其余 Sensor 都未在 sdk 验证）		
5Mega		
ov5642		SDK 可验证
3Mega		
ov3640		
ov3660		
mt9t111		
S5k5ca		
2Mega		
Ov2640		
gt2005		
hi253		
nt99250		
ov2655		
sid130B		
1Mega		
mt9d112		
mt9d113		
0.3Mega		
ov7675		
s5k6aa		

siv120b		
hi704		
Sp0838		
Sp0215		

以上列表所列出的 Sensor 为 sdk 中已经提供的 Sensor 驱动，如果客户需要选择其他模组的也可以，但是必须注意采用的 Sensor 模组必须符合以下条件：

1）、soc camera sensor + DVP Interface + YUV422

2）、RAW camera sensor 在软件 isp 中只支持 ov5647 以及 gc2235，其余需要外扩 ISP；

3）、Sensor 输出电平必须是 1.8v、3.3v 之一；

4）、Sensor AF 功能必须模组集成马达，集模组内置支持，RK 芯片未实现 ISP 功能无法通过控制马达实现 AF 的功能；

8. USB 摄像头支持说明

Usb 摄像头采用 linux 的 uvc 标准驱动，linux 编译时必须打开 uvc 驱动相关选项：

```
Device Drivers  --->
  <*> Multimedia support  --->
    [*]  Video capture adapters  --->
      [*]  V4L USB devices  --->
        <*>  USB Video Class (UVC)
```

目前 android 支持自动识别 uvc 驱动以及 rk29camera 驱动，针对 usb 摄像头目前只支持 yuv422（yuyv）数据，由于 usb 摄像头在高分辨率情况下采用的是 mjpeg 数据格式，目前暂不支持；

以下为 uvc 驱动支持摄像头列表，选用时最好采用下表中，并且支持输出 yuv422（yuyv）数据的摄像头：（注释：以下列表中的 uvc sensor 不一定支持，sdk 板调试确认支持的只用罗技 C110）

9 android camera 模块配置注意点

9.1 DV 分辨率设置（media_profiles.xml）

针对 CameraHal 版本在 v0.4.1 以前的版本，麻烦手动修改 media_profiles.xml 文件，修改规则详见本文档的 v0.3.2 及其以前版本的第 8.1 章节；

CameraHal v0.4.1 + Camera driver v0.3.1 版本在配置 media_profiles.xml 的方式上，有如下 2 种方式：

1). camerahal_module 启动时根据 /etc/media_profiles_default.xml 自动生成 /data/media_profiles.xml；

2). 手动修改 media_profiles.xml, 文件存储到如下路径/etc/media_profile.xml；
存储在/etc/media_profiles.xml 在兼容多个 Sensor 时的命名规则详见本文档的 v0.3.2 及其以前版本的第 8.1 章节，其中需要说明的一点：
在新版 board 文件的使用 new_camera_device 或是 new_camera_device_ex 注册的设备，由于为了兼容旧版 board 文件中的定义，用该方式注册的设备序号前置后置分别从 3 开始递增；
如果出现切换 dv 无法成功，或是录制 dv 时无效，怀疑是该配置文件出问题，麻烦按照以下方式进行纠错：

1）**查询以下文件是否存在，或者必须删除的文件**
(1)、/etc/media_profiles.xml （必须删除）
如果此文件存在，则默认使用该文件作为 DV 分辨率设置。
自动生成 DV 配置文件就必须将此文件删除，否则查询到有此文件存在就不会自动 生成 DV 配置文件。
(2)、/etc/media_profiles_default.xml（必须存在）
自动生成 DV 配置文件是拷贝该文件后再进行修改的，所以改文件必须存在。
(3)、/data/media_profiles.xml（自动生成的文件）
Camerahal_module 自动生成的 DV 配置文件，机器启动完毕后请确认有生成此文件。
自动生成文件只会在查询到 sensor 改变后，才会再自动生成。

2). **查询各软件版本号，确定有自动生成配置文件的功能**
getprop
[sys_graphic.cam_driver.ver]: [0.3.1]
[sys_graphic.cam_hal.ver]: [0.4.1]

3). 查看 DV 配置文件过程后台提示信息
请在机器启动完毕后，使用 logcat 命令查看 andorid 后台信息，以确保生成正确的 xml 文件。请关注以下提示信息：

(1). 是否有客户自定义的/etc/media_profiles.xml 存在，如果存在会出现下面信息。

D/CameraHal_Module: client have /etc/media_profiles.xml file, so we use client file first!

如果出现这条 log，此时请删除红色字的文件,再重新运行即可再次自动生成 xml。

(2). 如果没有客户自定义文件，则机器开始生成文件提示

2.1 如果/data/media_profiles 还未生成则提示如下：

D/CameraHal_Module(101): create file /data/media_profiles.xml from /etc/media_profiles_default.xml, and alter its configuration

2.2 如果/data/media_profiles 已经生成，但是 sensor 改变，则提示如下

D/CameraHal_Module(101): /data/media_profiles.xml is exist, but camera device is not same!

D/CameraHal_Module(101): create file /data/media_profiles.xml from/etc/media_profiles_default.xml, and alter its configuration

2.3:如果/data/media_profiles 已经生成，但是 sensor 不变，则提示如下

D/CameraHal_Module(101): /data/media_profiles.xml is exist, and camera device is same!

D/CameraHal_Module(101): create file /data/media_profiles.xml from/etc/media_profiles_default.xml, and alter its configuration

(3). /dev/video0 驱动版本号及查询帧率提示

D/CameraHal_Module(101): camera_request_framerate.632 Current camera driver version: 0.3.1

D/CameraHal_Module(101): CameraId:0 176x144(800x600) fps: 15

D/CameraHal_Module(101): CameraId:0 240x160(800x600) fps: 15

D/CameraHal_Module(101): CameraId:0 320x240(800x600) fps: 15

D/CameraHal_Module(101): CameraId:0 352x288(800x600) fps: 15

D/CameraHal_Module(101): CameraId:0 640x480(800x600) fps: 15

D/CameraHal_Module(101): CameraId:0 720x480(800x600) fps: 15

D/CameraHal_Module(101): CameraId:0 800x600(800x600) fps: 15

D/CameraHal_Module(101): CameraId:0 1280x720(1280x720) fps: 5

(4). /dev/video1 驱动版本号及查询帧率提示

D/CameraHal_Module(101): camera_request_framerate.632 Current camera driver version: 0.3.1

D/CameraHal_Module(101): CameraId:1 176x144(800x600) fps: 15

D/CameraHal_Module(101): CameraId:1 240x160(800x600) fps: 15

D/CameraHal_Module(101): CameraId:1 320x240(800x600) fps: 15

D/CameraHal_Module(101): CameraId:1 352x288(800x600) fps: 15

D/CameraHal_Module(101): CameraId:1 640x480(800x600) fps: 15

D/CameraHal_Module(101): CameraId:1 720x480(800x600) fps: 15

D/CameraHal_Module(101): CameraId:1 800x600(800x600) fps: 15

D/CameraHal_Module(101): CameraId:1 1280x720(1280x720) fps: 15

D/CameraHal_Module(101): CameraId:1 1920x1080(2048x1536) fps: 5

(5). 修改 xml 提示（以下信息省略 D/CameraHal_Module(101)的开头）

XML modify: camID(0) resolution:qcif(176x144) fps(15) isaddmark(0)

XML modify: camID(0) resolution:qvga(320x240) fps(15) isaddmark(0)
XML modify: camID(0) resolution:cif(352x288) fps(15) isaddmark(0)
XML modify: camID(0) resolution:480p(640x480) fps(15) isaddmark(1)
XML modify: camID(0) resolution:480p(720x480) fps(15) isaddmark(0)
XML modify: camID(0) resolution:720p(1280x720) fps(5) isaddmark(0)
XML modify: camID(0) resolution:qcif(176x144) fps(15) isaddmark(0)
XML modify: camID(0) resolution:qvga(320x240) fps(15) isaddmark(0)
XML modify: camID(0) resolution:cif(352x288) fps(15) isaddmark(0)
XML modify: camID(0) resolution:480p(640x480) fps(15) isaddmark(1)
XML modify: camID(0) resolution:480p(720x480) fps(15) isaddmark(0)
XML modify: camID(0) resolution:720p(1280x720) fps(5) isaddmark(0)

XML modify: camID(1) resolution:qcif(176x144) fps(15) isaddmark(0)
XML modify: camID(1) resolution:qvga(320x240) fps(15) isaddmark(0)
XML modify: camID(1) resolution:cif(352x288) fps(15) isaddmark(0)
XML modify: camID(1) resolution:480p(640x480) fps(15) isaddmark(1)
XML modify: camID(1) resolution:480p(720x480) fps(15) isaddmark(0)
XML modify: camID(1) resolution:720p(1280x720) fps(15) isaddmark(0)
XML modify: camID(1) resolution:qcif(176x144) fps(15) isaddmark(0)
XML modify: camID(1) resolution:qvga(320x240) fps(15) isaddmark(0)
XML modify: camID(1) resolution:cif(352x288) fps(15) isaddmark(0)
XML modify: camID(1) resolution:480p(640x480) fps(15) isaddmark(1)
XML modify: camID(1) resolution:480p(720x480) fps(15) isaddmark(0)
XML modify: camID(1) resolution:720p(1280x720) fps(15) isaddmark(0)
meida_profiles_xml_control time (500752)us

CamID(0)代表/dev/video0
CamID(1)代表/dev/video1
resolution:480p(640x480)代表分辨率为：640*480
Fps 代表对应分辨率的帧率
Isaddmark 代表 1：xml 文件中此项分辩率被注释掉。
0：xml 文件中有此项分辩率对应帧率。

(6). /data/media_profiles.xml 有被使用到

```
D/MediaProfiles(   99): CameraGroupFound(639): media_profiles_id: 0xffff0403
D/MediaProfiles(   99): getInstance(697): Create instance from /data/media_profiles.xml
```

9.2 4.0. Panorama(全景拍照) and FaceLock（人脸解锁）

4.0 中 Panorama(全景拍照)针对的是后置摄像头，如果机器没有后置摄像头的话，那么该功能无效；

4.0 中 FaceLock（人脸解锁）针对的是前置摄像头，如果机器没有前置摄像头的话，那么该功能无效；

摄像头的前置、后置配置详见《**4.4.2 3.0 kernel board 文件配置**》章节的说明

10 android camera 模块各项目 CTS 测试注意事项

10.1 testPreviewFpsRange 测试

10.1.1 android 2.3 版本 testPreviewFpsRange

该项测试主要检查 camera 的实际帧率是否与登记的帧率符合。各个项目中的 sensor 的实际帧率不一，所以针对该项测试，各个项目需要对登记帧率进行修改。针对 2.00sdk 以及 1.28sdk 打了 camera 相关补丁 20110826，需要修改 hardware/rk29/camera/CameraHal.cpp 中：

```
594     /*frame per second setting*/
595     parameterString = "15000,15000";
596     params.set(CameraParameters::KEY_PREVIEW_FPS_RANGE, parameterString.string());
597     parameterString = "(15000,15000)";
598     params.set(CameraParameters::KEY_SUPPORTED_PREVIEW_FPS_RANGE, parameterString.string());
```

针对 drivers/media/video/rk29_camera_oneframe.c 驱动的 0.0.3 版本：

```
00132: //Configure Macro
00133: #define RK29_CAM_VERSION_CODE KERNEL_VERSION(0, 0, 3)
```

Android 中 hardware/rk29/camera/CameraHal.h 中具备以下配置宏：

```
00051:
00052: #define CONFIG_AUTO_DETECT_FRAMERATE    0
00053:
00054: #if CONFIG_AUTO_DETECT_FRAMERATE
00055: #define CAMERA_DEFAULT_PREVIEW_FPS_MIN    8000          //8 fps
00056: #define CAMERA_DEFAULT_PREVIEW_FPS_MAX    15000
00057: #else
00058: #define CAMERA_FRONT_PREVIEW_FPS_MIN    8000          //8 fps
00059: #define CAMERA_FRONT_PREVIEW_FPS_MAX    15000
00060: #define CAMERA_BACK_PREVIEW_FPS_MIN     8000
00061: #define CAMERA_BACK_PREVIEW_FPS_MAX     15000
00062: #endif
00063:
```

该版本支持 android 在启动时自动检测 camera 的所有支持预览分辨率的帧率，即将 CONFIG_AUTO_DETECT_FRAMERATE 宏打开，但是该项功能打开后，在系统启动时会耗时接近 30s 检测 2 个摄像头的帧率，在尚未检测完毕时，进入 camera 会黑屏等待。如果关闭自动检测功能，麻烦配置上图所示的各个宏：

```
#define CAMERA_FRONT_PREVIEW_FPS_MIN      8000          //前置 sensor 帧率最小值
#define CAMERA_FRONT_PREVIEW_FPS_MAX      15000
#define CAMERA_BACK_PREVIEW_FPS_MIN       8000          //后置 sensor 帧率最小值
#define CAMERA_BACK_PREVIEW_FPS_MAX       15000
```

10.1.2 android 4.0.3 版本 testPreviewFpsRange

4.0.3 版本测试该项时，Camera 硬件抽象层从 kernel 获取相应 camera 的各个分辨率的帧率信息。Kernel 中各个项目的帧率信息设置详见**《4.4.2 3.0 kernel board 文件配置》**章节。这边需要注意：rk29_camera_oneframe.c（camera 驱动）的版本在 v0.x.5 时支持帧率测试（配

合 camera_test 工具），但是测试出来的帧率存在 2fps 的误差，所以在 testPreviewFpsRange 测试中各项目经常出现失败，针对该项测试需要更新至 v0.x.6 版本以上的 camera 驱动，然后配置 camera_test_v1.0 工具重新测试帧率，该工具测试出来的帧率信息详见**《8.1.2 android 4.0 media_profile.xm》**第 1 小节，输出的帧率信息直接填写到 kernel board（**《4.4.2 3.0 kernel board 文件配置》**章节说明）文件中的各个帧率对应宏中。

CameraHal v0.2.8

CameraHal v0.2.2 版本支持该项测试通过，但是每个预览分辨率（KEY_SUPPORTED_PREVIEW_SIZES）的帧率信息（KEY_SUPPORTED_PREVIEW_FPS_RANGE）由查询 kernel board（**《4.4.2 3.0 kernel board 文件配置》**章节说明）文件中的各个帧率对应宏来确定。但是这些宏定义只适合 sensor 在各个分辨率输出时固定帧率的情况，在某些 sensor 的配置中会将 sensor 的输出配置成动态帧率的，这样在测试该项测试时会出现概率性的测试不通过现象。CameraHal v0.2.8 版本在填写 KEY_SUPPORTED_PREVIEW_FPS_RANGE 等帧率信息时，修改成如下方式，将 kernel board 文件中定义的各个分辨率帧率的最大值和最小值作为 KEY_SUPPORTED_PREVIEW_FPS_RANGE 帧率信息的最大值和最小值。如果查询不到某项分辨率的帧率信息，采用 CameraHal.h 文件中 CONFIG_CAMERA_FRONT_PREVIEW_FPS_MIN/CONFIG_CAMERA_FRONT_PREVIEW_FPS_MAX/CONFIG_CAMERA_BACK_PREVIEW_FPS_MIN/CONFIG_CAMERA_BACK_PREVIEW_FPS_MAX 来分别定义前后置摄像头的最大最小帧率。如上所述，在 CameraHal v0.2.8 版本针对该项测试配置如下：

1、根据 CameraTest 测试帧率填写 kernel board 文件中各个帧率宏定义信息；

2、填写 CamerHal.h 文件中 CONFIG_CAMERA_FRONT_PREVIEW_FPS_MIN/CONFIG_CAMERA_FRONT_PREVIEW_FPS_MAX/CONFIG_CAMERA_BACK_PREVIEW_FPS_MIN/CONFIG_CAMERA_BACK_PREVIEW_FPS_MAX 这 4 个宏定义信息；

3、屏蔽 kernel board 文件中 CONFIG_SENSOR_QCIF_FPS_FIXED_XX 的宏定义，这样 CameraHal .h 定义的宏信息就生效；

CameraHal v0.2.a

1、根据 CameraTest 测试帧率填写 kernel board 文件中各个帧率宏定义信息；

2、填写 CamerHal.h 文件中 CONFIG_CAMERA_FRONT_PREVIEW_FPS_MIN/CONFIG_CAMERA_FRONT_PREVIEW_FPS_MAX/CONFIG_CAMERA_BACK_PREVIEW_FPS_MIN/CONFIG_CAMERA_BACK_PREVIEW_FPS_MAX 这 4 个宏定义信息；

3、在 0.2.a 版本中，第 2 点中的宏定义立即生效，不需要屏蔽 kernel board 文件中的宏定义；

10.2 android.hardware.cts.CameraGLTest 测试

该项测试只有 4.0.3 版本 android 有要求，2.3android 没有该类要求。需要注意事项如下：

1）、testCameraToSurfaceTextureMetadata

该项主要测试在打开 camera 之后，Surface 合成的速率是否能够符合 camera 的帧率要求，由于合成速率的问题，CameraHal 的预览缓冲需要开到 4 个（版本更新至 v0.2.1）。由于 cts 测试程序的要求，针对各个预览分辨率都必须能够支持 KEY_SUPPORTED_PREVIEW_FPS_RANGE，目前我们公司的 sensor 驱动针对各个分辨率的帧率可能没有不统一，这样就会导致该测试项测试不过。简单要求如下：

（1）、各个分辨率的帧率一样；

（2）、如果各分辨率的帧率不一样，那么必须与最低帧率成倍数关系，例如：5fps、10fps、15fps……；

CameraHal v0.2.8 版本的配置与 testPreviewFpsRange 测试配置说明一致；

9.1、9.2 两个章节所提到的测试，如果出现测试不过，并且是提示 720p 分辨率时出错，麻烦各项目检查各自 kernel 的 sensor 驱动文件，如果 sensor 没有直接提供 720p 序列，注意 sensor_try_fmt 函数的实现是否有以下代码，如果没有，请参考 ov2659.c 的代码实现：

```
02021: static int sensor_try_fmt(struct v4l2_subdev *sd, struct v4l2_mbus_framefmt *mf)
02022: {
02023:     struct i2c_client *client = v4l2_get_subdevdata(sd);
02024:     struct sensor *sensor = to_sensor(client);
02025:     const struct sensor_datafmt *fmt;
02026:     int ret = 0,set_w,set_h;
02027:
02028:     fmt = sensor_find_datafmt(mf->code, sensor_colour_fmts,
02029:                 ARRAY_SIZE(sensor_colour_fmts));
02030:     if (fmt == NULL) {
02031:         fmt = &sensor->info_priv.fmt;
02032:         mf->code = fmt->code;
02033:     }
02034:
02035:     if (mf->height > SENSOR_MAX_HEIGHT)
02036:         mf->height = SENSOR_MAX_HEIGHT;
02037:     else if (mf->height < SENSOR_MIN_HEIGHT)
02038:         mf->height = SENSOR_MIN_HEIGHT;
02039:
02040:     if (mf->width > SENSOR_MAX_WIDTH)
02041:         mf->width = SENSOR_MAX_WIDTH;
02042:     else if (mf->width < SENSOR_MIN_WIDTH)
02043:         mf->width = SENSOR_MIN_WIDTH;
02044:
02045:     set_w = mf->width;
02046:     set_h = mf->height;
02047:
02048:     if (((set_w <= 176) && (set_h <= 144)) && sensor_qcif[0].reg)
02049:     {
02050:         set_w = 176;
02051:         set_h = 144;
02052:     }
02053:     else if (((set_w <= 320) && (set_h <= 240)) && sensor_qvga[0].reg)
02054:     {
02055:         set_w = 320;
```

```
02056:          set_h = 240;
02057:      }
02058:      else if (((set_w <= 352) && (set_h<= 288)) && sensor_cif[0].reg)
02059:      {
02060:          set_w = 352;
02061:          set_h = 288;
02062:      }
02063:      else if (((set_w <= 640) && (set_h <= 480)) && sensor_vga[0].reg)
02064:      {
02065:          set_w = 640;
02066:          set_h = 480;
02067:      }
02068:      else if (((set_w <= 800) && (set_h <= 600)) && sensor_svga[0].reg)
02069:      {
02070:          set_w = 800;
02071:          set_h = 600;
02072:      }
02073:      else if (((set_w <= 1280) && (set_h <= 720)) && sensor_720p[0].reg)
02074:      {
02075:          set_w = 1280;
02076:          set_h = 720;
02077:      }
02078:      else if (((set_w <= 1024) && (set_h <= 768)) && sensor_xga[0].reg)
02079:      {
02080:          set_w = 1024;
02081:          set_h = 768;
02082:      }
02083:      else if (((set_w <= 1280) && (set_h <= 1024)) && sensor_sxga[0].reg)
02084:      {
02085:          set_w = 1280;
02086:          set_h = 1024;
02087:      }
02088:      else if (((set_w <= 1600) && (set_h <= 1200)) && sensor_uxga[0].reg)
02089:      {
02090:          set_w = 1600;
02091:          set_h = 1200;
02092:      }
02093:      else
02094:      {               /* ddl@rock-chips.com : Sensor output smallest size if  isn't support app  */
02095:          set_w = SENSOR_INIT_WIDTH;
02096:          set_h = SENSOR_INIT_HEIGHT;
02097:      }
02098:
02099:      mf->width = set_w;
02100:      mf->height = set_h;
02101:      mf->colorspace = fmt->colorspace;
02102:
02103:      return ret;
02104: }
```

2）、testSetPreviewTexturePreviewCallback

该项测试在 android 4.0 CameraHal v0.3.5 之前版本可能出现概率性不过，导致后续 camera 测试都失败的现象，麻烦更新至 v0.3.5.

10.3 android.hardware.cts. SystemFeaturesTest 测试

1）、testCameraFeatures

该项测试会检查 CameraHal 中支持的后置 sensor、后置 sensor auto focus、后置 sensor flash、前置 sensor 4 项配置与/etc/ permissions 目录中各 xml 文件定义的 feaure 是否匹配。如果不匹配，麻烦各项目人员修改 device\rockchip\rk29sdk 目录下 device.mk 文件中对相关 xml 文件的拷贝定义；

android.hardware.camera.flash-autofocus.xml

后置 sensor、后置 sensor auto focus、后置 sensor flash

android.hardware.camera.autofocus

后置 sensor、后置 sensor auto focus

android.hardware.camera.front

前置 sensor

android.hardware.camera

后置 sensor

注意 handheld_core_hardware.xml 文件中有
<feature name="android.hardware.camera" />
表示的是后置摄像头，与 android.hardware.camera 文件一致

10.4 android.hardware.cts. CamcorderProfileTest 测试

1）、testGet

该项测试针对 media_profiles.xml 文件定义的合法性进行检查。麻烦详见《8.1.2 android 4.0 media_profile.xml》章节的“sensor 帧率测试小节。

如果项目中只有一个 sensor，且该 sensor 在 kernel 在 board 文件中配置成前置 sensor，那么必须将 CameraHal 模块的版本更新至 v0.2.3 及其以上版本。

单个前置摄像头 CTS 测试注意事项

关于机器只有前置摄像头，没有后置摄像头的机器过 9.4 以及 9.3 这 2 项测试时注意点如下：

针对 android-cts-4.0.3_r2 以及 android-cts-4.0.3_r1：

1）、在 testGet 测试的是后置摄像头，所以在打上 Camera_Patch_v1.1(即 CameraHal 版本 v0.2.7 及其以上版本)，这一项才可以通过；

2）、打上 Camera_Patch_v1.1(即 CameraHal 版本 v0.2.7 及其以上版本)之后，机器其实在软件上认为是 2 个摄像头，但是打开后置时打开的其实是前置。所以 9.3 项中提到的 feaure 就必须包含后置摄像头才可以通过 9.3 项。Media_profiles.xml 文件中也必须包含后置摄像头的 dv 信息，前后置信息一致；

3）、v0.2.7 以及以后版本是否打开该项功能由 CameraHal.h 中的 CONFIG_CAMERA_SINGLE_SENSOR_FORCE_BACK_FOR_CTS 该宏来控制；

注意这样打开之后，camera 应用中就会出现 2 个摄像头，这个无法避免；

针对 android-cts-4.0.3_r3：

这个版本的 CTS，不需要按照 r2 以及 r1 步骤来操作，只需按照 9.4 以及 9.3 步骤操作即可。

1 、 CameraHal 版 本 v0.2.7 及 其 以 上 版 本 必 须 保 证 CONFIG_CAMERA_SINGLE_SENSOR_FORCE_BACK_FOR_CTS 宏定义为 0；

2、 android /etc/ permissions 目录中各 xml 文件定义的 feaure，只能包含前置摄像头，不能包括后置摄像头；

10.5 CTS Verifier FOV 测试

FOV 测试支持的版本信息如下：

CameraHal Ver： v0.4.0x15

Camera Driver Ver:

rk30_camera_oneframe(CIF driver): v0.3.f

generic_sensor(Camera Sensor driver) : v0.1.b

rk_camera(Camera io): v0.1.3

针对 FOV 测试注意事项如下：

1)、进入 CTS Verifyer FOV 测试，选择 SVGA 800X600 这个分辨率拍照测试，检查 FOV 为多少，然后将该 FOV 角度在 kernel 的设备注册时填入，参见本文档的 “**如何注册一个设备**”中的“**可扩展注册**”；

2)、由于 CIF 控制器的问题，某些 sensor 驱动必须通过少采行与列（即 crop）才能正常采集。这方面的修改目前都必须按照如下修改：

旧版本 Sensor 驱动：

Sensor_s_fmt 函数中，针对分辨率：

```
2216:     else if (((set_w <= 640) && (set_h <= 480)) && (sensor_vga[0].reg!=0xff))
2217:     {
2218:         winseqe_set_addr = sensor_vga;
2219:         set_w = 640;
2220:         set_h = 480;
2221:     }
2222:     else if (((set_w <= 800) && (set_h <= 600)))
2223:     {
2224:         if (sensor_svga[0].reg!=0xff) {
2225:             winseqe_set_addr = sensor_svga;
2226:             set_w = 800-16;
2227:             set_h = 600-12;
2228:         } else if (sensor_vga[0].reg!=0xff) {
2229:             winseqe_set_addr = sensor_vga;
2230:             set_w = 640-16;
2231:             set_h = 480-12;
2232:         } else if (sensor_uxga[0].reg!=0xff) {
2233:             winseqe_set_addr = sensor_uxga;
2234:             set_w = 1600-16;
2235:             set_h = 1200-12;
2236:         }
2237:     }
```

类似这种改动，必须全部删除，即 set_w 与 set_h 不要减。然后在函数结尾通过 Sensor_CropSet(mf, percent); 其中 percent 表示列要裁剪多少百分比；

新版 Sensor 驱动

在 sensor_s_fmt_cb_th 以及 sensor_s_fmt_cb_bh 有类似操作：

Mf->width -= 16;

Mf->height -= 16;

这些操作必须删除，然后在 sensor 驱动文件宏定义的地方通过以下操作实现：

```
#iundef    SENSOR_CROP_PERCENT
#define    SENSOR_CROP_PERCENT     0     // 定义列需要裁剪多少百分比；
```

4）以上修改完成后，即可以进行所有分辨率的测试。这个时候如果还是会出现某些分辨率测试无法通过，麻烦按照以下步骤确认：

(1)、测试前，输入 echo '1' > sys/module/rk30_camera_oneframe/parameters/debug, 然后测试无法通过的分辨率；

(2)、检查输出的 kernel log 中以下信息：

rk_cam_cif(2321): YUV420 NV12 CIF Host:800x600@(0,0) Sensor:800x600->800x600 User crop:(0,0,1600,1200)in(1600,1200) (zoom: 800x600@(0,0)->1280x720)

蓝色标识的是 Sensor 输出分辨率，红色标识的为 CIF 控制器采集分辨率，绿色标识为本次测试的分辨率，如果三者相同，那表示该分辨率直接采用 Sensor 输出，可以采用以下方式解决：

A、在 Sensor 驱动中（存在这种情况的肯定是旧版本驱动），找到对应的分辨率序列，直接填结束符失效；

注：

由于 FOV 测试要求的就是视角不变，所以在测试分辨率与全分辨率宽高比不一致的情况下会出现图像变形，属于正常现象。例如： 2Mega 1600x1200 Sensor 测试 176x144；

11 android camera 摄像头模组方向说明

在 android 2.3 目录下 gingerbread\hardware\rk29\camera\CameraHal.cpp 中，如下代码为设置摄像头方向信息：

```
extern "C" void HAL_getCameraInfo(int cameraId, struct CameraInfo *cameraInfo)
{
    char property[PROPERTY_VALUE_MAX];
    int hwrotation;

    memcpy(cameraInfo, &sCameraInfo[cameraId], sizeof(CameraInfo));
    if (screenIsTall == false) {
        property_get("ro.sf.hwrotation", property, "0");
        hwrotation = strtol(property,0,0);
        if ((hwrotation == 90) || (hwrotation == 270)) {
            if (cameraInfo->facing == CAMERA_FACING_BACK)
                cameraInfo->orientation = 90;
            else
                cameraInfo->orientation = 270;
        }
    } else {
        if (cameraInfo->facing == CAMERA_FACING_BACK)
            cameraInfo->orientation = 90;
        else
            cameraInfo->orientation = 270;
    }

    LOGD("HAL_getCameraInfo:%d  ro.sf.hwrotation:%d screenIsTall:%d",cameraInfo->orientation,hwrot
}
```

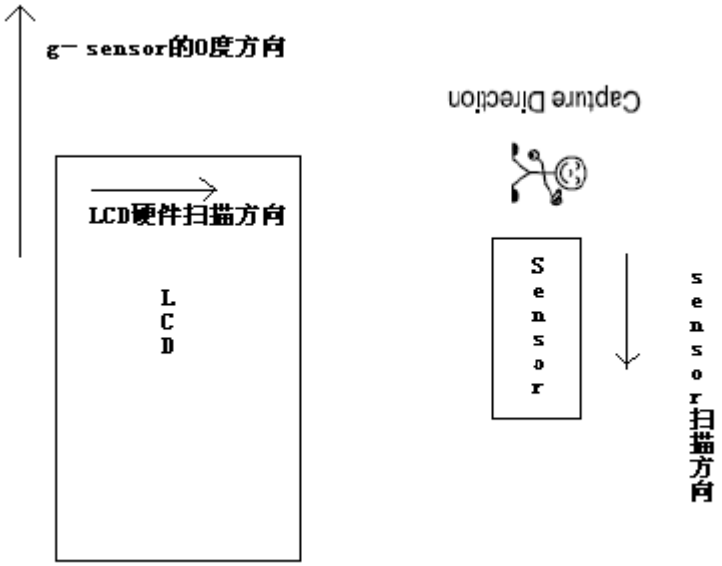
横屏、定义ro.sf.hwrotation为90、270后，前后置摄像头方向信息

竖屏，前后置摄像头方向信息

在 android 4.0 + kernel 3.0 版本中，已经将该定义移至 kernel 的 board 文件中，麻烦参考“**5.4.2 3.0 kernel board 文件配置**”章节。

在 android 中，camera 应用使用的是横屏显示，所以在硬件 PCB 设计时，摄像头模组的放置方向是和 LCD 屏的放置方向有关的。原则上必须符合如下规则：

1、摄像头模组扫描方向为横向，摄像头模组的长边必须与 LCD 屏的长边平行；如下图所示，（注释：按照模组厂一般规则，其模组规格书中，小人两手方向为 sensor 的长边方向，但是该规则不知是否所有模组厂都一致，这一点需要在硬件设计时与模组厂确认）



按上图所示，前置 sensor 在 CameraHal.cpp 中关于方向的设置应该为 270，如果朝向左边的 LCD 屏长边，那么设置成 90.

后置 sensor 的角度设置成 90 度，如果朝向左边的 LCD 屏长边，那么设置成 270.

按照上图放置 sensor 之后，最后 LCD 屏上显示图像与实际图像出现镜像效果:

1、 竖屏时，图像上下颠倒，横屏时图像左右颠倒，那么请调整 Sensor 的 Mirror 寄存器（或称为 Flip_x/Flip_h）;————左右镜像

2、 竖屏时图像左右颠倒，横屏时图像上下颠倒，那么请调整 Sensor 的 Flip 寄存器（或称为 Flip_y/Flip_v）;————————垂直镜像

3、 调整 sensor 的镜像时，修改的是 kernel 的 sensor 驱动，sensor 驱动中相关分辨率序列数组中有关寄存器内容都必须修改：

```
static struct reginfo sensor_init_data[] =
{
};

/* 720p 15fps @ 1280x720 */

static struct reginfo sensor_720p[]=
{
};

/*  1080p, 0x15fps, Oxyuv @1920x1080 */

static struct reginfo sensor_1080p[]=
{
};

/* 2592X1944 QSXGA */
static struct reginfo sensor_qsxga[] =
{
};
/* 2048*1536 QXGA */
static struct reginfo sensor_qxga[] =
{
```

4、各个 sensor 的 mirror 以及 flip 寄存器如下，仅供参考，以实际 sensor datasheet 为准

GC0308(i2c addr: 0x42):

P0:0x14	CISCTL_Mode1	8	0x00	RW	[7] hsync_always 1: hsync always on 0: hsync output at active output [6] NA [5:4] CFA sequence, determined once color filter is determined [3:2] NA [1] upside down [0] mirror

Gc0309(i2c addr: 0x42)：

P0:0x14	CISCTL_Mode1	8	0x00	RW	[7] hsync_always 1: hsync always on 0: hsync output at active output [6] NA [5:4] CFA sequence, determined once color filter is determined [3:2] NA [1] upside down [0] mirror

Gc0329(i2c addr: 0x62)：

Function	Register Address	Register Value
正常图像	0x17[1:0]	00
镜像翻转	0x17[1:0]	01
垂直翻转	0x17[1:0]	10
镜像垂直翻转	0x17[1:0]	11

Gc2015(i2c addr: 0x60)：

P0:0x29	CISCTL_mode1	8	0X20	RW	[7] HSYNC always [6] Close 2 frame dbrow [5:4] CFA sequence [3:2] dark CFA sequence [1] Updown image [0] mirror image

Gc2035(i2c addr: 0x78)：

P0:0x17	Mirror_updn	8	0X00	RW	[7:3] Reserved [1] Flip [0] mirror

Gt2005(i2c addr: 0x78)：

0x0101	VREVON	1	0x00	RW	[1] VREVON Selection verical flip 0h : Normal 1h : vertical flip
	HREVON	1			[0] HREVON Selection Horizontal mirror 0h : Normal 1h : Horizontal mirror

Hi253(i2c addr: 0x40)：

2.4. 方向调整

A:Normal　　B:X_flip　　C:Y_flip　　D:X_Y flip

```
//A: Normal
write_cmos_sensor(0x03, 0x00); //page 00
write_cmos_sensor(0x11, 0x90); //B[1:0] set "0"

//B: X_flip
write_cmos_sensor(0x03, 0x00); //page 00
write_cmos_sensor(0x11, 0x91); //B[0] set "1"

//C: Y_Flip
write_cmos_sensor(0x03, 0x00); //page 00
write_cmos_sensor(0x11, 0x92); //B[1] set "1"

//D: X_Y flip
write_cmos_sensor(0x03, 0x00); //page 00
write_cmos_sensor(0x11, 0x93); //B[1:0] set "1"
```

Hi704(i2c addr: 0x60)：

0x11 [page mode 0]: VDOCTL2 [default=0x90, r/w]

Bit	Function	Description	Default
B[7]	Windowing	User changes image size by setting WINROW[0x20,0x21:P0], WINCOL [0x22,0x23:P0], WINHGT [0x24,0x25:P0] and WINWID[0x26,0x27:P0]. (0:OFF, 1:ON)	1b
B[6:4]	Bad Frame Skip	It is used to skip bad frames when image size is changed. 001: Skip 1frame. 010: Skip 2frames, 011: Skip 3frames Note) Do not set 0.	001b
B[3]	Fixed Frame Rate2	It is used to exclude VBLANK at frame rate when frame time to be constant. Refer to 5.13[Fixed Frame Rate Timing]	0b
B[2]	Fixed Frame Rate1	Set frame time to be constant, regardless of the change of exposure time. (0:OFF, 1:ON) Refer to 5.13[Fixed Frame Rate Timing]	0b
B[1]	Y Flip	Vertical Flip Function (0:OFF, 1:ON)	0b
B[0]	X Flip	Horizontal Flip Function (0:OFF, 1:ON)	0b

Mt9d112(i2c addr: 0x7a/0x78)：

59 0x03B	1	0x0000	Vertical Flip 0 = Normal readout 1 = Readout is flipped (mirrored) vertically so that the row specified by y_addr_end_ is read out of the sensor first. Setting this bit will change the bayer pixel order (see Reg0x3024). The bit-order of bits [1:0] match the order in Reg0x301C but is reversed relative to earlier Micron Imaging sensors.

Table 35: 7: Mode Variables (continued)

Reg. #	Bits	Default	Name
	0	0x0000	Horizontal Mirror 0 = Normal readout 1 = Readout is mirrored horizontally so that the column specified by x_addr_end_ is read out of the sensor first. Setting this bit will change the bayer pixel order (see Reg0x3024).
	context B shadow register R0x20:0. Changes take effect only after REFRESH_MODE command.		

Nt99250(i2c addr: 0x6c)：

0x3022	Read_Mode_0	7:0	R/W	0x24	
	Y_Even_Inc	7:5	R/W	1	Increment applied to even addresses in Y (row) direction † 1: Normal readout
	X_Even_Inc	4:2	R/W	1	Increment applied to even addresses in X (column) direction † 1: Normal readout
	Mirror_Horizontal	1	R/W	0	Horizontal mirror 0: Normal readout 1: Mirror readout
	Flip_Vertical	0	R/W	0	Vertical flip 0: Normal readout 1: Flip readout

Ov2640(i2c addr: 0x60)：

				(8 MSBs in VSTRT[7:0] (0x19))
04	REG04	20	RW	Register 04 Bit[7]: Horizontal mirror Bit[6]: Vertical flip Bit[4]: VREF bit[0] Bit[3]: HREF bit[0] Bit[2]: Reserved Bit[1:0]: AEC[1:0] (AEC[15:10] is in register REG45[5:0] (0x45), AEC[9:2] is in register AEC[7:0] (0x10))

Ov2655(i2c addr: 0x60)：

```
i2c_salve_Address = 0x60;

MIRROR
write_i2c(0x3090, 0x08);

FLIP
write_i2c(0x307c, 0x01);flip

MIRROR&FLIP
write_i2c(0x307c, 0x01)
write_i2c(0x3090, 0x08);

NORML
write_i2c(0x307c, 0x00);no mirror/flip
write_i2c(0x3090, 0x08);
```

Ov2659(i2c addr: 0x60)：

MIRROR

20

OV2659 Camera Module Softwa

write_i2c(0x3821, 0x07)
write_i2c(0x3820, 0x81)

FLIP
write_i2c(0x3821, 0x01)
write_i2c(0x3820, 0x87)

MIRROR&FLIP
write_i2c(0x3821, 0x07)
write_i2c(0x3820, 0x87)

NORML
write_i2c(0x3821, 0x01)
write_i2c(0x3820, 0x81)

Ov3640(i2c addr: 0x78)：

```
i2c_salve_Address = 0x78;

MIRROR
write_i2c(0x307c, 0x12);mirror
write_i2c(0x3090, 0xc8);
write_i2c(0x3023, 0x0a);

FLIP
write_i2c(0x307c, 0x11);flip
write_i2c(0x3023, 0x09);
write_i2c(0x3090, 0xc0);

MIRROR&FLIP
write_i2c(0x307c, 0x13);flip/mirror
write_i2c(0x3023, 0x09);
write_i2c(0x3090, 0xc8);

NORML
write_i2c(0x307c, 0x10);no mirror/flip
write_i2c(0x3090, 0xc0);
write_i2c(0x3023, 0x0a);
```

Ov5642(i2c addr: 0x78)：

```
i2c_salve_Address = 0x78;

MIRROR
write_i2c(0x3818, 0x81);
write_i2c(0x3621, 0xe7);

FLIP
write_i2c(0x3818, 0xe1);
write_i2c(0x3621, 0xc7);

MIRROR&FLIP
write_i2c(0x3818, 0xa1);
write_i2c(0x3621, 0xe7);
```

NORML

```
write_i2c(0x3818, 0xc1);
write_i2c(0x3621, 0xc7);
```

Ov5640：（该设置似乎有问题，麻烦联系 ov FAE）

Ov7670：

OV7670/OV7171 *CMOS VGA (OmniPixel®) CAMERACHIP™ Sensor*

Table 5 Device Control Register List (Continued)

Address (Hex)	Register Name	Default (Hex)	R/W	Description
1E	MVFP	01	RW	Mirror/VFlip Enable Bit[7:6]: Reserved Bit[5]: Mirror 0: Normal image 1: Mirror image Bit[4]: VFlip enable 0: Normal image 1: Vertically flip image Bit[3]: Reserved Bit[2]: Black sun enable Bit[1:0]: Reserved

Ov7675(i2c addr: 0x78)：

figure 4-1 mirror and flip samples

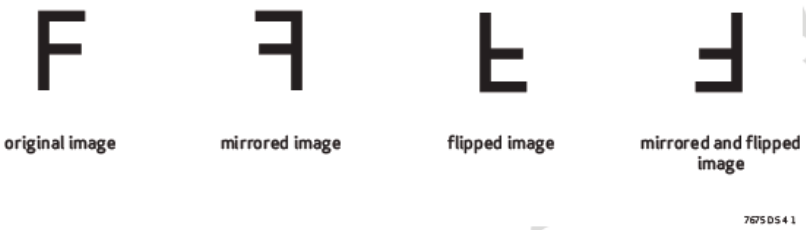


table 4-1 image windowing control functions

address	register name	default value	R/W	description
0x1E	MVFP	0x01	RW	Mirror/VFlip Enable Bit[5]: Mirror 0: Normal image 1: Mirror image Bit[4]: VFlip enable 0: Normal image 1: Vertically flip image

Sid103b(i2c addr: 0x37)：

			1 Dynamic mode – Normal operation mode	
0x04	CNTR_B	0x00	[7:6] Control fixed frame mode 00 Normal operation mode 01 Fixed frame mode @ exposure time <= frame size Normal operation mode @ exposure time > frame size 10/11 Fixed frame mode [4] Select register group between timing Group A & Group B 0 : Select Group_B Registers 1 : Select Group_A Registers [3:2] Clock (PCLK) divider – max 1/8 PCLK 00: PCLK 01: 1/2PCLK 10: 1/4PCLK 11: 1/8PCLK [1] Control vertical flip image 0 Normal image 1 Vertical flip image [0] Control horizontal flip (mirror) image 0 Normal image 1 Horizontal flip (mirror) image	R/W

12 Camera Digital Zoom

Rk2918 camera driver v0.0.4 及其以上版本开始支持数码变焦功能，

```
00136: //Configure Macro
00137: /*
00138: *             Driver Version Note
00139: *v0.0.1 : this driver first support rk2918;
00140: *v0.0.2 : fix this driver support v4l2 format is V4L2_PIX_FMT_N
00141: *          and V4L2_PIX_FMT_YUV422P;
00142: *v0.0.3 : this driver support VIDIOC_ENUM_FRAMEINTERVALS;
00143: *v0.0.4 : this driver support digital zoom;
00144: */
00145: #define RK29_CAM_VERSION_CODE KERNEL_VERSION(0, 0, 4)
00146:
```

针对数码变焦功能，驱动的实现方法是依旧 IPP 的裁剪缩放功能实现的，所以在实现上必须事先将 sensor 数据采集到内存中，然后经过 IPP 裁剪放大到用户指定的内存区域。同言之，为了支持数码变焦，必须预先分配足够的内存给 vip 采集，内存分配值见 **[rk29_camera.c (drivers\media\video)]** 文件中 PMEM_CAMIPP_NECESSARY:

```
00040: #if (PMEM_CAM_FULL_RESOLUTION == 0x500000)
00041: #define PMEM_CAM_NECESSARY    0x1200000        /* 1280*720*1.5*4(preview) + 7.5M(
00042: #define PMEM_CAMIPP_NECESSARY     0x800000
00043: #elif (PMEM_CAM_FULL_RESOLUTION == 0x300000)
00044: #define PMEM_CAM_NECESSARY    0xe00000         /* 1280*720*1.5*4(preview) + 4.5M(
00045: #define PMEM_CAMIPP_NECESSARY     0x500000
00046: #elif (PMEM_CAM_FULL_RESOLUTION == 0x200000) /* 1280*720*1.5*4(preview) + 3M(ca
00047: #define PMEM_CAM_NECESSARY    0xc00000
00048: #define PMEM_CAMIPP_NECESSARY     0x400000
00049: #elif ((PMEM_CAM_FULL_RESOLUTION == 0x100000) || (PMEM_CAM_FULL_RESOLUTION == 0
00050: #define PMEM_CAM_NECESSARY    0x800000         /* 800*600*1.5*4(preview) + 2M(cap
00051: #define PMEM_CAMIPP_NECESSARY     0x400000
00052: #elif (PMEM_CAM_FULL_RESOLUTION == 0x30000)
00053: #define PMEM_CAM_NECESSARY    0x400000         /* 640*480*1.5*4(preview) + 1M(cap
00054: #define PMEM_CAMIPP_NECESSARY     0x400000
00055: #else
00056: #define PMEM_CAM_NECESSARY    0x1200000
00057: #define PMEM_CAMIPP_NECESSARY     0x800000
00058: #endif
```

内核 menuconfig 中：

<*>RK29XX Camera Sensor Interface driver

RK29XX Camera Sensor Interface Work Mode (VIP OneFrame Mode) --->

RK29XX camera sensor interface work with IPP (VIP work with IPP) --->

RK29XX camera digital zoom with IPP (Digital zoom with IPP on) --->

红色标识的 config 配置决定了是否分配 PMEM_CAMIPP_NECESSARY 内存。

13 Camera Memory

CameraHal v0.2.4 + Camera Driver v0.1.8 版本及其以上版本支持 Camera 模块必需内存（预览内存 preview buffer、拍照内存 raw buffer、编码输出内存 JPEG buffer）从 ION 模块中动态分配得到，同时兼容原来 pmem 预留内存中分配的方式。采用何种分配方式由以下宏来决定：

Kernel ： menuconfig：

Device Drivers --->
Multimedia support --->
Video capture adapters --->
RK29XX Camera Sensor Interface driver
RK29XX Camera Sensor Interface Work Mode (VIP OneFrame Mode) --->
RK29XX camera sensor interface work with IPP (VIP work with IPP) --->
RK29XX camera digital zoom with IPP (Digital zoom with IPP on) --->
RK29XX camera memory (Camera memory from pmem) --->

Android CameraHal：

hardware\rk29\camera\CameraHal_Mem.h 文件中

```
00001: /*
00002: *Author: zyc@rock-chips.com
00003: */
00004:
00005: #define CAMERA_MEM_ION                    0
00006: #define CAMERA_MEM_PMEM                   1
00007: /*
00008: *NOTE:
00009: *    configuration macro
00010: *
00011: */
00012: #define CONFIG_CAMERA_MEM                 CAMERA_MEM_PMEM
```

hardware\rk29\camera\Android.mk(增加链接 libion 库)

```
LOCAL_SHARED_LIBRARIES:= \
        libgui\
        libjpeghwenc\
        libjpeg\
        libyuvtorgb\
        libion
```

Camera 内存从 ion 模块动态分配的话，那么 Camera 内存、UI 内存、VPU 内存都可以共享，达到节省内存的目的。但是这个必需要 ION 模块的支持，确认 ION 模块是否支持的方式：

Kernel 启动时有如下信息：

Rockchip ion module(version: 1.0) is successfully loaded

确认以上信息后，打开以上 camera 提到的宏配置，重新编译代码后，打开 camera 模块，android 的 log 中有以下信息：

Ion(version: 1.0) is successfully opened by camera

这时表示 camera 内存是从 ion 设备中分配的。如果发现以下信息：

/dev/pmem_cam isn't registered,CameraHal_Mem current configuration isn't support ION memory

那表示 CameraHal_Mem.h 中 CONFIG_CAMERA_MEM 宏配置成 pmem 模式，kernel menuconfig 中又配置成 ion 模式。

14 Camera 模块各源码版本规则说明

1、源码版本获取方式：

在 shell 命令中输入以下命令：

#getprop

[sys_graphic.cam_driver.ver]: [0.1.a]

[sys_graphic.cam_hal.ver]: [0.2.7]

sys_graphic.cam_driver.ver：

针对的是 cif 驱动的版本号，即 rk30_camera_oneframe

Kernel 中其它 camera 模块的版本直接获取对应 sys/module/xxx/parameters/version：

例如 ov2659 的版本：

Cat sys/module/ov2659/parameters/version

RK_SwIsp 驱动版本：

cat sys/module/rk_swisp_drv/parameters/version

256

数字转化成版本号：va.b.c

a= (virsion>>16)&0xff = 0

b = (virsion>>8)&0xff = 1

c = (virsion)&0xff = 0

Apk 版本获取方式：

打开 camera 应用时，android LOG 中包含版本打印：

sys_graphic.camera.apk.ver: 0.0.1

2、版本号规则说明

CamerHal v0.2.d，Kernel v0.2.b 及其以后版本， 版本号尾数偶数为正式版本（即经过品质部专项测试的版本），奇数为开发版本(只经过自测)。

3、源码版本历史记录

源码版本历史记录可以分别在 CameraHal.h 以及 rk30_camera_oneframe.c，rk29_camera_oneframe.c 中查询。

15 android4.0 预览垂直及水平镜像问题说明

由于模组关系，可能会造成同样一份 sensor 驱动代码在某些产品上看起来图像是左右镜像，或者上下镜像的。配置 board 中的 sensor 旋转角度也许可以解决 camera 应用图像镜像问题，但是其他应用出来的图像可能还是不正常的，如 POCO 相机等应用（这些应用没有用到 board 中定义的相关角度信息）。为了解决此类问题，首先要确定模组是否与驱动相匹配。即在不修改 board 中旋转角度（前置 270，后置 90）情况下，打开 camera 应用查看图像是否正常（也可通过 camera_test 测试程序查看，参照 15 章说明），如果出现镜像问题则需要在序列中加入镜像处理，即需要重新定义 sensor 相关序列，操作方法参照 前 述第 5.4.2.2 节。

16 Camera_test 测试程序使用说明

该测试程序可验证 camera 的基本功能，特别是可用来测试 sensor 各分辨率的帧率。1.1 版本兼容 29 和 30 平台。使用方法如下：

1、 将 camera_test 文件 push 到/system/bin 目录下

2、 执行 su 命令

3、 chmod 777 /system/bin/camera_test

4、 可执行 camera_test -h 获取使用帮助信息

常用参数有：

-i 测试帧率，具体使用请参照 9.1.2 节

-d 指定哪个 sensor，参数为 /dev/videoX， X 为 0 或 1

-r 指定要测试的分辨率，格式如 800x600

-z 数码变焦测试

其他还有-f,-F,-e,-L 等参数，具体使用可通过-h 查询。此外，使用 camera_test 测试程序也可判断出模组的方向，只要输入 camera_test –z –i 640x480 查看图像状态即可。

17 针对视频通话远端图像镜像问题说明

在 android 中，前置摄像头在本地预览显示时增加了水平镜像 mirror 处理，该处理是放在显示端进行，所以针对某些视频通话 apk（skype 等）在传送给远端的数据如果直接通过 camera 的 preview 接口获得的话，就会出现远端图像相对于本地预览图像颠倒的问题。该问题只在前置摄像头情况下存在。

CameraHal v0.3.17 版本及其以上版本支持针对特定视频通话 apk，在发送预览数据前先进行一次镜像操作，配置步骤如下：

```
00177:
00178: #define CONFIG_CAMERA_SINGLE_SENSOR_FORCE_BACK_FOR_CTS    0
00179: #define CONFIG_CAMERA_FRAME_DV_PROC_STAT    0
00180: #define CONFIG_CAMERA_FRONT_MIRROR_MDATACB  1
00181: #define CONFIG_CAMERA_FRONT_MIRROR_MDATACB_ALL  0
00182: #define CONFIG_CAMERA_FRONT_MIRROR_MDATACB_APK  "<com.skype.raider>,"
00183: #define CONFIG_CAMERA_PRVIEW_BUF_CNT        4
00184: #define CONFIG_CAMERA_UVC_INVAL_FRAMECNT    5
00185: #define CONFIG_CAMERA_ORIENTATION_SKYPE     0
00186: #define CONFIG_CAMERA_FRONT_ORIENTATION_SKYPE     0
00187: #define CONFIG_CAMERA_BACK_ORIENTATION_SKYPE      0
00188:
```

1、 CONFIG_CAMERA_FRONT_MIRROR_MDATACB 该宏配置是否打开该项功能；

2、 CONFIG_CAMERA_FRONT_MIRROR_MDATACB_ALL 该宏配置是否针对所有 apk 都执行镜像动作；

3、 在 CONFIG_CAMERA_FRONT_MIRROR_MDATACB_ALL 配置关闭的情况下，可以由 CONFIG_CAMERA_FRONT_MIRROR_MDATACB_APK 来配置需要进行镜像动作的 apk，将 apk 名称用<>填写入该宏即可；

4、 针对 apk 名称的获取，可以用该 apk 打开 camera，查看 camera 以下 LOG 获得：D/CameraHal(100): Calling process is: com.android.gallery3d

18 Camera 插值说 明

针对目前版本，插值只需要使用 new_camera_device_ex 进行 Camera 设备的完整注册，其中 res 项填写想要插值的目标全分辨率即可；

19 RK30_Camera_Patch_v3.1 补丁常见问题

1、版本兼容情况说明

1）、sensor 旧版驱动 + board 文件旧的设备注册方式 ————— yes

2）、sensor 旧版驱动 + board 文件新的设备注册方式 ————— yes

注意：采用这种注册方式，使用 board 文件旧的设备注册方式注册的 flash IO 会失效，建议如下：

2.1）、 采用第一种方式，即 sensor 驱动以及设备注册方式都采用旧的；

2.2）、 原 sensor 驱动中没有 flash 功能的可以忽略；

2.3）、 将 sensor 驱动移植成新版的，即第 4 种方式；flash 的实现在 board 文件的对应回调中实现；

3）、sensor 新版驱动 + board 文件旧的设备注册方式 ————— no

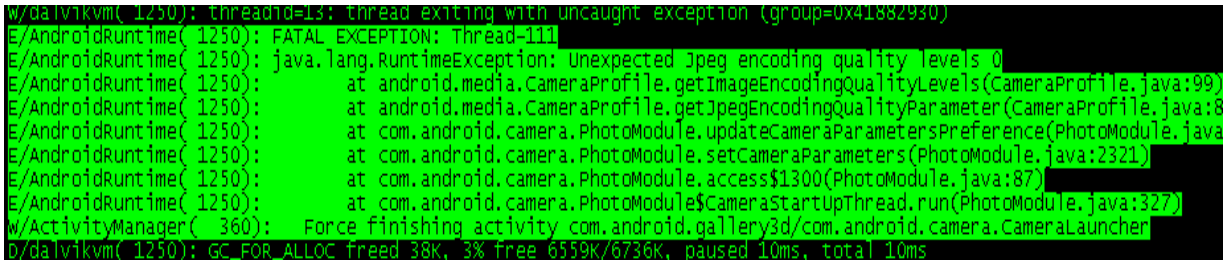
4）、sensor 新版驱动 + baord 文件新的设备注册方式 ————— yes

原 board 文件中，尽管采用旧的设备注册方式，也必须定义如下结构体：

```
static struct rkcamera_platform_data new_camera[] = {
    new_camera_device_end
};
```

2、打完补丁之后出现 camera 无法打开的问题

1）、出现如下 log：



出现这个问题就是 media_profiles.xml 出现问题；处理方式如下：

（1）、 将打补丁之前，项目工程中/etc/media_profiles.xml 推送到/etc 目录中，重启机器即可；

（2）、 按照本文档 8.1 章节关于自动生成 media_profiles.xml 的方式确认问题；如果确认后还是出现问题，麻烦发邮件到 Camera 的相关人员即可；

2)、出现如下 log：

```
D/CAM_activity( 1247): should enable panorama? false
V/CameraHolder( 1247): open camera 0
E/CameraService(  100): CameraService::connect × (pid 1247) rejected (invalid cameraId 0).
E/CameraHolder( 1247): fail to connect Camera
E/CameraHolder( 1247): java.lang.RuntimeException: Fail to connect to camera service
E/CameraHolder( 1247):  at android.hardware.Camera.native_setup(Native Method)
E/CameraHolder( 1247):  at android.hardware.Camera.<init>(Camera.java:340)
E/CameraHolder( 1247):  at android.hardware.Camera.open(Camera.java:302)
E/CameraHolder( 1247):  at com.android.camera.CameraManager.cameraOpen(CameraManager.java:283)
E/CameraHolder( 1247):  at com.android.camera.CameraHolder.open(CameraHolder.java:210)
E/CameraHolder( 1247):  at com.android.camera.Util.openCamera(Util.java:315)
E/CameraHolder( 1247):  at com.android.camera.PhotoModule$CameraStartUpThread.run(PhotoModule.java:318)
W/dalvikvm( 1247): threadid=14: thread exiting with uncaught exception (group=0x41835930)
E/AndroidRuntime( 1247): FATAL EXCEPTION: Thread-112
E/AndroidRuntime( 1247): java.lang.RuntimeException: openCamera failed
E/AndroidRuntime( 1247):        at com.android.camera.Util.openCamera(Util.java:320)
E/AndroidRuntime( 1247):        at com.android.camera.PhotoModule$CameraStartUpThread.run(PhotoModule.java:318)
E/AndroidRuntime( 1247): Caused by: com.android.camera.CameraHardwareException: java.lang.RuntimeException: Fail to connect to camer
a service
E/AndroidRuntime( 1247):        at com.android.camera.CameraHolder.open(CameraHolder.java:219)
E/AndroidRuntime( 1247):        at com.android.camera.Util.openCamera(Util.java:315)
E/AndroidRuntime( 1247):        ... 1 more
E/AndroidRuntime( 1247): Caused by: java.lang.RuntimeException: Fail to connect to camera service
E/AndroidRuntime( 1247):        at android.hardware.Camera.native_setup(Native Method)
E/AndroidRuntime( 1247):        at android.hardware.Camera.<init>(Camera.java:340)
E/AndroidRuntime( 1247):        at android.hardware.Camera.open(Camera.java:302)
E/AndroidRuntime( 1247):        at com.android.camera.CameraManager.cameraOpen(CameraManager.java:283)
E/AndroidRuntime( 1247):        at com.android.camera.CameraHolder.open(CameraHolder.java:210)
E/AndroidRuntime( 1247):        ... 2 more
W/ActivityManager(  359):   Force finishing activity com.android.gallery3d/com.android.camera.CameraLauncher
```

（1）、在 shell 中执行 ls dev 命令查看是否存在 camera 相应的设备节点 video0、video1；

（2）、如果不存在，那么确认按照新的设备注册方式注册的设备在 IO、i2c 、cif 的配置上是否有误；

（3）、将新版的 sensor 驱动替换成项目工程的原来 sensor 驱动，然后检查问题是否存在；

（4）、如果出现前后摄像头切换出问题的，也麻烦步骤 1—3 确认；

20 RK_SWISP 使用说明

1：android 配置

Android4.1:

1：到目录 andorid4.1/device/rockchip/rk30sdk/打补丁包中相应位置的补丁 rkswisp.patch

如果补丁打不上，请对比 andorid4.1/device/rockchip/rk30sdk/文件夹下的修改，修改相应文件

2：重新编译 android,生成新固件。重新烧写固件

Androdi 4.2:

1: 到 andorid4.2 的目录/device/rockchip/rk30sdk/打补丁包中相应位置的补丁 rkswisp_rk30sdk.patch

如果补丁打不上，请对比 andorid4.1/device/rockchip/common/文件夹下的修改，修改相应文件

2: 到 andorid4.2 的目录/device/rockchip/common/打补丁包中相应位置的补丁 rkswisp_common.patch

如果补丁打不上，请对比 andorid4.1/device/rockchip/rk30sdk/文件夹下的修改，修改相应文件

3:重新编译 android,生成新固件。重新烧写固件

2：kernel 配置

将补丁包中文件一一替换到 kernel 里的文件，重新编译 kernel，重新烧写 kernel.img。

RK_SwIsp 效果微调

1：如果客户对模组的色彩饱和度和对比度不满意，可以通过串口命令进行调节，调节后必须重新打开 camera 才会生效。

命令为：

echo '20' > sys/module/ov5647/parameters/saturation　　色彩饱和度增加 20%

echo '-10' > sys/module/ov5647/parameters/contrast　　对比度减少 10%

取值范围为-100~100

Ov5647sensor 输出帧率有两种选择，15fps or 30fps

15fps 的预览画质相对好一些。30fps 的预览画质相对差一些。

客户可以自行选择，修改下面显示的宏定义。

```
#define OV5647_USE_30_OR_15_FPS          0 // 1: 30fps,preview picture bad
                                           // 0: 15fps,preview picture better
```

21 关于 OV5640/5642 上电后马达有咔的响声的说明

针对近期出现的个别摄像头模组（如 ov5640、ov5642），在开机过程中马达会预先启动的问题，硬件上增加马达电源控制电路，软件上控制 AF_VDD28 电源在 camera sensor 初始化完成之后再上电。

SDK 上实现的预留 IO 如下(具体请参考原理图)：

1、RK3066 : GPIO3_B7/SDMMC0_WP (默认下拉口)

2、RK3188 : GPIO3_B1/SDMMC0_WP (默认下拉口)

客户也可根据需要选择自己的 IO。

以下适用于 baord 文件新的设备注册方式。

arch/arm/plat-rk/include/plat/rk_camera.h 中增加注册初始化 AF 宏：

```
00215:
00216: #define new_camera_device_af(af_io)\
00217:     .io = {\
00218:         .gpio_af = af_io\
00219:     },
00220:
```

可扩展宏定义：

#define new_camera_device_af(af_io)

af_io: 马达电源控制 IO

若要进行控制，只需填入相应的 IO 引脚。

具体使用方式参见上文“4.2 如何注册一个 sensor 设备”中的可扩展注册章节。

如果客户想自定义接口，可按照如下方式添加(目前只实现了 board-rk30-sdk-camera.c 和 board-rk3168-tb-camera.c)：

(1)在 board 文件中添加宏：

```
#define CONFIG_SENSOR_AF_IOCTL_USR      0
```

(2)添加接口定义：

```
#if CONFIG_SENSOR_AF_IOCTL_USR
static int sensor_af_usr_cb (struct rk29camera_gpio_res *res,int on)
{
    #error "CONFIG_SENSOR_AF_IOCTL_USR is 1, sensor_af_usr_cb function must be writed!!";
}
#endif
```

(3)callback 赋值：

```
static struct rk29camera_platform_ioctl_cb  sensor_ioctl_cb = {
    #if CONFIG_SENSOR_POWER_IOCTL_USR
    .sensor_power_cb = sensor_power_usr_cb,
    #else
    .sensor_power_cb = NULL,
    #endif

    #if CONFIG_SENSOR_RESET_IOCTL_USR
    .sensor_reset_cb = sensor_reset_usr_cb,
    #else
    .sensor_reset_cb = NULL,
    #endif

    #if CONFIG_SENSOR_POWERDOWN_IOCTL_USR
    .sensor_powerdown_cb = sensor_powerdown_usr_cb,
    #else
    .sensor_powerdown_cb = NULL,
    #endif

    #if CONFIG_SENSOR_FLASH_IOCTL_USR
    .sensor_flash_cb = sensor_flash_usr_cb,
    #else
    .sensor_flash_cb = NULL,
    #endif

    #if CONFIG_SENSOR_AF_IOCTL_USR
    .sensor_af_cb = sensor_af_usr_cb,
    #else
    .sensor_af_cb = NULL,
    #endif
};
```

打开宏开关 CONFIG_SENSOR_AF_IOCTL_USR，实现接口即可。例如,实现为：

```
static int sensor_af_usr_cb (struct rk29camera_gpio_res *res,int on)
{
    int camera_af = res->gpio_af;
    if (camera_af != INVALID_GPIO) {
        gpio_set_value(camera_af, on);
    }
}
```