密级状态：绝密(　　)　秘密(　　)　内部资料(　　)　公开(√)

RKISPV11_Camera_ISP_User_Manual

文件状态： ［　］草稿 ［　］正式发布 ［√］正在修改	文件标识：	
	当前版本：	1.0
	作　　者：	张云龙、黄春成
	完成日期：	2016-9-6

历 史 版 本

版本	日 期	描 述	作 者	审核
V1.0	2016-9-6	建立文档，主要介绍 RK1108 CVR Camera 的注意事项	张云龙、黄春成	

目录

1. 文档适用平台

文档适用于 RockChips 公司 RK1108 CVR 平台。

2. 硬件说明

2.1. MIPI Camera Sensor

(模组的 MIPI Lane 数 >= PHY 支持的 MIPI Lane 数） 满足这一条件都可以连接到对应的 PHY，但是最后实际使用的 Lane 数以 PHY 支持的 Lane 数为准；

MIPI Camera Sensor 在选用时，建议事先查阅 RockChip 的认证列表：《RKISPV11_Camera_Module_AVL》，确认是否调试通过.

2.2. Camera Sensor 选型说明

1、事先获取 RockChip 的认证列表：《RKISPV11_Camera_Module_AVL》；
2、列表中已经有相关型号，并且状态显示 Ready，那么建议按照列表中的模组配置信息让模组厂进行打样；
3、列表中没有相关型号，或是想选择不同配置（镜头、VCM）的模组，那么建议填写《RockChip 摄像头模组调试需求申请表》，同时发给 RockChip。

注：Camera Sensor 调试周期在 4 周左右；
模组配置更换 调试周期在 3 周左右；

3. 文件目录说明

Rk1108_cvr：

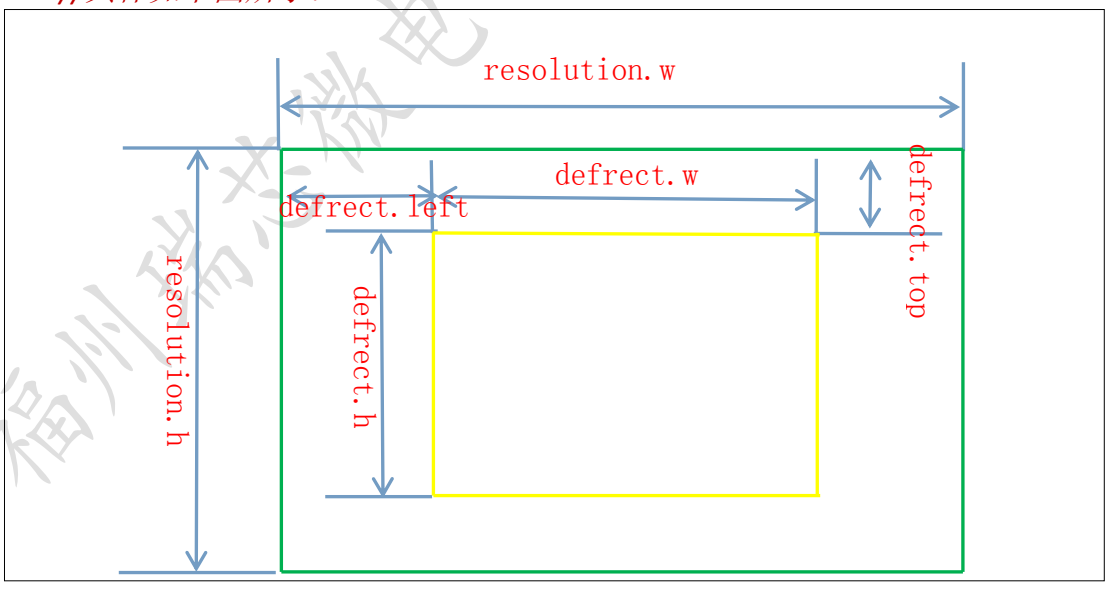
```
|
| kernel
    |
    |arch/arm/boot/dts                dts 配置文件
    |drivers/media
        |
        |platform/rk-isp11            ISP host 驱动
        |i2c/soc_camera/rockchip/     camera sensor 驱动
|external
    |
    |libcamerahal/out
        |
        |inc/CameraHal                camerhal 头文件
        |lib /libcam_hal.so           cameahal 库文件
```

4. MIPI Sensor 注册

rk1108_cvr.dts 文件中：

```
&i2c1 {
    status = "okay";//是否加载模块，默认开启
    camera0: camera-module@36 {
        status = "okay";//是否加载模块，默认开启
        compatible = "omnivision,ov2710-v4l2-i2c-subdev";
        //omnivision sensor 类型
        //ov2710-v4l2-i2c-subdev 中 ov2710 为 sensor 型号
        //需要与驱动名字一致，具体见章节错误！未找到引用源。
        reg = <0x36>;// Sensor I2C 设备地址
        device_type = "v4l2-i2c-subdev";//设备类型
        clocks = <&clk_mipicsi_out>;//sensor clickin 配置
        clock-names = "clk_mipicsi_out";
        pinctrl-names = "rockchip,camera_default", "rockchip,camera_sleep";
        pinctrl-0 = <&cam0_default_pins>;
        pinctrl-1 = <&cam0_sleep_pins>;
        rockchip,pd-gpio = <&gpio3 8 GPIO_ACTIVE_HIGH>;
        //Sensor PowerDown GPIO 配置
        rockchip,pd-gpio = <&gpio3 8 GPIO_ACTIVE_LOW>;
        //powerdown 的管脚分配及有效电平
        rockchip,pwr-gpio = <&gpio0 18 GPIO_ACTIVE_HIGH>;
        //power 的管脚分配及有效电平
        rockchip,rst-gpio = <&gpio3 25 GPIO_ACTIVE_LOW>;
        //reset 的管脚分配及有效电平
        rockchip,camera-module-mclk-name = "clk_mipicsi_out";//mclk 时钟源配置
        //在 rk1108-clocks.dtsi 中 clk_mipicsi_out 的时钟源为 xin24m，
        clk_mipicsi_out: clk_mipicsi_out_div {
            ......
            clocks = <&xin24m>;
            ......
        };
        xin24m: xin24m {
            compatible = "rockchip,rk-fixed-clock";
            clock-output-names = "xin24m";
            clock-frequency = <24000000>;//24MHZ
            #clock-cells = <0>;
        };
        在 PLTFRM_CAM_ITF_MIPI_CFG(dphy_index, vc , nb_lanes, bit_rate)，
        bit_rate 进行设置 Mclk。
        当信号为并口时，时钟源应该选择“clk_cif_out”
        rockchip,camera-module-regulator-names = "vdd_cam", "avdd_cam";
        //camera vdd 和 avdd 名称
        rockchip,camera-module-regulator-voltages = <1450000 3300000>;
        //camera vdd 电压：0x1.45V avdd 电压：0x3.30V
```

```
        rockchip,camera-module-facing = "back";//前后置配置
        rockchip,camera-module-name = "LA6110PA";//Camera 模组名称
        rockchip,camera-module-len-name = "YM6011P";//Camera 模组镜头
        rockchip,camera-module-fov-h = "128";//模组水平可视角度配置
        rockchip,camera-module-fov-v = "55.7";//模组垂直可视角度配置
        rockchip,camera-module-orientation = <0>;//模组角度设置
        rockchip,camera-module-iq-flip = <0>;//IQ 上下翻转
        rockchip,camera-module-iq-mirror = <0>;//IQ 左右镜像
        //以上 2 个属性控制摄像头的效果参数镜像配置，一般都是设置成 0，但 是发现以下现象：
        拍摄白墙，图片的上半部偏色与下半部偏色不一致，或者左右半部偏色不一致，即可以将这 2 个属性置成 1；
        rockchip,camera-module-flip = <0>;
        rockchip,camera-module-mirror = <0>;
        //以上 2 个属性控制摄像头驱动中的镜像配置，如果图像旋转 180 度，可以将这 2 个属性修改成相反的值即可旋转 180；
        rockchip,camera-module-defrect0 = <1920 1080 0 0 1920 1080>;
       // resolution.w：sensor 输出列数,
        //resolution.h:sensor 输出行数,
        // defrect.left:输出偏移列数,
        // defrect.top:输出偏移行数,
        // defrect.w:输出列数, defrect.left+defrect.w<=resolution.w,
        //defrect.h:输出行数,defrect.h+defrect.top<=resolution.h,
        //具体如下图所示：
```




```
        rockchip,camera-module-flash-support = <0>;//flash 控制开关
        rockchip,camera-module-mipi-dphy-index = <0>;
        //sensor 实际使用的 phy,要与硬件实际连接对应
};
        ......
cameraN: camera-module@addrN {
```

```
        //支持多个 camera 配置
            ......
        };
};
&cif_isp0 {
        rockchip,camera-modules-attached = <&camera0 &camera1 &camera2>;
        //配置需要使用的 camera 列表,连接到 ISP 设备节点
        status = "okay";
};
```

5. Camera 设备驱动

5.1. 驱动框架

rk1108_cvr.dts 文件中：

```
&i2c1 {
        status = "okay";
        camera0: camera-module@36 {
            status = "okay";
            ......
        }
        ......
        CameraN: camera-module@addrN {
            status = "okay";
            ......
        }
}
```

Sensor 设备驱动采用 i2c 设备驱动方式，因此驱动中以 struct i2c_driver 的驱动架构实现，如下代码：

```
static struct i2c_driver ov4689_i2c_driver = {
    .driver = {
        .name = ov4689_DRIVER_NAME,
        .owner = THIS_MODULE,
        .of_match_table = ov4689_of_match
    },
    .probe = ov4689_probe,
    .remove = ov4689_remove,
    .id_table = ov4689_id,
};
```

其中 i2c_driver 中有.driver、.probe、.remove、.id_table 四个子成员。.driver 主要是标识名称、.probe 主要是用于 sensor run 涉及的处理函数，.remove 移出当前设备，释放相应空间。

5.2. 驱动解析

1) camera 匹配

在.driver 中有 3 个子成员，.name 为当前设备的名称，不是一个实体，仅用来标识；.owner 指针指向的当前的这个 module；而.of_match_table 中定义的字符串为 dts 文件中注册设备 compatible 需要匹配；如下：

驱动程序中

```
static struct of_device_id ov4689_of_match[] = {
    {.compatible = "omnivision,ov4689-v4l2-i2c-subdev"},
    {},
};
```

rk1108_cvr.dts 文件中

```
camera1: camera-module@36-1 {
    status = "disabled";
    compatible = "omnivision,ov4689-v4l2-i2c-subdev";
    reg = <0x36>;
    device_type = "v4l2-i2c-subdev";
```

2) 其他设备解析

在 Rk_camera_module.c 文件中有如下：

```
#define OF_OV_GPIO_PD "rockchip,pd-gpio"
#define OF_OV_GPIO_PWR "rockchip,pwr-gpio"
#define OF_OV_GPIO_FLASH "rockchip,flash-gpio"
#define OF_OV_GPIO_TORCH "rockchip,torch-gpio"
#define OF_OV_GPIO_RESET "rockchip,rst-gpio"

#define OF_CAMERA_MODULE_NAME "rockchip,camera-module-name"
#define OF_CAMERA_MODULE_LEN_NAME "rockchip,camera-module-len-name"
#define OF_CAMERA_MODULE_FOV_H "rockchip,camera-module-fov-h"
#define OF_CAMERA_MODULE_FOV_V "rockchip,camera-module-fov-v"
#define OF_CAMERA_MODULE_ORIENTATION "rockchip,camera-module-orientation"
#define OF_CAMERA_MODULE_FOCAL_LENGTH "rockchip,camera-module-focal-length"
#define OF_CAMERA_MODULE_FOCUS_DISTANCE "rockchip,camera-module-focus-distance"
#define OF_CAMERA_MODULE_IQ_MIRROR "rockchip,camera-module-iq-mirror"
#define OF_CAMERA_MODULE_IQ_FLIP "rockchip,camera-module-iq-flip"
#define OF_CAMERA_MODULE_FLIP "rockchip,camera-module-flip"
#define OF_CAMERA_MODULE_MIRROR "rockchip,camera-module-mirror"
#define OF_CAMERA_FLASH_SUPPORT "rockchip,camera-module-flash-support"
#define OF_CAMERA_FLASH_EXP_PERCENT "rockchip,camera-module-flash-exp-percent"
#define OF_CAMERA_FLASH_TURN_ON_TIME "rockchip,camera-module-flash-turn-on-time"
#define OF_CAMERA_FLASH_ON_TIMEOUT "rockchip,camera-module-flash-on-timeout"
#define OF_CAMERA_MODULE_DEFRECT0 "rockchip,camera-module-defrect0"
#define OF_CAMERA_MODULE_DEFRECT1 "rockchip,camera-module-defrect1"
#define OF_CAMERA_MODULE_DEFRECT2 "rockchip,camera-module-defrect2"
#define OF_CAMERA_MODULE_DEFRECT3 "rockchip,camera-module-defrect3"
#define OF_CAMERA_MODULE_MIPI_DPHY_INDEX "rockchip,camera-module-mipi-dphy-index"

#define OF_CAMERA_MODULE_REGULATORS "rockchip,camera-module-regulator-names"
#define OF_CAMERA_MODULE_REGULATOR_VOLTAGES "rockchip,camera-module-regulator-voltages"
#define OF_CAMERA_MODULE_MCLK_NAME "rockchip,camera-module-mclk-name"
```

这些设备与章节**错误！未找到引用源。**相对应，详细对应的节点看其中的内容介绍。

5. 3. Camera 驱动介绍

驱动文件分类，主要按不同类型的 sensor 区别，如下有：

Aptina Camera Sensor：ar0330cs_v4l2-i2c-subdev.c　imx_camera_module.c

OminiVision Camera Sensor：ov4689_v4l2-i2c-subdev.c　ov_camera_module.c

公共的函数放在 Rk_camera_Module.c。其中一些关键的函数、结构体解析如下介绍：

1) struct ar0330cs_custom_config 结构体信息：

```
static struct aptina_camera_module_custom_config ar0330cs_custom_config = {
    .start_streaming = ar0330cs_start_streaming,
    .stop_streaming = ar0330cs_stop_streaming,
    .s_ctrl = ar0330cs_s_ctrl,
    .s_ext_ctrls = ar0330cs_s_ext_ctrls,
    .g_ctrl = ar0330cs_g_ctrl,
    .g_timings = ar0330cs_g_timings,
    .check_camera_id = ar0330cs_check_camera_id,
    .set_flip = ar0330cs_set_flip,
    .configs = ar0330cs_configs,
    .num_configs = sizeof(ar0330cs_configs) / sizeof(ar0330cs_configs[0]),
    .power_up_delays_ms = {5, 20, 0}
};
```

. start/stop_streaming *//Sensor 启动/停止数据流接口*

. s_ctrl/ g_ctrl *//设置/获取 sensor 相关参数，目前主要是 aec 设置，*

Aec：用自动曝光设置接口函数如下：

*static int ar0330cs_write_aec(struct aptina_camera_module *cam_mod)*

.check_camera_id*//校验 sensor id 接口，在模块初始化进行校验*

.set_flip*//sensor 镜像控制*

*接口函数：static int ov4689_set_flip(struct ov_camera_module *cam_mod,struct pltfrm_camera_module_reg reglist[],int len)*

.configs*//sensor 配置信息，具体配置信息如下示例：*

.power_up_delays_ms*//power up 延时设置详细见*

2) struct aptina_camera_module_config 结构体信息

```
static struct aptina_camera_module_config ar0330cs_configs[] = {
    {
        .name = "2048_1536_30fps",
        .frm_fmt = {
            .width = 2048,
            .height = 1536,
            .code = V4L2_MBUS_FMT_SGRBG10_1X10
        },
        .frm_intrvl = {
            .interval = {
                .numerator = 1,
                .denominator = 30
            }
        },
        .auto_exp_enabled = false,
        .auto_gain_enabled = false,
        .auto_wb_enabled = false,
        .reg_table = (void *)ar0330cs_init_tab_2048_1536_30fps,
        .reg_table_num_entries =
            sizeof(ar0330cs_init_tab_2048_1536_30fps)
            /
            sizeof(ar0330cs_init_tab_2048_1536_30fps[0]),
        .v_blanking_time_us = 5000,
        PLTFRM_CAM_ITF_MIPI_CFG(0, 2, 98, AR0330CS_EXT_CLK)
    }
};
```

. frm_fmt *//Sensor 支持的分辨率，可根据实际 sensor 进行修改，其中*

.width：sensor 输出列数，

.height：sensor 输出行数，

.code：sensor 输出数据格式

. frm_intrvl *//帧率信息*

.denominator：sensor 输出帧率

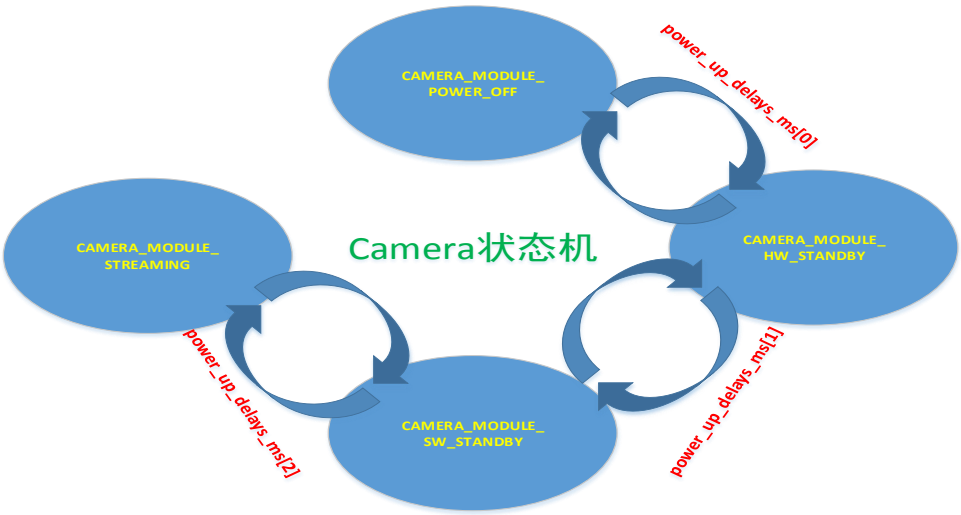
.auto_exp_enabled*//自动曝光使能*

曝光设置接口函数：

*static int ov4689_write_aec(struct ov_camera_module *cam_mod)*

. reg_table *//Sensor 各个功能的寄存器设置信息，初始化序列，移植时，要根据实际 sensor 对应的初始化序列进行修改*
PLTFRM_CAM_ITF_MIPI_CFG(dphy_index, vc ,bit_rate, nb_lanes,)
//dphy_index:dphy 选择，
vc:虚拟通道，
bit_rate:sensor mipi 传输带宽
nb_lanes:数据 lane 数，

3) Camera 状态机



上图为 camera 状态机跳转，
CAMERA_MODULE_POWER_OFF：camera 掉电状态
CAMERA_MODULE_HW_STANDBY：camera 上电状态
CAMERA_MODULE_SW_STANDBY：cameara PD\RESET 设置完成时的状态
CAMERA_MODULE_STREAMING：camera 正常 run 状态
在这 4 个状态跳转过程中需要一定延时等待，具体延时时间如 1)所示：.power_up_delays_ms = {5, 20, 0}

4) dts 配置：
根据实际的硬件连接、sensor 要求，修改上章节**错误！未找到引用源。**的各项信息，尤其注意 i2c 地址、pinctrl、DVDD 配置、pd、reset 引脚及其有效电平。

5) 上下电
int gc_camera_module_s_power(struct v4l2_subdev *sd, int on)
该函数中主要实现：

1. 按照 sensor 上电要求，初始化 dts 文件中配置的 GPIO（PowerDown、Rest）；
2. 调用 pltfrm_camera_module_s_power 通知电源管理模块控制 camera 相关电源；
3. 上电时调用 check_camera_id 校验 Sensor id；

6. I2C 问题排查

1．根据样机原理图、模组规格书、sensor datasheet 检查硬件：确认样机到模组的电源、power down、reset 连接是否正确。

2．电源检查：一般需要三路电源 AVDD、DVDD、DOVDD，确认供电是否符合 sensor datesheet 要求，特别是 DVDD 是否符合要求；

3．power down 脚检查：检查 cam0 和 cam1 的连接是否正确，是否有接反的情况，有效电平的控制是否符合要求；

power down 引脚的控制有 Camera driver 来控制

4．reset 脚检查：检查 reset 连接是否正确，是否有未连接的情况，有效电平是否符合要求；

5．i2c 通道是否正确，i2c 的设备地址是否配置正确；

6．i2c 访问时检查 mclk 配置是否正确，输出占空比为 50%的 24Mhz 时钟；